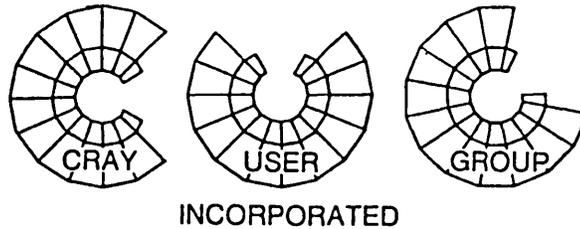


SAN DIEGO
1994

Spring Proceedings

Cray User Group, Inc.





PROCEEDINGS

Thirty-Third Semi-Annual Cray User Group Meeting

San Diego, California
March 14–18, 1994

No part of this publication may be reproduced by any mechanical, photographic, or electronic process without written permission from the author(s) and publisher. Permission requests should be made in writing to:

Cray User Group Secretary
c/o Joan Palm
655A Lone Oak Drive
Eagan, MN 55121 USA

The CUG *Proceedings* is edited and produced by Karen Winget,
Fine Point Editorial Services, 1011 Ridge Valley Court, Shepherdstown, WV 25443

Cover designed by Carol Conway Leigh

Autotasking, CF77, CRAY, Cray Ada, CRAY Y-MP, CRAY-1, HSX, MPGS, SSD, SUPERSERVER, UniChem, UNICOS, and X-MP EA are federally registered trademarks and CCI, CFT, CFT2, CFT77, COS, CRAY APP, Cray C++ Compiling System, CRAY C90, CRAY EL, Cray NQS, CRAY S-MP, CRAY T3D, CRAY X-MP, CRAY XMS, CRAY-2, Cray/REELlibrarian, CRInform, CRVTurboKiva, CSIM, CVT, Delivering the power . . . , Docview, EMDS, IOS, MPP Apprentice, Network Queuing Environment, Network Queuing Tools, OLNET, RQS, SEGLDR, SMARTE, SUPERCLUSTER, SUPERLINK, Trusted UNICOS, and UNICOS MAX are trademarks of Cray Research, Inc.

Cray User Group, Inc.

BOARD OF DIRECTORS

President

Gary Jensen (NCAR)

Vice-President

Jean Shuler (NERSC)

Secretary

Gunter Georgi (Grumman)

Treasurer

Howard Weinberger (TAI)

Director

Claude Lecoivre (CEL-V)

Regional Director Americas
Fran Pellegrino (Westinghouse)

Regional Director Asia/Pacific
Kyukichi Ohmura (CRC)

Regional Director Europe
Walter Wehinger (RUS)

PROGRAM COMMITTEE

Jean Shuler, Chair (NERSC)
Robert Price (Westinghouse)
Helene Kulsrud (IDA)
Phil Cohen (Scripps)

LOCAL ARRANGEMENTS

Anke Kamrath
Ange Mason
Mike Vildibill
Jay Dombrowski
Phil Cohen
Dan Drobnis
Gail Bamber
Russ Sinco
Sandy Davey
Jayne Keller

CONTENTS

GENERAL SESSIONS

Chairman's Report <i>John F. Carlson (CRI)</i>	3
Cray Research Corporate Report <i>Robert Ewald (CRI)</i>	6
CRI Software Report <i>Irene Qualters (CRI)</i>	14
Unparalleled Horizons: Computing in Heterogeneous Environments <i>Reagan Moore (SDSC)</i>	21
Cray T3D Project Update <i>Steve Reinhardt (CRI)</i>	28

PARALLEL SESSIONS

Applications and Algorithms

Porting Third-Party Applications Packages to the Cray MPP: Experiences at the Pittsburgh Supercomputing Center <i>Frank C. Wimberley, Susheel Chitre, Carlos Gonzales, Michael H. Lambert, Nicholas Nystrom, Alex Ropelewski, William Young (PITTSCC)</i>	39
Some Loop Collapse Techniques to Increase Autotasking Efficiency <i>Mike Davis (CRI)</i>	44
Collaborative Evaluation Project of Ingres on the Cray (CEPIC) <i>C.B. Hale, G.M. Hale, and K.F. Witte (LANL)</i>	65
Providing Breakthrough Gains: Cray Research MPP for Commercial Applications <i>Denton A. Olson (CRI)</i>	73
Asynchronous Double-Buffered I/O Applied to Molecular Dynamics Simulations of Macromolecular Systems <i>Richard J. Shaginaw, Terry R. Stouch, and Howard E. Alper (BMSPRI)</i>	76
A PVM Implementation of a Conjugate Gradient Solution Algorithm for Ground-Water Flow Modeling <i>Dennis Morrow, John Thorp, and Bill Holter (CRI and NASA/Goddard)</i>	80

Graphics

Decimation of Triangle Meshes <i>William J. Schroeder (GE-AE)</i>	87
--	----

Visualization of Volcanic Ash Clouds <i>Mitchell Roth (ARSC) and Rick Guritz (Alaska Synthetic Aperture Radar Facility)</i>	92
A Graphical User Interface for Networked Volume Rendering on the Cray C90 <i>Allan Snaveley and T. Todd Elvins (SDSC)</i>	100
<i>Management</i>	
Heterogeneous Computing Using the Cray Y-MP and T3D <i>Bob Carruthers (CRI)</i>	109
Future Operating Systems Directions <i>Don Mason (CRI)</i>	112
Future Operating Systems Directions—Serverized UNICOS <i>Jim Harrell (CRI)</i>	114
<i>Mass Storage Systems</i>	
Storage Management Update <i>Brad Strand (CRI)</i>	119
RAID Integration on Model-E IOS <i>Bob Ciotti (NASA-Ames)</i>	123
Automatic DMF File Expiration <i>Andy Haxby (Shell U.K.)</i>	136
ER90@Data Storage Peripheral <i>Gary R. Early and Anthony L. Peterson (ESYSTEMS)</i>	140
EMSS/Cray Experiences and Performance Issues <i>Anton L. Ogno (EXXON)</i>	143
AFS Experience at the Pittsburgh Supercomputing Center <i>Bill Zumach (PITTSC)</i>	153
AFS Experience at the University of Stuttgart <i>Uwe Fischer and Dieter Mack (RUS)</i>	157
Cray Research Status of the DCE/DFS Project <i>Brian Gaffey (CRI)</i>	161
<i>Networking</i>	
SCinet '93—An Overview <i>Benjamin R. Peek (Peek & Associates, Inc.)</i>	169
Experiences with OSF-DCE/DFS in a 'Semi-Production' Environment <i>Dieter Mack (RUS)</i>	175
ATM—Current Status <i>Benjamin R. Peek (Peek & Associates, Inc.)</i>	180
Network Queuing Environment <i>Robert E. Daugherty and Daniel M. Ferber (CRI)</i>	203

Operating Systems

Livermore Computing's Production Control System, 3.0 <i>Robert R. Wood (LLNL)</i>	209
ALACCT: Limiting Activity Based On ACID <i>Sam West (CRI)</i>	212
Priority-Based Memory Scheduling <i>Chris Brady (CRI), Don Thorp (CRI), and Jeff Pack (GRUMMAN)</i>	221
Planning and Conducting a UNICOS Operating System Upgrade <i>Mary Ann Ciuffini (NCAR)</i>	224
UNICOS Release Plans (1994–1995) <i>Janet Lebens (CRI)</i>	231
UNICOS 8.0 Experiences <i>Hubert Busch (ZIB)</i>	237
UNICOS 8.0—Field Test Experiences <i>Douglas A. Spragg (EUTEC)</i>	239

Operations

SDSC Host Site Presentation <i>Daniel D. Drobnis and Michael Fagan (SDSC)</i>	247
Tools for the Cray Research OWS-E Operator under UNICOS 8.0 <i>Leon Vann (CRI)</i>	251
Cray Research Product Resiliency <i>Gary Shorrel (CRI)</i>	255

Performance Evaluation

UNICOS 7.C versus 8.0 Test Results <i>C.L. Berkey (CRI)</i>	260
Workload Characterization of Cray Supercomputer Systems Running UNICOS for the Optimal Design of NQS Configuration in a Site <i>Young W. Lee and Yeong Wook Cho (KIST) and Alex Wight (Edinburgh University)</i>	271
Cray C90D Performance <i>David Slowinski</i>	291

Software Tools

Cray File Permission Support Toolset <i>David H. Jennings and Mary Ann Cummings (Naval Surface Warfare Center)</i>	295
Tools for Accessing Cray Datasets on Non-Cray Platforms <i>Peter W. Morreale (NCAR)</i>	300
Centralized User Banking and User Administration on UNICOS <i>Morris A. Jette, Jr., and John Reynolds (NERSC)</i>	304

Fortran 90 Update	
<i>Jon Steidel (CRI)</i>	313
C and C++ Programming Environments	
<i>David Knaak (CRI)</i>	320
The MPP Apprentice™ Performance Tool: Delivering the Performance of the Cray T3D®	
<i>Winifred Williams, Timothy Hoel, and Douglas Pase (CRI)</i>	324
Cray Distributed Programming Environment	
<i>Lisa Krause (CRI)</i>	333
Cray TotalView™ Debugger	
<i>Dianna Crawford (CRI)</i>	338
Fortran I/O Libraries on T3D	
<i>Suzanne LaCroix (CRI)</i>	344

User Services

User Contact Measurement Tools and Techniques	
<i>Ted Spitzmiller (LANL)</i>	349
Balancing Services to Meet the Needs of a Diverse User Base	
<i>Kathryn Gates (MCSR)</i>	353
Applications of Multimedia Technology for User Technical Support	
<i>Jeffrey A. Kuehn (NCAR)</i>	360
Integrated Performance Support: Cray Research's Online Information Strategy	
<i>Marjorie L. Kyriopoulos (CRI)</i>	365
New and Improved Methods of Finding Information via CRINFORM	
<i>Patricia A. Tovo (CRI)</i>	369
Online Documentation: New Issues Require New Processes	
<i>Juliana Rew (NCAR)</i>	372
MetaCenter Computational Science Bibliographic Information System: A White Paper	
<i>Mary Campana (Consultant) and Stephanie Sides (SDSC)</i>	376
Electronic Publishing: From High-Quality Publications to Online Documentation	
<i>Christine Guzy (NCAR)</i>	380

JOINT SESSIONS

MSS/Operating Systems

Workload Metrics for the NCAR Mass Storage System	
<i>J.L. Sloan (NCAR)</i>	387
Scientific Data Storage Solutions: Meeting the High-Performance Challenge	
<i>Daniel Krantz, Lynn Jones, Lynn Kluegel, Cheryl Ramsey, and William Collins (LANL)</i>	392
Beyond a Terabyte System	
<i>Alan K. Powers (SS-NAD)</i>	402

Operations/Environmental MIG

Overview of Projects, Porting, and Performance of Environmental Applications

Tony Meys (CRI)

411

Operations/MPP MIG

T3D SN6004 is Well, Alive and Computing

Martine Gigandet, Monique Patron, Francois Robin (CEA-CEL)

419

System Administration Tasks and Operational Tools for the Cray T3D System

Susan J. Crawford (CRI)

426

Performance Evaluation/Applications and Algorithms

I/O Optimisation Techniques Under UNICOS

Neil Storer (ECMWF)

433

I/O Improvements in a Production Environment

Jeff Zais and John Bauer (CRI)

442

New Strategies for File Allocation on Multi-Device File Systems

Chris Brady (CRI), Dennis Colarelli (NCAR),

Henry Newman (Instrumental), and Gene Schumacher (NCAR)

446

Performance Evaluation/MPP MIG

The Performance of Synchronization and Broadcast Communication on the Cray T3D Computer System

F. Ray Barriuso (CRI)

455

High Performance Programming Using Explicit Shared Memory Model on the Cray T3D

Subhash Saini, Horst Simon (CSC—NASA/Ames) and Charles Grassl (CRI)

465

Architecture and Performance for the Cray T3D

Charles M. Grassl (CRI)

482

User Services/Software Tools

Confessions of a Consultant

Tom Parker (NCAR)

491

SHORT PAPERS

Xnewu: A Client-Server Based Application for Managing Cray User Accounts

Khalid Warraich and Victor Hazlewood (Texas A&M)

505

QEXEC: A Tool for Submitting a Command to NQS

Glenn Randers-Pehrson (USARL)

507

ATTENDEE LIST

513

AUTHOR INDEX

- Alper, H.E. (Bristol-Meyers Squibb), 76
Barriuso, F.R. (CRI), 455
Bauer, J. (CRI), 442
Berkey, C.L. (CRI), 260
Brady, C. (CRI), 221, 446
Busch, H. (ZIB), 237
Campana, M. (SDSC consultant), 376
Carlson, J.F. (CRI), 3
Carruthers, B. (CRI), 109
Chitre, S. (PITTSC), 39
Cho, Y.W. (KIST), 271
Ciotti, B. (NASA/Ames), 123
Ciuffini, M. (NCAR), 224
Colarelli, D. (NCAR), 446
Collins, W. (LANL), 392
Crawford, D. (CRI), 338
Crawford, S.J. (CRI), 426
Cummings, M., (Naval Surface Warfare Center), 295
Daugherty, R.E. (CRI), 203
Davis, M. (CRI), 44
Drobnis, D.D. (SDSC), 247
Early, G.R. (ESYSTEMS), 140
Elvins, T.T. (SDSC), 100
Ewald, R. (CRI), 6
Fagan, M. (SDSC), 247
Ferber, D.M., 203
Fischer, U. (RUS), 157
Gaffey, B. (CRI), 161
Gates, K. (MCSR), 353
Gigandet, M. (CEA-CEL), 419
Gonzales, C. (PITTSC), 39
Grassl, C. (CRI), 465, 482
Guritz, R. (Alaska Synthetic Aperture Radar Facility), 92
Guzy, C. (NCAR), 380
Hale, C.B. (LANL), 65
Hale, G.M. (LANL), 65
Harrell, J. (CRI), 114
Haxby, A. (Shell U.K.), 136
Hazlewood, V. (Texas A&M), 505
Hoel, T. (CRI), 324
Holter, B. NASA/Goddard), 80
Jennings, D.H. (Naval Surface Warfare Center), 295
Jette, M.A., Jr. (NERSC), 304
Jones, L. (LANL), 392
Kluegel, L. (LANL), 392
Knaak, D. (CRI), 320
Krantz, D. (LANL), 392
Krause, L. (CRI), 333
Kuehn, J.A. (NCAR), 360
Kyriopoulos, M.L. (CRI), 365
Labens, J. (CRI) 231
LaCroix, S. (CRI), 340
Lambert, M.H. (PITTSC), 39
Lee, Y.W. (KIST), 271
Mack, D. (RUS), 157, 175
Mason, D. (CRI), 112
Meys, T. (CRI), 411
Moore, R. (SDSC), 21
Morreale, P. (NCAR), 300
Morrow, D., (CRI)??, 80
Newman, H. (Instrumental), 446
Nystrom, N. (PITTSC), 39
Ogno, A.L. (EXXON), 143
Olson, D.A. (CRI), 73
Pack, J. (GRUMMAN)), 221
Parker, T. (NCAR), 491
Pase, D. (CRI), 324
Patron, M. (CEA-CEL), 419
Peek, B.R. (Peek & Assoc.), 169, 180
Peterson, A.L. (ESYSTEMS), 140
Powers, A.K. (SS-NAD), 402
Qualters, I. (CRI), 14
Ramsey, C. (LANL), 392
Randers-Pehrson, G. (USARL), 507
Reinhardt, S. (CRI), 28
Rew, J. (NCAR), 372
Reynolds, J. (NERSC), 304
Robin, F. (CEA-CEL), 419
Ropelewski, A. (PITTSC), 39
Roth, M. (ARSC), 92
Saini, S. (CSC-NASA/Ames), 465
Schroeder, W.J. (GE-AE), 87
Schumacher, G. (NCAR), 446
Shaginaw, R.J. (Bristol-Meyers Squibb), 76
Shorrel, G. (CRI), 255
Sides, S. (SDSC), 376
Simon, H. (CSC-NASA/Ames), 465
Sloan, J.L. (NCAR), 387
Slowinski, D. (CRI), 291
Snively, A. (SDSC), 100
Spitzmiller, T. (LANL), 349
Spragg, D.A. (EUTEC), 239
Steidel, J. (CRI), 313
Storer, N. (ECMWF), 433
Stouch, T.R. (Bristol-Meyers Squibb), 76
Strand, B. (CRI), 119
Thorp, D. (CRI), 221
Thorp, J. (CRI), 80
Tovo, P.A. (CRI), 369
Vann, L. (CRI), 251
Warraich, K. (Texas A&M), 505
West, S. (CRI), 212
Williams, W. (CRI), 324
Wimberley, F.C. (PITTSC), 39
Witte, K.F. (LANL), 65
Wood, R.R. (LLNL), 209
Young, W. (PITTSC), 39
Zais, J. (CRI), 442
Zumach, B. (PITTSC), 153

GENERAL SESSIONS

|

CHAIRMAN'S REPORT

John F. Carlson

Cray Research, Inc.
Eagan, Minnesota

Good morning. Thank you for inviting me to participate today. I want to take the time available to review our 1993 performance and discuss where we plan to go in our strategic plan covering us from now through 1996.

I am taking some time to review our financial condition because you are all investors in Cray Research. Whether or not you own any of our common stock, you and your colleagues have invested your efforts and energy and professional challenges in Cray Research. I want to use this opportunity to show that your investment is well placed.

Simply stated, we had a very good year in 1993. We delivered on our 1993 plan by increasing revenue, improving margins, reducing costs and delivering our shareholders solid profits for the year of \$2.33 per share versus a loss for 1992.

Our financial performance enables us to continue investing at least 15 percent of our revenues into research and development. No matter what changes the market brings or we bring to the market, that commitment will not change.

We achieved those financial results while also delivering two major hardware platforms and a number of software advances -- including the creation of CraySoft -- our initiative to make Cray software available on non-Cray platforms, right down to PCS.

I am pleased to note, also, that we delivered our new hardware and software products on time, as promised.

Clearly, we have recovered from our 1992 restructuring and are realizing the operating savings we hoped would result from those difficult actions.

For a few specifics, our 1993 revenue grew 12 percent to \$895 million from \$798 million in 1992. This substantial increase resulted principally from strong sales of large C90 systems. We sold 12 C916 systems in 1993 -- double the number sold a year earlier. And having moved on from the C90's introductory stages, we achieved better margins -- which certainly helped our earnings results.

We also improved our balance sheet along the way. We grew our total assets by 15 percent to about \$1.2 billion from 1992, generated about \$170 million in cash, improved stockholder's equity by \$56 million and our book value at year-end 1993 was roughly \$30 a share.

Return to shareholder's equity was eight percent for the year and return on capital employed rose to 11 percent. While these figures are relatively good, they do not yet hit the levels we have targeted for the company. Our targets, long term, are to deliver 18 to 20 percent improvements in stockholder's equity and 15 percent improvement on our return on capital employed.

Also, we ended 1993 with an increase in inventories of about \$52 million. That increase is one figure going the wrong direction and reflects the ramp-up investments we made to launch both the T3D and CS6400 and also the high level of deliveries we anticipate completing this quarter.

In 1993 we signed a total net contract value of orders of \$711 million compared with \$598 million for 1992. This 19 percent increase really reflects the strong demand for C916s and T3Ds. Our order backlog at year-end was \$409 million, just shy of the record \$417 million we reported in 1992. And, yes, the delayed acceptance of a C916 from December 1992 to January, 1993 was included in the 1992 record number. Backing out that system for comparison purposes confirms that our backlog number for 1993 is going in the right direction, upward.

Today, we have about 500 systems installed around the world. Those systems are broadening out geographically and by industry. In 1993, we added new first-time customers in Malaysia and Czechoslovakia and installed our first system in Africa at the South African weather bureau. We also received orders for our first systems to China -- in both the PVP and SMP line. Three Superserver systems are now installed at Chinese universities. The PVP mainline system will be installed at the Chinese Weather service, assuming the export license is granted as we expect.

Right now we have systems installed in 30 countries.

The orders break out by market sector to: 44 percent government customers, 29 percent commercial and 27 percent with universities.

Our commercial and industrial customers continue to use our systems to deliver solutions for their technical and scientific computing needs. I expect this sector to continue to grow and increase its overall percentage of total orders.

Insofar as our product mix is concerned, 1993 was a very good year. As you know we stepped up deliveries of our C916 systems. We also announced the availability of extended C90 family ranging in size from two to 16 processors. This range of availability helps make the C90 product more flexible and available to our customers in convenient configurations. Convenient in size to fit your mission and, hopefully, your budget.

The C90 continues to set the performance pace for our customers. As you may know, more than 20 key codes used by our customers run at sustained speeds exceeding six gigaflops. Five of these 20 codes reached speeds of more than 10 gigaflops each. And as we speak, more performance records are being set. These records are far more than bragging points. They are the ultimate measures of getting real work done on real problems. That will remain a corporate priority for all our systems.

In the MPP arena, we had a very good launch for the T3D system in September. On announcement day we figure we captured third place in the MPP market and expect to be the number one supplier by the end of this year.

By year-end we had 15 orders in hand. I am particularly pleased to note that this week we are able to announce our first commercial and first petroleum customer for the T3D -- Exxon Corporation. It was particularly gratifying to read their news release in which they described the T3D as a critical tool to their growth strategies because it is the first production quality MPP system in the market. As I said with the C90 line, that means Exxon will be doing real work on real problems day in and day out on the T3D.

We're also announcing this week some significant benchmark results for the T3D on the NAS parallel benchmark suite. Now I know that there has been a rush to claim moral or technical victory by any number of MPP companies using NAS results and I promise not to enter into that debate. But I do want to bring two points to your attention.

First point has little to do with MPP benchmarks. Rather, it has to do with the results they are compared

to. The 256-processor system is the first of the MPP crowd to approach the C916's performance on any of the NAS parallel benchmarks beyond the Embarrassingly Parallel test. So the C916, the workhorse of the supercomputing field continues to be the performance leader on these benchmarks. But at 256 processors the T3D is starting to give the C916 a run for its money on highly parallel computing, and since the T3D is showing near-linear scalability we're confident it will widen its performance and scalability leadership as we move to 512-processors and larger configurations. We're also glad it took another system from Cray to approach the C916's performance.

Second point is to note that these results arrived just six months after the T3D was introduced. We all know that we were late to the MPP party. But I think it is even more important to note what is being accomplished following our arrival. Some of our competitors have had two years or more to improve their performance against the C916. I can only assume that they have not announced their complete 256 results to date because they can't efficiently scale up that far.

The strength of the T3D -- and those systems to follow -- reflects the input received from many of you here. This system stands as an example of what can be accomplished when we listen to our customers.

The strength of the T3D will be enhanced by another program that had an excellent year -- the Parallel Applications Technology Partners program, or PATP. When MPP performance shows steady, consistent improvement in a wide range of applications, it will be in part due to the efforts underway between Cray and its PATP partners, PSC, EPFL, JPL and Los Alamos and Livermore labs. Great things are coming from these collaborations.

The third sibling of the Cray architecture family is our Superservers operation, based in Beaverton, Oregon. They also delivered a beautiful, bouncing and robust baby in 1993 -- the CS6400. The CS6400 is a binary compatible extension of the Sun product line. It serves as a high performance server for networks of workstations and smaller servers running the huge range of Sun applications. Any program that runs on a Sun system will run on the CS6400 without modification and the current product is Solaris 2.3 compatible.

We expect this newest arrival to create opportunities for us in markets where we have not traditionally had a presence. They include mainstream commercial applications in financial services, banking, telecommunications, government data processing applications, et cetera. It is proving to be an excellent

alternative to a mainframe in the so-called "rightsizing" market. It is also important to note that the CS6400 is an excellent alternative to small-configuration MPP systems for both the commercial and scientific and technical markets.

Introduction of the Superserver subsidiary brings me to an important point and sets the stage to discuss our strategic direction. First the important point: Cray Research is focused on its mission of providing the highest performing systems to its customers. We are not architecture constrained. In fact, we are the only firm that can provide top-performing systems in all three major architectures -- Parallel Vector; Massively Parallel and Symmetric Multiprocessing.

You and your colleagues want solutions. We want to help you get them. It matters little to us whether your ideal solution comes from one specific architecture or another. What matters is that Cray Research continue as a technical partner in finding the solutions.

We don't have to -- nor do we want to -- make two sales every time we talk with you. We don't have to sell you on one architecture or another and then, next sell you on our product over a competitor's. We remain focused on helping deliver the highest performance possible in each architecture. We are certain that if we continue to stick to our knitting with that focus, we will be successful as you become successful with our systems.

Now I want to tie this together to our strategic plan. Our plan is simple. We have to grow in order to accomplish the things that are important to you and to us.

If we want to remain at the head of the performance class, we must continue to invest huge sums each year in R&D. Those sums are only available if we continue to grow revenues, earnings, margins and backlogs. We wouldn't have to change anything in our strategy if the technical and scientific market was growing, say 30 to 40 percent as it did in the 1970s. But the reality is that the market is flat and, depending on how you measure it, has been for four consecutive years.

At the same time, we don't want to change who we are. We are a technology driven company. We will always be a technology driven company committed to achieving the highest performance possible at the best price.

So, while our focus can't change, our tactics can. As we maintain our marketshare leadership at a relatively flat level in the scientific market -- for now, at least -- we see that a growing need for efficient, price/effective

systems to solve big problems in other markets. Commercial entities of all sizes and shapes are recognizing that they have done a great job in accumulating huge data bases, but are limited in how well they can manipulate or mine these data bases to their advantage.

Some businesses have said they forgot the second part of the equation -- "what are we going to do with all this information and how, pray tell, are we going to use it to grow our businesses?"

Here's where we believe Cray Research can help.

We believe that our technologies can help. Across all three of the important high performance architectures our products are distinguished by their technical balance. Balance that unlocks the problem-solving potential of increasingly fast microprocessors -- regardless of the architecture in which they are embedded.

We combine fast processors with high-performance memory and I/O subsystems and robust, proven software to deliver unsurpassed balance right to your desktop. That won't change. But we may be able to add new applications in other parts of a commercial enterprise.

We plan to use this unique strength to grow our volume and market reach. Instead of finding a Cray system at the R&D center only, I can picture one being used by the marketing or finance functions as well.

Just this last year we've seen new customers emerge from non-traditional fields. Like Wall Street, I expect to see more and more utilization at new locations.

In doing so, much like the early successes of the superserver product, we'll probably draw some attention. Some of our nervous competitors will whisper that we've "lost focus" as we compete for and win commercial business. Quite the contrary, I believe our move to add customers in non-traditional supercomputing areas is an example of our unique focus. We see this approach as the best way to ensure that we don't blink when we face the future. We remain committed to the top of the performance pyramid. We remain focused on the need to invest at least 15 percent of our revenues in to R&D so we can move our performance up the scale to match your demands. And we remain focused on the importance of long-term, cooperative relationships with our customers. And that may be the best single investment we've ever made.

Thank you again for inviting me today.

Cray Research Corporate Report

Robert H. Ewald

Cray Research, Inc.
900 Lowater Road
Chippewa Falls, WI 54729

ABSTRACT

This paper provides an overview of Cray Research's business in 1993, an update on progress in 1994, and a vision of Cray in the future.

1 1993 Results

1993 proved to be a very good year for Cray Research, Inc. We achieved our 1993 plan by delivering a broad mix of high performance systems and software, and achieved a return to solid profitability. 1993 was also a year of substantial change for Cray Research, but before describing the changes, I will review some terminology. The "Supercomputer Operations" business unit is focused on high performance, cost effective computational tools for solving the world's most challenging scientific and industrial problems and produces our parallel vector and highly-parallel products.

In this technical computing world we have defined three segments that encompass different customer characteristics. The "Power" segment can be recognized as those with "grand" challenge problems. These customers are typically from the government and university sector and require the highest performance solutions of scientific and engineering problems. Most of the applications are mathematically-oriented and are created by the customer. This segment typically invests more than \$10 million in computer products. The "Endurance" segment can be characterized as customers with production problems to be solved. There are a combination of industrial, government, and academic customers each solving high performance production engineering, scientific, and data-oriented problems. In this segment,

applications are frequently from third party providers. Price/Performance is of major concern for these products, ranging from about \$3 million to \$10 million. The "Agile" market segment is comprised of industrial, government, and academic customers with engineering, scientific, and data-oriented problems. The applications are generally third party and their price/performance is of paramount importance for the production capability. The hardware investment here is less than \$3 million.

The "Superserver" business unit addresses applications that require rapid manipulation and analysis of large amounts of data in a networked production environment, and produces products based on Sun Microsystem's SPARC architecture. Many of the applications are at non-traditional Cray customers in the business or commercial segment.

As shown in Figure 1, we are changing our organization to reflect these concepts. Les Davis and I share the office of Chief Operating Officer and we have two major business units: Supercomputer Operations and Superserver Operations. Changes within the organization since the last CUG meeting include:

- Gary Ball, Vice President of Government Marketing and General Manager of "Power" Systems - in addition to his previous job, Gary is leading our efforts to continue to lead in the high-end of supercomputing.

- Rene Copeland, General Manager of "Endurance" Systems - Rene is leading our efforts to ensure that we provide products and services for the industrial, production-oriented customers.

- Dan Hogberg, General Manager of "Agile" Systems - Dan is leading our work with deskside and departmental systems.

- Dave Thompson, Vice President of Hardware Research & Development - Dave has moved from the Software Division to lead our hardware development and engineering efforts. Dave is teamed with Tony Vacca who also leads the Triton and other projects.

- Don Whiting, Sr. Vice President, Operations - we have combined all product operational divisions (Integrated Circuits, Printed Circuits, Manufacturing, and Systems Test & Checkout) under Don to improve our operational processes and efficiency.

- Larry Betterley, Vice President of Finance & Administration - Larry leads the operational Finance and Administration groups within the business units.

The Superservers business unit remains the same, and in addition, Paul Iwanchuk and Penny Quist are acting leaders of new initiatives in Government Systems & Technology and Data Intensive computing.

Figure 2 shows the progression of orders and the results for 1993, indicating our best year ever for orders for our larger products and a good base of orders for Agile and Superserver products.

During 1993 we installed half of the systems shipped in North and South America and one-third in Europe. As shown in Figure 3, that brings the total installed base to 54 percent North and South America, 30 percent Europe, and 11 percent Japan. We expect that more of our future business will come from non-U.S. based customers, so the U.S. percentage will continue to decline.

The Agile installs in 1993 were distributed 45 percent to the Americas, 33 percent to Europe, and 14 percent to Japan. This brings the total Agile installed base to 40 percent Americas, 33 percent Europe, and 19 percent Japan, as shown in Figure 4. This again reflects the increasing "internationalization" of our business.

Figure 5 shows the installed base at year-end 1993 by industry. The most significant changes in the distribution that occurred during the year were increases in the university, aerospace, and environmental segments, and a decrease in the government research lab business reflecting changing government spending.

Figure 6 shows the distribution of the 500 systems installed at customer sites. The Y-MP and EL products are dominant with C-90s and T3Ds just beginning to ramp up.

There were 38 new customers in the Supercomputer business unit and 12 in the Superserver business unit. We welcome all of you to the growing Cray family and hope you will share with us your ideas for the future of computing.

We ended 1993 with the strongest offering of products we have ever fielded. We offer three architectures, each with superior price/performance: parallel vector, massively parallel, and symmetric multiprocessing. The parallel vector product line was expanded by extending the Cray C-90 to configurations ranging from 2 to 16 processors. We began shipping the C-92, C-94, and C-98 at mid-year. All the C-90 series systems feature the newest, fastest memory technology available - four megabit SRAM (static random access memory). We also introduced the big memory machines - D-90 with up to 4 billion words of memory.

Two new departmental supercomputers were introduced - the EL92 and EL98. The EL92 is the company's smallest, lowest-priced system to date with a U.S. starting price of \$125,000 and a footprint of about four square feet. The Cray EL92 is designed to extend Cray Research compatibility to office environments at attractive prices. The EL98 is available with two, four, six, or eight processors, offering up to 512 million words of central memory and providing a peak performance of one gigaflops (billion floating point operations per second).

During 1993 we expanded our product offerings for the scientific and technical market by introducing our first massively parallel supercomputer - the Cray T3D. We now have five installed and are expecting to install over thirty this year. Reaching that target will make us the leading MPP vendor in 1994.

In addition to these products, the CS6400, a SPARC-based symmetric multiprocessing system, was introduced with configurations ranging from 4 to 64 processors. This platform is for commercial data management applications as well as scientific and technical users. We expect to install over 50 of these new machines in 1994.

New announcements in software include the release of Network Queing Environment (NQE) and our Fortran compiler (CF90). Our new business entity "CraySoft" was successfully initialized to provide Cray software products and technologies to the non-Cray platforms.

2 1994 Plans

As we start 1994, we have set some aggressive goals for ourselves that include:

- Continuing to have a strong financial foundation with some growth over 1993.
- Continue to invest about 15 percent of revenue in R&D to develop hardware and software products and services.
- Continue with R&D work on all three product families: parallel vector, MPP, symmetric multiprocessing.
- Continue to unify our UNICOS software base across our PVP and MPP platforms.
- Continue to make more software available on workstations and servers.
- Improve the application availability, performance, and price/performance on our systems.
- Continue our reliability improvements.
- Better understand the "cost of ownership" of our products and make improvements.

3 Strategic Summary

Cray Research is one of the world's leading technology companies. At the heart of our company are four core technology competencies (Figure 7) which set us apart from others, and upon which we will build our future:

- 1) Hardware and manufacturing technology to create fast, balanced computers.
- 2) Applications and software that support production computing.
- 3) The ability to solve problems in parallel.
- 4) Distributed computing skills that allow us to deliver results directly to the user via a network.

We will focus on these four technology strengths to create a set of products and services that help solve our customers' most challenging problems. We will leverage our technology to provide hardware products that range from desktop systems to the world's most powerful computers as shown in Figure 8. Our software products and services will enable us to deliver production computing results to the user anytime, anywhere. To enable this, some of our software and services will be applied to other vendors systems - primarily workstations. Our CraySoft initiative addresses this market.

We will also expand our service offerings to better help our customers solve their problems. We will develop new distribution mechanisms in the form of service products that enable us to package our hardware, software, and application products as total solutions to our

customer's problems. We will develop plans to enable our customers to buy total computational services on demand, in ways that are consistent with their needs and their internal budget constraints (shown as Anytime, Anywhere Solutions in Figure 9).

Spanning all of our systems is the key element that allows the customer to buy and use our systems - the application programs that solve the customer's problem. We will port, adapt, optimize, create, or attract the applications that the customers require. We will lead the industry in converting applications to run in parallel and deliver the performance inherent in parallel systems.

We also recognize that our core competencies, products, and services can be applied to problems which are more "commercial" in nature. We will focus our supercomputing business on technically-based problems, and will focus our Superserver business and some new efforts on helping solve open system, performance-oriented commercial problems. Typically, these commercial problems will require the rapid manipulation and analysis of large amounts of data in a networked, production environment as businesses seek to better understand their data and business processes and make better decisions more rapidly. Within both technical and commercial segments, we will also become more customer driven.

We will also open a pipeline between our technology and selected government customers with our government systems and technology effort. As the government customers require, we may:

- 1) Tailor existing products to better suit application needs.
- 2) Perform custom R&D work.
- 3) License selected core technologies for new applications.

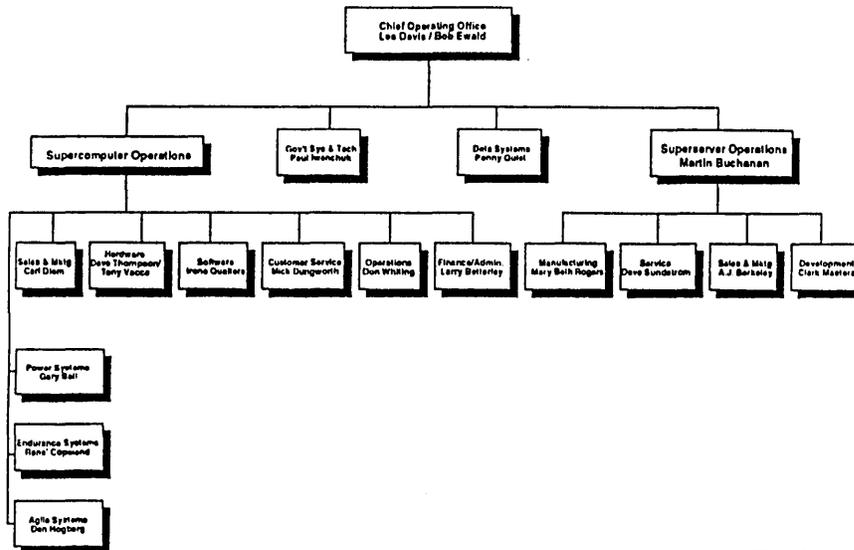
In the commercial markets, we will also tailor our products to meet a new set of problems - those that are more "data intensive." We will build our Superserver business on top of Sun's business. We will understand the market requirements for extracting and creating new information from databases for commercial, industrial, and scientific applications. We will then apply our technology and products to that marketplace, and consider partnerships to strengthen our position, particularly at the high-end of the commercial business.

Putting all of these pieces together yields a customer driven, technology company as depicted in Figure 10.

4 References

[CRI93] *Cray Research, Inc. Annual Report, February, 1994. Cray Research, Inc.*

CRI OPERATIONAL ORGANIZATION

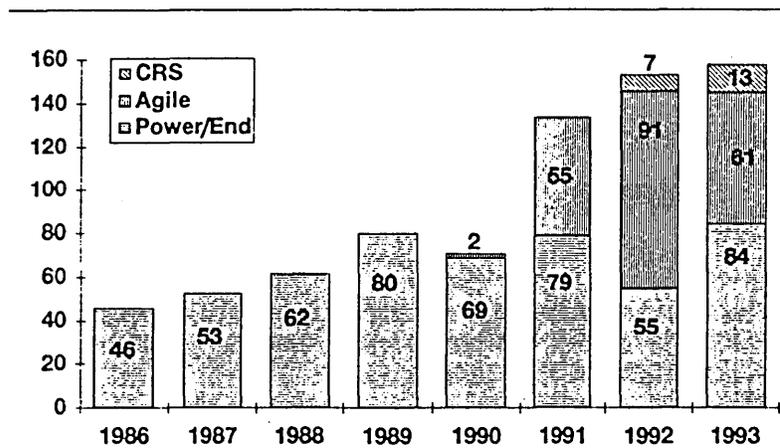


February 1994



Figure 1

Orders per Year



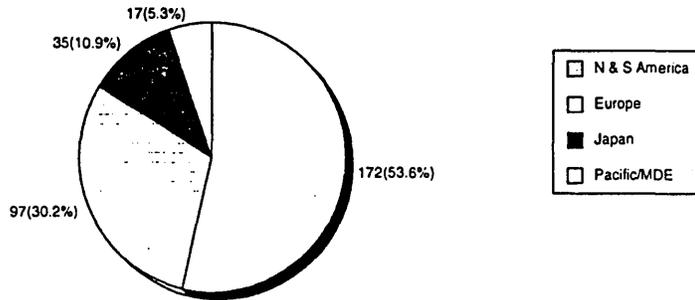
Sales and Marketing
3/10/94, slide 8

CRI 4Q93 Overview



Figure 2

**Cray Installed Base
By Geography
December 31, 1993**

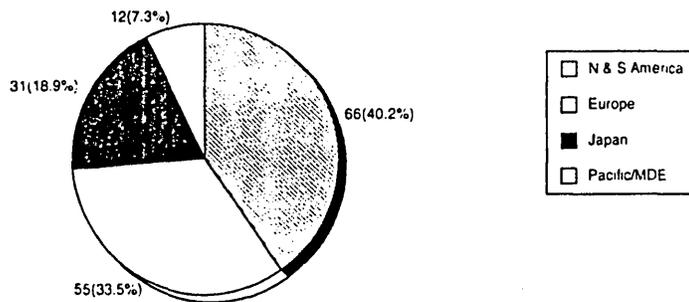


CRAY *delivering the performance*

CUG SanDiego / 031494 / RHE. smk

Figure 3

**Cray Agile Installed Base
By Geography
December 31, 1993**

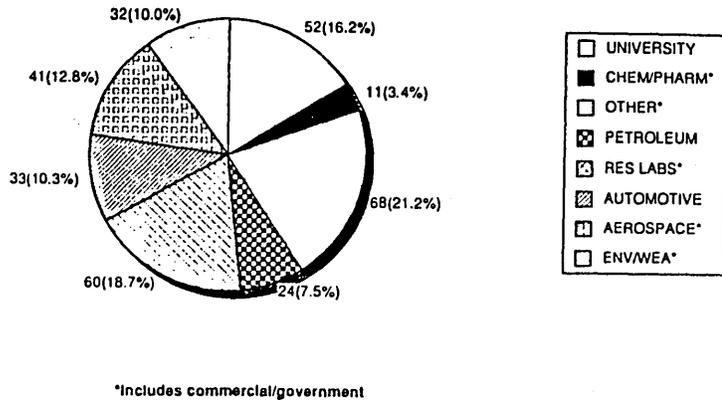


CRAY *delivering the performance*

CUG SanDiego / 031494 / RHE. smk

Figure 4

Cray Customer Installed Base By Industry As of December 31, 1993

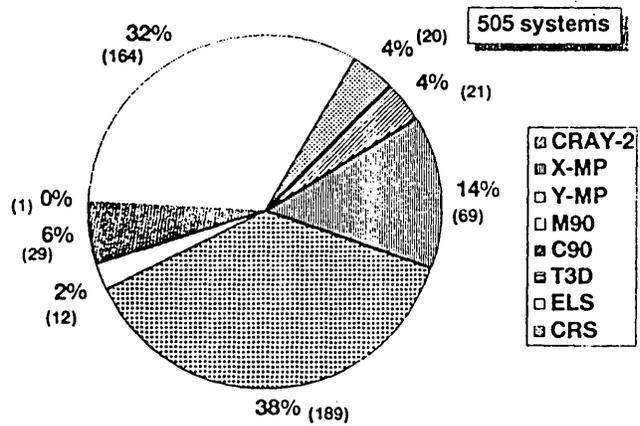


CRAY *delivering the performance*

CUG San Diego / 031484 / H1E.emb

Figure 5

Installed Customer Systems

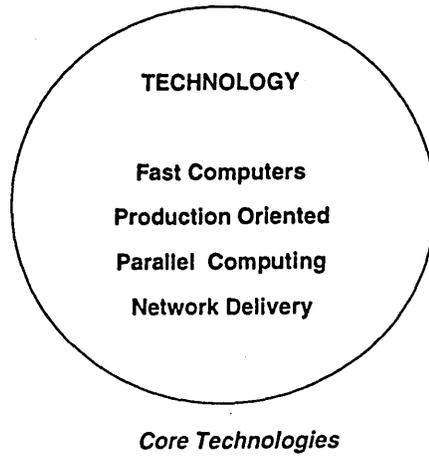


Sales and Marketing
3/10/94, slide 12

CRI 4Q93 Overview

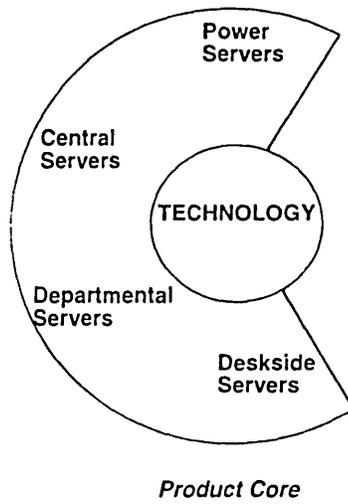
CRAY
RESEARCH INC.

Figure 6



CRAY ...delivering the performance

Figure 7



CRAY ...delivering the performance

Figure 8

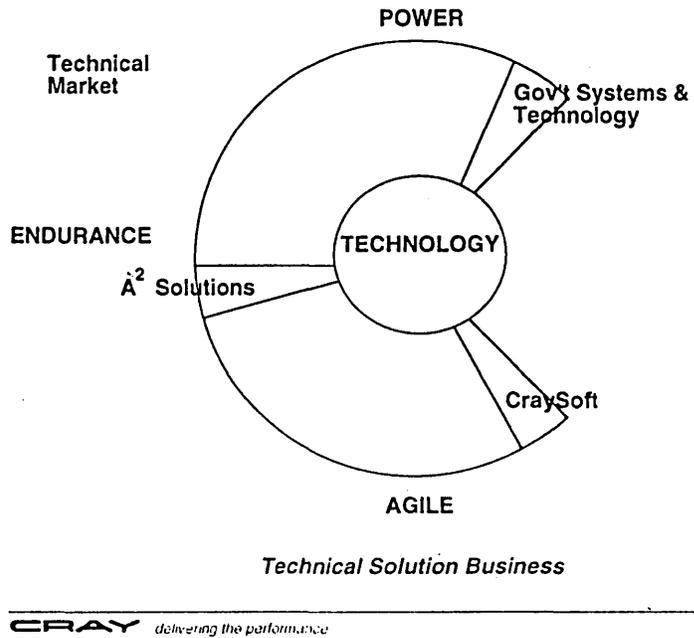


Figure 9

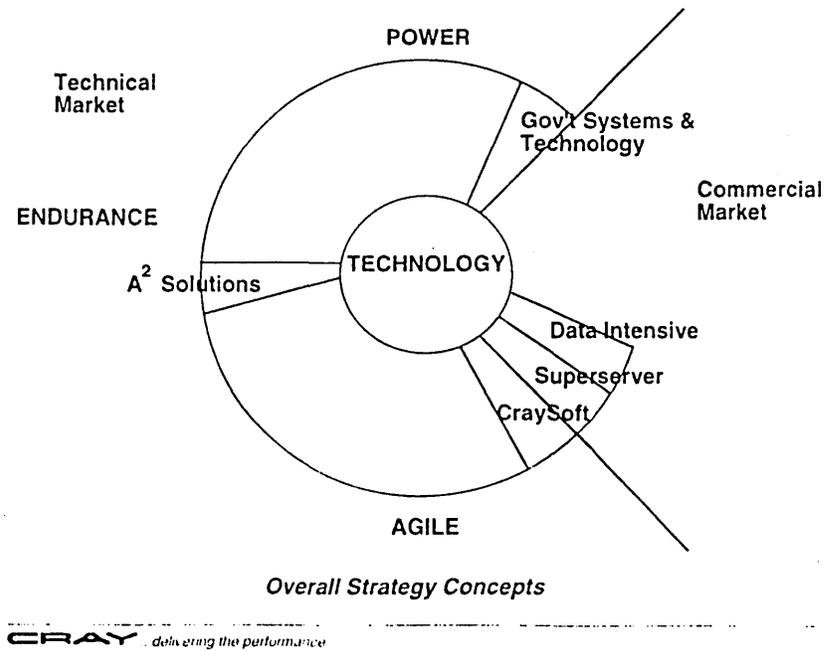


Figure 10

CRI Software Report

Irene M. Qualters
Senior Vice President

Cray Research, Inc.
655F Lone Oak Drive
Eagan, Minnesota 55121

This paper describes the Cray Research Software Division plans and strategies for 1994 and beyond.

1 Introduction

Since the last CUG, CRI improved the software reliability while increasing the installed base by 30%, improving performance, and increasing functionality. We also changed our organization to create smaller groups, allowing greater focus and agility. This paper covers these topics as well as recent and planned product releases for operating systems, storage systems, connectivity, and programming environments. The progress by CraySoft is presented and product retirements are noted. At the end of this paper, the status of the recent CRAY T3D installations are highlighted.

2 Software Division Reorganization

Our two largest groups, UNICOS and Networking, were reorganized into three new groups: Operating Systems, Networks, and Storage Systems. Dave Thompson (the former UNICOS group leader) was promoted to Vice President, Hardware R&D in Chippewa Falls. Paul Rutherford is the leader for the new Storage Systems group. Wayne Roiger is the group leader for Networking. Kevin Matthews is acting as the group leader for Operating Systems, until a permanent leader is chosen. (See figure 1 for the updated organization chart.)

3 Reliability

From August 1993 through February 1994 the Software Problem Report (SPR) backlog, incoming rate and fix rates improved (see figure 2). The six-month rolling software and system MTTI also improved over this period (see figure 3). The box plot metric (figure 4) confirmed this improvement, but shows stability is still an issue at some sites. The dashed lines at the bottom of the chart indicate sites with stability substantially below the mean. The number of sites with low stability has decreased considerably over this period, nevertheless in

1994 we will emphasize reliability improvements at sites with low MTTIs.

4 Operating Systems

4.1 UNICOS 8.0 Features (3/10/94)

UNICOS 8.0 was released on March 10, 1994. Its themes include robustness and resiliency, performance, security, standards, and new hardware. On March 9, 1994, UNICOS 8.0 received an official DoD Orange/Red Book B1 MDIA rating. This culminates four years of effort, making CRI the sole supercomputer vendor evaluated as a network node. UNICOS 8.0 improves our compliance with standards with direct FDDI connectivity and improved POSIX 1003.2 compatibility.

The new platforms supported include M90, C90, EL98, EL92 (UNICOS support for T3D), J90, DA-60/62/301, DD-301 and ND-12/14 disks, DD-5, RAID 10 disks for EL, EMASS ER90 D2 and Datatower, 3490E tapes, SSD-E 128i and FCA-1 channel adapter. UNICOS 8.0 will also be supported on CRAY X-MP systems (with EMA support) and CRAY-2 systems.

4.2 Field Tests - UNICOS 8.0

Extensive field tests were completed. We ran six field tests, installed four prereleases, and have nine additional installs in progress. We tested the widest variance of platforms to date: CRAY X-MP, CRAY Y-MP, CRAY C90, and CRAY Y-MP/EL platforms. We included a production tape site—with excellent results. Stability improved, compared to 7.0; the SPR count was reduced and the severity of SPRs declined.

UNICOS 8.0 multithreading performance measurements at NASA/Ames showed a 60% reduction in system overhead, compared with UNICOS 7.0.

4.3 UNICOS 9.0 (3Q95)

UNICOS 9.0 will support Triton and IOS-F hardware. It will support additional standards: X/Open XPG4 brand-

ing, ATM, and ONC+ (NFS V3). It will improve the heterogeneous computing capabilities with the Shared File System (SFS) and UNICOS MAX.

A major theme for UNICOS 9.0 is Reliability, Availability, and Serviceability (RAS). Specific RAS features are UNICOS under UNICOS and checkpointing tapes.

UNICOS 9.0 potentially will support additional peripherals: Escon/FCA-2, Redwood Autoloader, and IBM NTP.

4.4 UNICOS MAX Releases

UNICOS MAX 1.1 will be released in 2Q94. It includes improvements in resiliency and interactive scheduling. It supports the CRAFT programming model.

UNICOS MAX 1.2 is planned to be released in 4Q94. It will support phase-II I/O and job rolling (coarse-grain timesharing). Phase-II I/O adds HISP channels that act as direct data channels between IOCs and the MPP. The control remains on the host and the IOCs maintain physical HISP/LOSP connections to the host. This generally increases the number of I/O gateways that can function in parallel.

UNICOS MAX 1.3 is planned to be released in 2Q95, with support for phase-III I/O. Phase-III I/O allows IOCs to attach to the MPP, without physical connections to the host. Control remains on the host; the host controls the remote IOCs through virtual channels that pass through the MPP. This allows the number of IOCs to exceed the number that will physically connect to the host.

5 Storage Systems

Several new file systems will be available in 1994 and 1995. All of these features are unbundled and must be ordered separately. The DCE/DFS (Distributed File System) will be released for UNICOS 8.0 in 3Q94. ONC+/NFS3 will be available with UNICOS 9.0. The Shared File System (SFS) will first be available in UNICOS 8.2 and then in UNICOS 9.0. SFS will support multiple UNICOS hosts sharing ND disks.

Planned hierarchical storage management improvements include DMF 2.2, UniTree, and FileServ support. With the introduction of DMF 2.3, we plan to offer client/server for SFS. This means a DMF host could act as a DMF server for systems that share the SFS disks.

On Y-MP ELs we will offer the following peripheral upgrades: SCSI disks (DD-5s), IPI disks (DD-5i),

Metrum tape library, and SCSI tape and disk controller (SI-2).

6 Connectivity

6.1 ATM

ATM pilot tests will run from 2Q94 through 4Q94. The ATM will be connected to CRAY Y-MP and CRAY C90 hosts through Bussed Based Gateways (BBGs) and with native connections on CRAY Y-MP EL systems. The BBGs will support OC3 and OC12 (622 Mb/s). The native CRAY Y-MP EL connections will support OC3 (155 Mb/s).

Software support for ATM will be in UNICOS 9.0. Long term plans are to prototype OC48 (2.6 Gb/s) in 1995 on IOS-F and to prototype OC192 (8 Gb/s) in this decade on IOS-F.

6.2 NQX

NQX helps NQS on UNICOS to interoperate with NQE. When one adds NQX to NQS and FTA, the result is the NQE capabilities on a UNICOS host. NQE will replace RQS and Stations.

7 Cray Application Programming Environment

7.1 Cray Programming Environment 1.0

The CF90 Programming Environment 1.0 was released in December 1993. Cray Research was the first vendor to release full native Fortran 90. Twenty-six CF90 licenses have been purchased.

On average, CF90 compiled code runs within 10% of the speed of CF77 code. Some codes run faster, some slower.

A SPARC version of CF90 will be available in 3Q94.

The CF77 6.1 Programming Environment for the CRAY T3D will be released in 2Q94. Its main feature is support for the CRAFT programming environment.

The C++ Programming Environment 1.0 for the CRAY T3D is will also be released in 2Q94. With this release, the MathPack.h++ Tools.h++ class libraries will be available. These libraries are unbundled and must be purchased separately.

7.2 Programming Environment 2.0

The CF90 Programming Environment 2.0 for all platforms (PVP, MPP, and SPARC) will be released in 2Q95. A goal we expect to meet for CF90 2.0 is to exceed the performance of CF77.

The C++/C Programming Environment 2.0 for PVP and MPP platforms will be released 2Q95. Note: the

C++/C Programming Environment 2.0 is intended to replace the Cray Standard C Programming Environment 1.0 and the C++ Programming Environment 1.0. C++/C is both an ANSI Standard C compiler and a C++ compiler.

7.3 *Distributed Programming Environments*

The Distributed Programming Environment (DPE) 1.0 is scheduled for release in 3Q94 with support for the SunOS and Solaris platforms. It includes the front-end for CF90 that allows test compiles on the workstation and `dpe_f90`, which performs remote compiles on the host. It provides ATExpert and Xbrowse locally on the workstation linked through ToolTalk to the host.

DPE 2.0 is planned to be released in 2Q95 for the Solaris platform. It will add a full CF90 2.0 cross compiler and native Cray TotalView support through ToolTalk. With DPE 2.0, the compiler will produce host object code on the workstation and the `dpe_f90` command will upload the data to the host and link it on the host with transparent remote commands.

8 CraySoft™

CraySoft released its first product in December 1993: NQE for Solaris. This included NQS, a load balancer, queuing clients, and the File Transfer Agent (FTA.)

In 3Q94, CraySoft plans to release NQE 1.1 for multiple vendors including workstations and servers from IBM, SGI, HP, Sun and Cray Research.

Also in 3Q94, CraySoft plans to release DPE 1.0 for Solaris. (See the Distributed Programming Environments section above.)

In addition, in 3Q94, CraySoft plans to release CF90 1.0 for Solaris.

9 Product Retirement

Ada and Pascal will be placed in maintenance mode one year from now. The last feature development for these products is complete. They will be supported on CRAY Y-MP, CRAY M90, and CRAY C90 platforms through UNICOS 9.0. They will not be offered on new hardware, such as the CRAY J90 series.

CrayDoc will replace DocView; DocView will be placed in maintenance mode one year from now. The final OS platform on which DocView will be supported is UNICOS 9.0. CrayDoc will be available with UNICOS 8.0.3

10 CRAY T3D Product Status

Six sites have installed CRAY T3D systems, and we have received fifteen additional orders. The user base is diverse, including industry, government, and university customers. The hardware in the field has been highly reliable. The software is stable and maturing. The I/O is performing as expected: over 100 MB/s on one IOG to SSD and over 350 MB/s on four IOGs to SSD.

The CRAY T3D system is the only MPP that has run all eight of the NAS Parallel Benchmarks on 256 processors. All other vendors have been unable to scale all eight benchmarks to this level of parallelism. The CRAY C916 system runs all but one of the benchmarks faster than a 256 processor CRAY T3D system, but the CRAY T3D system is approaching the CRAY C916 performance on many of these benchmarks and is expected to exceed the CRAY C916 performance when scaled to 512 processors. The scaling has been excellent; the performance on 256 processors was almost double that of 128 processors (see figure 5).

The CRAY T3D system is proving to be an excellent graphics rendering engine. Microprocessors excel at this task, compared with vector processors. The 256 processor CRAY T3D system runs a ray-tracing application 7.8 times faster than a CRAY C916 system.

The heterogeneous nature of the CRAY T3D system is proving to be an advantage for some codes. The SUPERMOLECULE code is a good example. It contains a mixture of serial and parallel code. Running the serial code on a fast CRAY Y-MP processor substantially improves the scalability. If all the code, including the serial portion, is run on the MPP, the code runs only 1.3 times faster on 256 processors than on 64 processors. When the serial portion of the code is run on a CRAY Y-MP CPU, the program runs 3.3 times faster on 256 processors than on 64 processors (see figure 6).

11 Summary

Cray Research remains committed to an Open Supercomputing strategy.

We build our systems using standards, such as POSIX, System V, Solaris, X/Open, ANSI, and COSE. We concentrate on performance such as scalable parallel processing and automatic parallelizing compilers. We excel in resource management such as comprehensive accounting and NQS production batch facilities. We

have the most comprehensive UNIX security on a major computing platform, including "Trusted UNICOS", with an official Orange/Red book B1 MDIA rating. These security features are very useful for commercial sites.

We are constantly improving our data accessibility with features such as ONC+/NFS3, DCE/DFS, hierarchical storage management (DMF, UniTree, FileServ), FDDI, HiPPI, and ATM. Finally, we will continue improvements in providing a cohesive environment, including COSE (CDE), the CraySoft Network Queueing Environment, Fortran 90, and technology agreements with workstation vendors such as Sun Microsystems, Inc.

We will continue our investments in Open Supercomputing to constantly improve methods for bringing supercomputing to the desktop.

12 Appendix: Explanation of Box Plots (Figure 4)

12.1 Comparing Estimated Software and System Time Between Failures

Figure 4 compares the distributions of customer Software and System MTTI estimates. The solid line represents the median customer Software MTTI, and the dotted line represents the median customer System MTTI. The boxplots allow us to see how these distributions change over time. The graph compares the MTTI distributions for Software and Systems for of all customer machines.

12.2 Boxplots

Boxplots are often used to give an idea of how a population or sample is distributed, where it is centered, how spread out it is, and whether there are outliers. The box itself contains the middle 50% of the data. The line in the middle represents the median. The whiskers extend outward to cover all non-outlier data. Outliers are plotted as small dashes. An outlier is a point that "lies out" away from the main body of the group.

The median is a "measure of location." That means it is a nice metric for telling "where we are." (Where we are centered.) The boxes help show us how spread out we are.

While boxplots do not display everything there is to know about a data set, they are quite useful in allowing us to compare one data set to another. By lining boxplots up side by side we can often tell whether two or more data sets are located around the same central

value, or whether they have the same amount of spread.

We use log paper in the Y axis, since otherwise we would not be able to observe what is happening in the lower quartile, and this is probably where we would like to focus our attention.

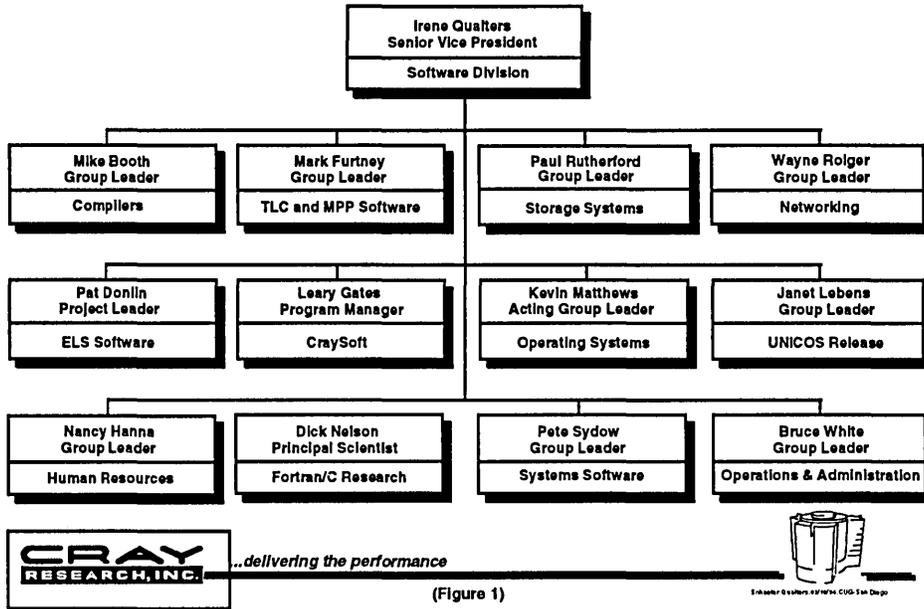
12.3 Where does the Data Come From?

For each site a list was obtained from the IR database of times when Software or System crashes occurred. From this information we were able to estimate the MTTI for each site at any moment in time. In each graph the most recent month is not included. (We expect data that will be reported late to bring the lower part of the last month's box down.) For more information please feel free to contact David Adams by email (dadams@cray.com) or phone (612) 683-5332.

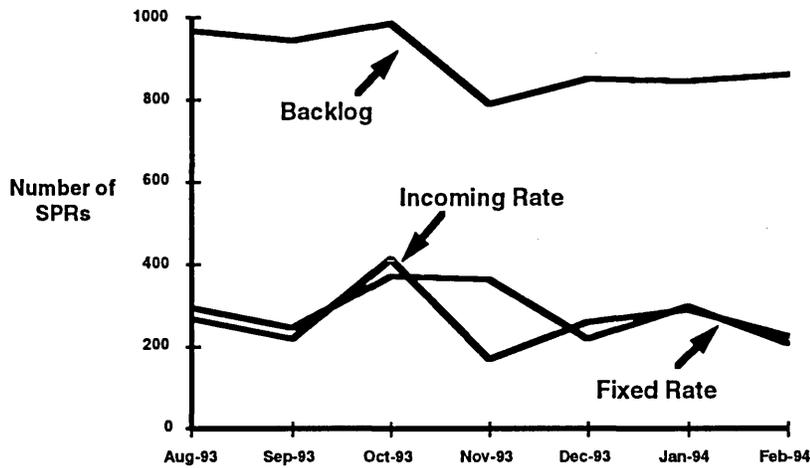
12.4 Analysis

All of the medians seem to be fairly stable. (They are not moving up or down significantly.) The shaded notch in the middle of each box is a 95% confidence interval on the median for the box. If any two boxes have shaded notches that do not overlap horizontally, then those boxes have significantly different medians in the statistical sense.

Supercomputer Operations Software Division

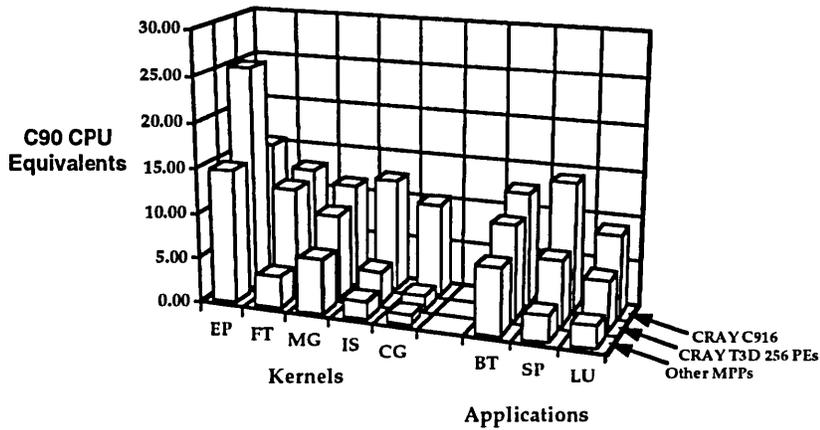


SPR Status



NAS Parallel Benchmarks

Highest Performance Reported on Any Size Configuration



Note: No Other Major MPP Vendor Scaled All 8 Codes to 256 PEs



...delivering the performance

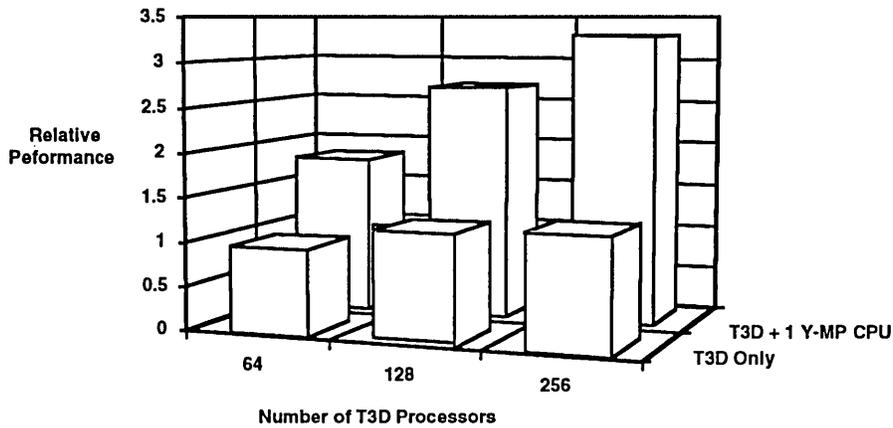
(Figure 5)



16mb/1.5mb/127mb/100.5mb/100.5mb

SUPERMOLECULE

Homogeneous versus Heterogeneous Performance



...delivering the performance

(Figure 6)

Molecule = 18-Crown-6 3-21G
Updated 2/3/94



16mb/1.5mb/127mb/100.5mb/100.5mb

Unparalleled Horizons: Computing in Heterogeneous Environments

Reagan W. Moore

San Diego Supercomputer Center
San Diego, California

Abstract

The effective use of scalable parallel processors requires the development of heterogeneous hardware and software systems. In this article, the Intel Paragon is used to illustrate how heterogeneous systems can support interactive, batch, and dedicated usage of a parallel supercomputer. Heterogeneity can also be exploited to optimize execution of task decomposable applications. Conditions for super-linear speedup of applications are derived that can be achieved over both loosely and tightly coupled architectures.

Introduction

Scalable heterogeneous parallel architectures are a strong contender for future Teraflop supercomputers. From the systems perspective, heterogeneous architectures are needed to support the wide range of user requirements for interactive execution of programming tools, for production execution of parallel codes, and for support for fast disk I/O. From the application perspective, heterogeneity can also lead to more efficient execution of programs. On heterogeneous systems, applications can be decomposed into tasks that are executed on the appropriate hardware and software systems. For

a certain class of applications, superlinear speedups can be achieved. The execution rate of the application can be increased by a factor greater than the number of decomposed tasks.

To demonstrate the viability of heterogeneous parallel architectures, a comparison will be presented between the job mixes supported on the Cray C90 and the Intel Paragon XP/S-30 at the San Diego Supercomputer Center. The observed C90 workload cannot be efficiently executed on a homogeneous massively parallel computer. The heterogeneous hardware and software systems on the Paragon, however, do provide support for a job mix similar to that of the C90. The operating system software that controls the heterogeneous resources on the Paragon will be described. Conditions for achieving superlinear speedup will be derived that are valid for both tightly coupled architectures such as the C90/T3D, and for loosely coupled architectures such as a Paragon and C90 linked by a high-speed network.

Heterogeneity in Application Resource Requirements

The Cray C90 supercomputer supports a job mix with widely varying application resource requirements. In addition, the

resource demands can vary significantly between the CPU time, memory size, and disk space required by a job. Jobs that need a large fraction, from one-quarter to one-half, of any of these resources are "boulders" that can dramatically affect the performance of the C90. Examples of these types of resource demands are support for fast turn-around for interactive job development, execution of large memory production jobs, and execution of jobs requiring large amounts of disk space. "Boulders" constitute the dominant need for support for heterogeneity on the C90.

At SDSC, "boulders" are controlled through a dynamic job mix

scheduler [1-3]. The scheduler automatically packs jobs in the C90 memory while satisfying scheduling policy constraints. The turn-around time of particular classes of jobs is enhanced while maintaining high system utilization. The limiting resource that is controlled on the C90 is memory. Enough jobs are kept in memory to ensure that no idle time occurs because of I/O wait time.

Job mix statistics for the C90 and the Paragon for the month of January, 1994 are given in Table 1. The C90 at SDSC has 8 CPUs, 128 MWords of memory, and 189 GWords of disk space.

Table 1
Interactive and batch workload characteristics for the C90 and the Paragon for the month of January, 1994

	C90	Paragon
Number of Interactive jobs	101,561	4,731
Number of Batch jobs	8,279	1,111
CPU time interactive (processor-hrs)	410	13,409
CPU time batch (processor-hrs)	3,595	145,860
Average batch CPU time (processor-hrs)	0.43	131

There are several noteworthy items about the job mix on the C90. Users rely on the C90 to support interactive job development. These jobs constitute over 90% of the jobs, but use less than 10% of the CPU time. Thus the dominant use of the C90 by number of jobs is for fast interactive support of researcher development efforts. The dominant use by execution time is for deferred execution of batch jobs. Even for jobs submitted to the batch queues, typically half of the runs are for short execution times of less than five minutes. Excluding these short execution time batch jobs, the long-running batch jobs execute for about one hour.

Typical types of support needed for job development on the C90 are shown in Table 2.

On the C90, these development support functions are run in competition with the batch production jobs. Although they comprise a small fraction of the total CPU time, their need for fast turn-around times does impose a heavy load on the operating system. On a heterogeneous architecture, these functions could be executed on a separate set of resources. Another characteristic of the development tasks is their need for a comprehensive set of UNIX system calls. Excluding file I/O manipulation, most production batch

On the Paragon, traditional UNIX interactive development tasks are executed on the service nodes, while parallel compute jobs are executed on the 400 compute nodes. Note that in Table 2, the two dominant development efforts in terms of CPU time are archiving of data and compilation of codes. At SDSC, these functions are migrated off of the Service nodes onto compute nodes for execution in parallel. Parallel MAKE jobs are typically executed on 4 compute nodes on the Paragon. The compute nodes to the right of the solid vertical line in Figure 1 have 32 MBytes of memory, while the remaining compute nodes have 16 MBytes of memory. The nodes to the left of the dashed vertical line are reserved for interactive execution of parallel compute jobs; the other nodes are controlled by the batch system.

The statistics presented in Table 1 indicate that this heterogeneous system supports a workload whose characteristics are similar to that of the C90. The number of interactive jobs on the Paragon only includes those jobs explicitly run in the interactive compute partition. Adding the traditional UNIX tasks executed on the service nodes would significantly increase this number. As on the C90, the amount of CPU time used by batch jobs is over 90% of the total time used. The number of interactive jobs on the Paragon exceeds the number of batch jobs by over a factor of 4. Thus the Paragon is supporting a large interactive job mix with a concomittant need for fast turn-around, in addition to a production workload that is processed through batch. The preferred number of nodes for the batch jobs is 64. Thus the average batch job execution time is about two hours.

Up to 25 login sessions are simultaneously supported on a single

service node on the Paragon. To support more users, additional service nodes are added to the system. Similar scaling is used for disk support, with a separate MIO node used to control each 5 GByte RAID array. Adding more disk to the Paragon is accomplished by adding more MIO nodes.

Since batch compute jobs tend to require a smaller subset of the UNIX system call functionality, one important feature of the Paragon is the ability to run different operating system kernels on different nodes. The Sandia National Laboratory and the University of New Mexico have developed a "nanokernel" (called SUNMOS) that is less than 256 kBytes in size that supports high-speed message passing between nodes. Bandwidths as high as 160 MB/sec have been reported for the SUNMOS operating system [4]. The reduced size of the operating system allows larger in-core problems to be run.

Operating System Support for Heterogeneous Systems

The Paragon architecture can consist of nodes with both varying hardware and software capabilities. Hence a job mix scheduling system must recognize different node types and schedule jobs accordingly. At SDSC, such a system is in production use. Modifications have been made to the NQS batch system to support scheduling policies and packing of jobs onto the 2-D mesh. Job packing is done by a modified 2-D buddy algorithm. Scheduling is controlled by organizing nodes into uniform node sets, with assignment of lists of node sets to each NQS queue. Jobs submitted to a particular queue are then scheduled to run on only those nodes belonging to the node sets associated with the queue. This allows jobs to be scheduled to use large memory nodes, or to use nodes

that are executing the SUNMOS operating system. The scheduling policy picks the highest priority job for execution. If not enough nodes are available, nodes may be held idle until the job can fit on the mesh.

Superlinear Speedup

Heterogeneous systems can be used to improve individual application performance, as well as to support productivity requirements. For applications that can be decomposed into multiple tasks, performance can be increased by assigning each task to hardware/software systems that execute that task the quickest. An example is assigning sequential set-up tasks to nodes with very fast execution rates, while assigning highly parallel solution algorithms to parallel systems. For problems that iterate between job setup and job solution, it is possible to pipeline the calculations and do most of the work in parallel. If the execution rate of each task is sufficiently faster on different compute platforms, a superlinear speedup can be achieved. The solution time obtained by distributing the application can be faster by a factor larger than the number of tasks into which the application is decomposed.

Simple algebraic equations can be derived to illustrate this effect. Consider two tasks, a sequential task, 1, and a highly parallel task, 2, that are executed iteratively on two compute platforms, a fast sequential platform, A, and a fast parallel platform, B. After the initial setup for task 1, data is pipelined through multiple iterations until convergence is reached. Thus on average, task 1 and task 2 can execute in parallel. The execution time on platform A is given by

$$T_A = T_{A1} + T_{A2}$$

where T_{A1} is the time to execute task 1 on platform A and T_{A2} is the time to execute task 2 on platform A. With similar definitions, the time for execution on platform B is

$$T_B = T_{B1} + T_{B2}$$

Assume that task 1 executes faster on platform A, and task 2 executes faster on platform B. The execution time for the application distributed between the two platforms and executing in parallel is the maximum of T_{A1} and T_{B2} .

The speedup, S , is the ratio of the minimum of the stand alone execution times on the two platforms, divided by the execution time on the distributed system. The speedup is then given by

$$S = \min (S_A, S_B)$$

$$S_A = T_A / \max (T_{A1}, T_{B2})$$

$$S_B = T_B / \max (T_{A1}, T_{B2})$$

Each execution time can be modeled as the amount of work (N_1 is the number of operations for task 1) divided by the execution rate of the particular platform (R_{A1} is the execution rate of task 1 on platform A). Thus

$$T_{A1} = N_1 / R_{A1}$$

$$T_{B2} = N_2 / R_{B2}$$

The speedup can be calculated as a scaled function of the relative amount of work, h , of the two tasks, where

$$h = N_1 / N_2 * R_{B2} / R_{A1}$$

The maximum obtainable speedup can then be shown to depend only on the relative execution rates of the two tasks on the two platforms. A plot of the dependence of the speedup versus the relative amount of work between the two tasks is shown in Figure 2 when the ratio $RA1/RB1$ is greater than the ratio $RB2/RA2$.

The maximum attainable speedup is given by

$$S = 1 + RB2/RA2.$$

When $RB2$ is greater than $RA2$, then $S > 2$, and superlinear speedups are achievable. Note that there can be a substantial range in the relative load balance over which superlinear speedup is attainable. The speedup is given by the lower peak and corresponds to the slower running task on platform A being processed on platform B in the same amount of time as the faster task on platform A.

The maximum speedup corresponds to perfect load balancing, which occurs when both of the distributed tasks execute in the same amount of time. Thus, the maximum speedup occurs when

$$TA1 = TB2$$

or

$$h = 1$$

Summary

Scalable heterogeneous parallel architectures are able to support the heterogeneous workloads seen on present vector supercomputers. They achieve this by assigning different types of work to different hardware resources. On the Paragon, the ability to schedule jobs in an environment with nodes with different amounts of memory and even different operating systems is necessary for handling heterogeneous work loads. By decomposing applications into multiple tasks, it is possible to take advantage of heterogeneous architectures and achieve superlinear speedups, with applications decomposed into two tasks executing over twice as fast as the original.

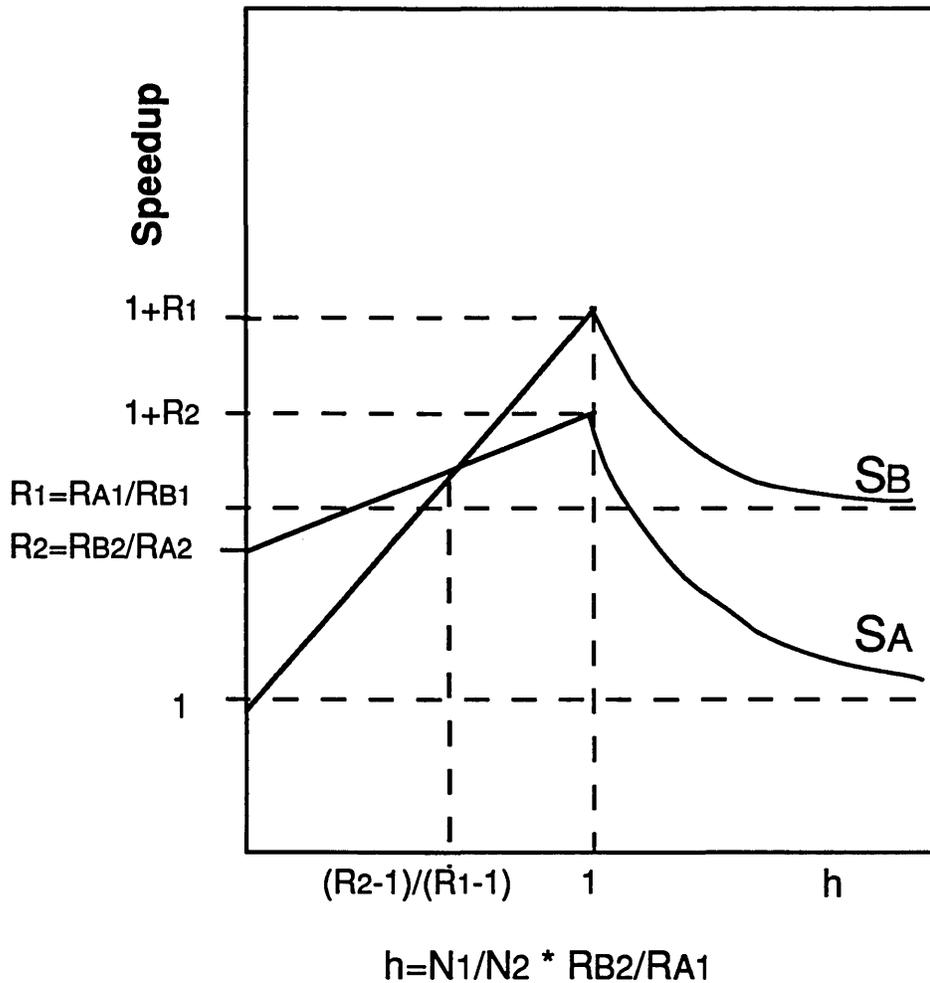
Acknowledgements

This work was funded in part by the National Science Foundation under Cooperative Agreement Number ASC-8902825 and in part by the National Science Foundation and Advanced Research Projects Agency under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives. The support of Mike Vildibill and Ken Steube in generating the statistics is gratefully acknowledged.

All brand and product names are trademarks or registered trademarks of their respective holders.

Figure 2

Speedup versus Scaled Load Distribution



References

1. Moore, Reagan W., "UNICOS Performance Dependence on Submitted Workload," Proceedings, Twenty-seventh Semiannual Cray User Group Meeting, London, Great Britain (April 1991), GA-A20509.
2. Moore, Reagan W., Michael Wan, and William Bennett, "UNICOS Tuning Strategies and Performance at the San Diego Supercomputer Center," Proceedings, Twenty-sixth Semiannual Cray User Group Meeting, Austin, TX (Oct. 1990), GA-A20286.

3. Wan, Michael and Reagan Moore, "Dynamic Adjustment/Tuning of UNICOS," Proceedings, Twenty-fifth Semiannual Cray User Group Meeting, Toronto, Canada (April 1990), GA-A20062.
4. Private communication, Rolf Riesen, Parallel Computing Science Department, Sandia National Laboratories.

CRAY T3D Project Update

Steve Reinhardt
Cray Research, Inc.
655F Lone Oak Drive
Eagan, MN 55121 USA
spr@cray.com

This paper describes significant CRAY T3D project events which have occurred since the last CUG meeting, emphasizing those events which will effect how programmers or users will use the machine and the performance they can expect.

1.0 Introduction¹

At the time of the last CUG meeting, in September, 1993, in Kyoto, the first customer CRAY T3D system had been shipped to the Pittsburgh Supercomputing Center. Hardware was stable, software was growing out of its infancy, and performance results were available beyond common industry kernels. Currently the CRAY T3D system is shipping in volume, and customers are using the CRAY T3D successfully for production computing and MPP application development. Topics covered in this paper include: shipments, reliability, CRAY T3D hardware plans, software plans, performance (kernel, I/O, and application), and third-party application work.

2.0 Shipments

The CRAY T3D architecture spans system sizes from 32 to 2048 processing elements (PEs). As of March 1994, we have shipped nine systems to customers. Those customers

1. The work described in this paper was supported in part by the Advanced Research Projects Agency (ARPA) of the U.S. Government under Agreement No. MDA972-92-H-0002 dated 21 January, 1992.

include governmental, industrial, and academic organizations. The industrial shipments include seismic and electronics customers. Machines reside in Europe, Japan, and the United States. Shipments include all of the chassis types which will be built: multi-chassis liquid-cooled (MC), single-chassis liquid-cooled (SC), and multi-chassis air-cooled (MCA). The largest customer system contains 256 PEs.

3.0 Reliability

CRI developed and delivered the CRAY T3D in 26 months, and some customers expressed concerns about the reliability of a machine developed so quickly. Given the small amount of total CRAY T3D experience we have, we cannot call the data conclusive, but some trends are already emerging. Overall the reliability of CRAY T3D systems is growing quickly. We plan to end the year 1994 with an MTTI of 1 month.

Effect on Y-MP host. Many current CRI customers wish to add an MPP component to their production system, but are extremely attentive to any impact this may cause to their existing CRAY Y-MP or CRAY Y-MP C90 production workload. Because of these concerns, we designed the CRAY T3D hardware and software to be isolated from the

Hardware Update

normal functions of the host, in order to minimize reliability problems. In practice this has worked well. We have had some early growing pains, which we believe are now behind us. Even including these early growing pains, our data shows that for every six interruptions caused by host hardware and software, the CRAY T3D system has caused one extra interruption.

CRAY T3D itself. In isolation, the reliability of the CRAY T3D system has been about 1 week. The hardware MTTI has been about 7 weeks. The software MTTI has been about one week.

Many customers were concerned about the binary-only release policy for CRAY T3D software because of the potential for slow response to problems observed on site. In practice, this has turned out not to be an issue. The OS has a handful of different packages, each of which can usually be upgraded separately. This allows us to deliver tested changes quickly. The infrastructure (tools and processes) is based on that being used by CRI compilers, which have been binary-only for several years, and hence is well proven.

4.0 Hardware Update

When we announced the CRAY T3D two memory sizes were quoted, 16MB (2MW) and 64MB (8MW). Several of the early systems were shipped with 16MB memories. We have now shipped a 64MB memory system.

We plan to allow the follow-on system to the CRAY Y-MP/EL to be a host to a CRAY T3D system during 1995.

5.0 Software Plans

The software for the early shipments enabled users to run production jobs effectively and to develop further MPP applications. Future software releases will provide better performance, ease of use, and flexibility.

5.1 UNICOS MAX operating system

Release 1.1. (2Q94) Improvements to UNICOS MAX are being released in monthly updates. By the time of release 1.1, all OS support for the CRAFT programming model will be in place. Resilience will be improved by the ability

to switch in the redundant hardware nodes more easily. Scheduling will be improved by allowing small programs to leap-frog in front of large programs which are waiting for resources; large programs will wait only a finite time.

Release 1.2 (4Q94) I/O connectivity and performance will be improved by the delivery of Phase II I/O, which allows a "back-door" channel from a Model E I/O cluster to connect directly to a CRAY T3D I/O gateway. This will especially improve the I/O picture for CRAY T3D customers who have host systems with few processors. Machine sharing will be improved by the implementation of rolling. Rolling enables a running program to be suspended, "rolled" out of its partition to secondary storage and all resources freed, another program to be run in the partition, and then finally for the original program to be rolled in and resumed. With rolling, very large, long-running "hog" jobs can co-exist with many small development programs.

Release 1.3 (1H95) The delivery of Phase III I/O will increase the I/O connectivity and performance again, enabling Model E IOCs to be connected directly to the CRAY T3D for both data and control, and thus allowing I/O to scale with the size of the CRAY T3D and be less controlled by the size of the host.

5.2 Compilers/Programming Environments

Release 1.1 (2Q94) The CRAFT programming model will enable users to program the CRAY T3D as a global-address-space machine, with data parallel and work-sharing constructs. We expect that many applications can be ported to the CRAY T3D more quickly with CRAFT than with a message-passing decomposition. The 1.1 release will allow C++ users to exploit the power of the CRAY T3D system for their programs, with compilation and debugging abilities and the class libraries most frequently used for scientific computations. Access to multi-PE operation from C++ will be via the PVM message-passing library.

3Q94 Users whose applications are dominated by operations that need only 32-bit data and operation will gain a significant performance improvement from the release of a signal processing option in Fortran in the third quarter of 1994. A subset of the Fortran 90 language and the mathematical intrinsic functions, suitable for signal processing, will be provided, along with visual tool support. Access to multi-PE operation will be via PVM.

4Q94 Users who do I/O which is spread across the memory of multiple processors will be able to do this more easily with the release of *global I/O*, which simplifies the task of doing I/O on shared objects and synchronizing access to files which are shared among PEs.

CF90 2.0 (1H95) The full Fortran 90 language will be delivered to CRAY T3D users with release 2.0 of the CF90 programming environment. Access to multi-PE operation will be via PVM. Implementation of the CRAFT model within CF90 is being scheduled.

Users will see improving application performance throughout this period as a consequence of further compiler and library optimizations. (See below under Kernel performance.)

The CRAFT programming model will deliver, we believe, an appropriate balance between the programmer's ease of use and the compiler's ability to deliver high performance. [Pase94] Many researchers believe that the HPF language will deliver good portability. [HPF93] Each of these languages is a global-address-space model for distributed memory computers. Widespread MPP applications development depends on the emergence of a standard language. We believe that the implementation of each of these languages will add to the body of knowledge about languages of this type. We will expect that these efforts will both contribute to the Fortran 95 standard committee, and that is where we will expect to resolve any conflicts between the two.

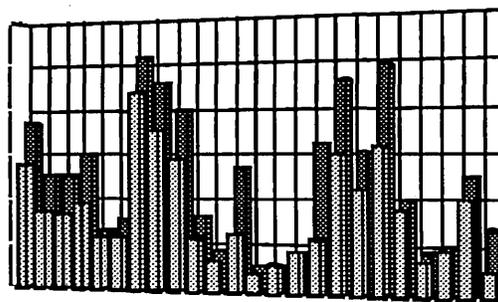
6.0 Performance

6.1 Livermore loops

The Livermore Fortran Kernels measure the performance of a processor for certain common operations. Figure 1 displays the performance of the CRAY T3D single-PE and

compiler in September of 1993 in the front row and the current performance in the back row.

Livermore Loops



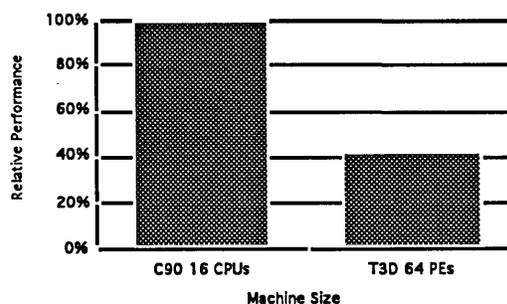
6.2 I/O performance

For 2 MB transfers, the CRAY T3D system can sustain across one HISP/LOSP channel pair more than 100 MB/s to a disk file system. When using 4 channel pairs in parallel, 4MB transfers can sustain more than 350 MB/s to disk.

6.3 Seismic Application Performance

Three-dimensional pre-stack depth migration describes the Grand Challenge of seismic computing. The application requires a high computational rate, but especially a high I/O rate. A CRAY Y-MP C90 implementation of this method was one of the 1993 Gordon Bell Award winners. The 3DPSDM program implements this technique for the CRAY T3D in Fortran and message-passing with some assembly code used [Wene93]. The absolute performance of 64 CRAY T3D PEs is 42% of the performance of a C90/16. The CRAY T3D is about 3.5 times more cost-

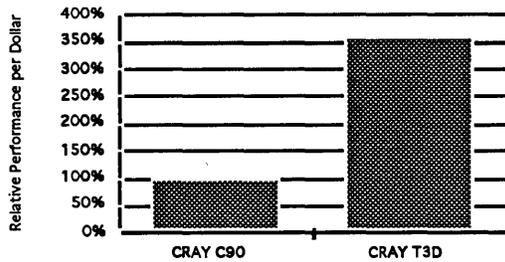
3D Prestack Migration Absolute Performance



Performance

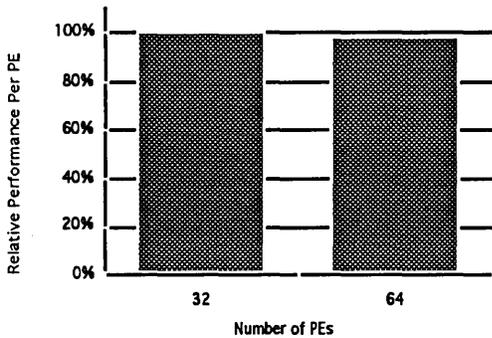
effective for this application.

3D Prestack Migration Performance Per Dollar



The application scales very well on many PEs.

3D Prestack Migration Scaling on CRAY T3Ds

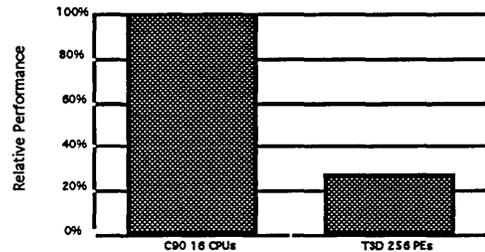


6.4 Climate Modeling Application Performance

The Parallel Ocean Program (POP) performs climate modeling calculations on scalable computers. [Duko93, Duko94] The program is structured as an SPMD computation, with the overall domain being decomposed into a block on each PE. Its basic structure is representative of many problems which devolve to the solution of partial different equations. On other MPPs, POP has spent more than 25% of its time communicating; on the CRAY T3D it spends less than 5% of its time communicating. POP run-

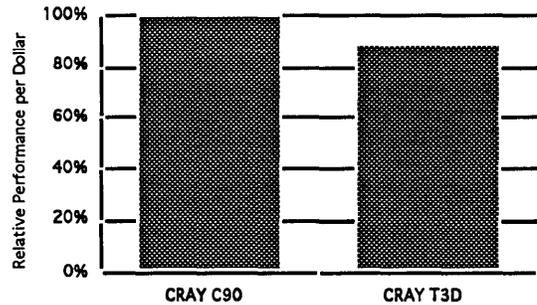
ning on 256 PEs of a CRAY T3D runs about 27% as fast as it does on a C90/16.

POP Absolute Performance



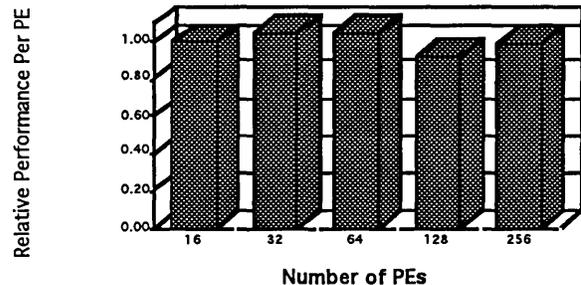
The price-performance of the CRAY T3D is 88% of the C90/16.

POP Performance Per Dollar



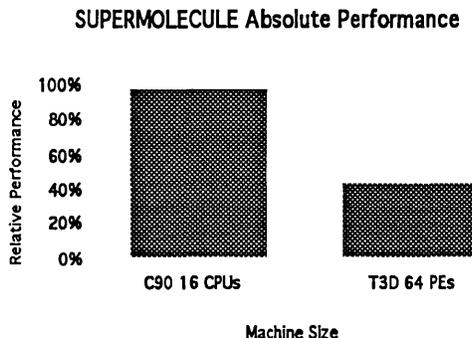
POP scales very well up to 256 PEs.

POP Scaling on CRAY T3Ds

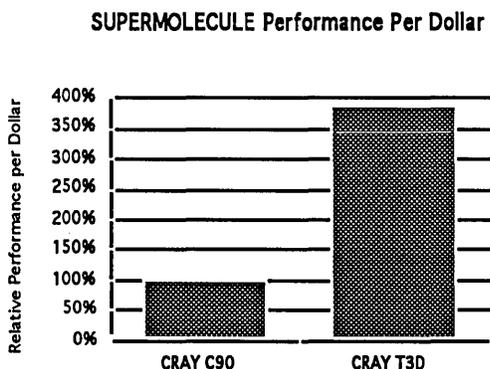


6.5 Chemistry Application Performance

The SUPERMOLECULE program is a public domain code whose structure and function is representative of third-party chemistry applications. [Sarg93, Feye93] It implements the Hartree-Fock method and is used to understand the interaction of drug molecules. The absolute performance of a 64-PE CRAY T3D is 45% of a C90/16.



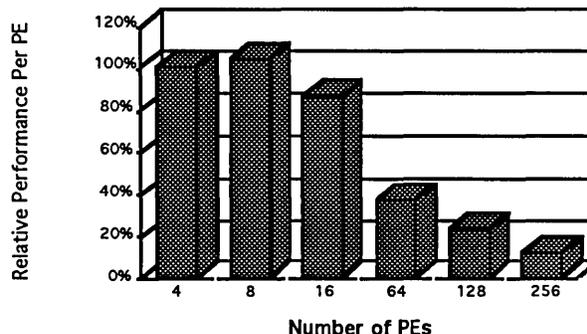
The price-performance of a CRAY T3D is almost four times that of the C90.



The scaling of SUPERMOLECULE, however, is not good, and in fact the time to solution does not decrease

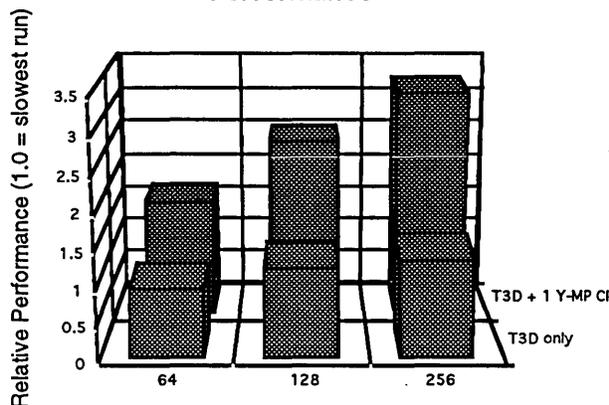
noticeably by using more than 64 processors. A matrix

SUPERMOLECULE Scaling on CRAY T3Ds



diagonalization step is being performed serially on the CRAY T3D; Amdahl's Law limits the speedups which are possible. However, because of the close linkage between the CRAY T3D and its parallel vector host, the serial portion of the code can be executed on a single processor of the host, and at much higher speed than a single processor of the CRAY T3D. When that portion of the code is run on a CRAY Y-MP processor, the program can use effectively more PEs on the CRAY T3D side. In this way a large pro-

SUPERMOLECULE Heterogeneous Performance



gram can exploit the coupled system for faster time to solution than either system could provide by itself.

7.0 Third-Party Application Work

The success of the CRI MPP project depends heavily on the availability of third-party applications programs to enable many users to ignore the complexity of a distributed memory computer and yet tap the very high performance of the CRAY T3D system. CRI is working with vendors of the following applications programs, with a goal of having some of these applications available from the vendor for the CRAY T3D system by the end of 1994.

chemistry	CHARMm Discover Gaussian X-PLOR
structural analysis	LS-DYNA3D PAMCRASH
CFD	FIRE FLO67 STAR-CD
electronics	DaVinci Medici TSuprem
petroleum	DISCO GEOVECTEUR
mathematical libraries	IMSL NAG Elegant Mathematics

8.0 Summary

The CRAY T3D computer system has been delivered to customers working in several scientific disciplines, and is enabling production MPP computing for those customers. Development of new MPP applications on the CRAY T3D system is fueling greater exploitation of the latent performance of the system.

9.0 References

[Duko93] *A Reformulation and Implementation of the Bryan-Cox-Semtner Ocean Model on the Connection Machine*, J.K. Dukowicz, R.D. Smith, and R.C. Malone. *J. Atmos. Ocean. Techn.*, 10, 195-208 (1993).

[Duko94] *Implicit Free-Surface Methods for the Bryan-Cox-Semtner Ocean Model*, J.K. Dukowicz and R.D. Smith. To appear in *J. Geophys. Res.*

[Feye93] *An Efficient Implementation of the Direct-SCF Algorithm on Parallel Computer Architectures*, M. Feyereisen and R.A. Kendall. *Theoret. Chim. Acta* 84, 289 (1993).

[Geis93] *PVM3 User's Guide and Reference Manual*, Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek and Vaidy Sunderam. Oak Ridge National Laboratory ORNL/TM-12187, May 1993..

[HPF93] *High Performance Fortran*, High Performance Fortran Language Specification version 1.0, May 1993. Also available as technical report CRPC-TR 92225, Center for Research on Parallel Computation, Rice University. To appear in "Scientific Programming."

[MacD92] *The Cray Research MPP Fortran Programming Model*, Tom MacDonald. Proceedings of the Spring 1992 Cray Users' Group Meeting, pp. 389-399.

[Pase94] *The CRAFT Fortran Programming Model*, Douglas M. Pase, Tom MacDonald, and Andrew Meltzer. To appear in *Scientific Programming*, 1994.

[Rein93] *CRAY T3D System Software*, Steve Reinhardt. Proceedings of the Fall 1993 Cray Users' Group Meeting, pp. 36-40.

[Sarg93] *Electronic Structure Calculations in Quantum Chemistry*, A.L. Sargent, J. Almlof, and M. Feyereisen. *SIAM News* 26(1), 14 (1993).

[Wene94] *Seismic Imaging in a Production Environment*, Geert Wenes, Janet Shiu, and Serge Kremer. Proceedings of the Spring 1994 Cray Users' Group (CUG) Meeting.

PARALLEL SESSIONS

Applications and Algorithms

Porting Third-Party Applications Packages to the Cray MPP: Experiences at the Pittsburgh Supercomputing Center

Frank C. Wimberly, Susheel Chitre, Carlos Gonzalez, Michael H. Lambert,
Nicholas Nystrom, Alex Ropelewski, William Young

March 30, 1994

1 Background

Early in 1993 it was decided that the Pittsburgh Supercomputing Center would acquire the first Cray T3D MPP supercomputer to be delivered to a customer site. Shortly after that decision was made we began a project to implement a number of widely used third-party packages on the new platform with the goal that they be available for production use by the time the hardware was made available to the PSC user community.

The wide use of such packages on the Center's Cray C90 (C916/512) led us to appreciate the importance of their availability. Approximately 30 to 40 percent of the cycles on the C90 are delivered to users of applications packages. Previously acquired massively parallel supercomputers at the Center had seen less widespread use than the vector supercomputers, probably because of the lack of such packages. These other MPP's were not underutilized but they were used by a relatively smaller set of users, who had developed their own codes.

In selecting the targets for the porting effort we took into account: demand for the package on the C90; whether we had a source license and the right to modify the program and make the result available to users; and, whether message passing parallel versions already existed which would give us a headstart on a T3D version. Based on these criteria we selecting the following packages:

- * GAMESS
- * Amber 4
- * CHARMM
- * MaxSegs
- * Gaussian 92

In addition, later in the year we began evaluating FIDAP and X-PLOR as additional candidates for porting.

Although we did not have access to T3D hardware until later in the summer extensive porting began early in the year by means of access to T3D emulators running at CRI and shortly thereafter on PSC's own C90. The emulator proved to be an excellent tool for validating program syntax and

correctness of results. Its usefulness for program optimization was limited in that performance data was restricted to information about locality of memory references. Since several programs worked correctly on the emulator before the hardware became available we turned to the T3D simulator, running on a Y-MP at CRI, as a means of testing performance. Although the simulator was important in operating system development it was not as useful as we had hoped since for testing applications programs it ran too slowly to permit realistic runs. The hardware became available shortly after these attempts to use the simulator so this was not a significant impediment to progress.

By the time of delivery of the 32 PE T3D to PSC in August 1993 all of the five initially selected packages ran in one version or another on the emulator. Some of them ran "heterogeneously" between the C90 and the emulator and some ran "standalone" within the emulator. Since Heterogeneous PVM was not available at that time, a PSC developed communications package, DHSC (for Distributed High Speed Communication), was used for communication between the two platforms. Within a few weeks after delivery of the hardware versions of all five packages were running either on the T3D or distributed between the T3D and the C90. Again, DHSC was used for the heterogeneous programs. There were various problems with the run time system and with the CF77 compiler (for instance, incorrect handling of dynamic arrays) which prevented the programs from running immediately on the hardware even though they had run on the emulator. CRI was very responsive to the discovery of these software problems and as a result of our efforts and support from Cray personnel we were pleased with the progress we had made.

We have recently begun to place more emphasis on performance as opposed to establishing program correctness. As the focus has moved to performance we have been getting regular hardware upgrades. The latest occurred in early March of the current year, and we now have 256 PE's and four I/O gateways on the T3D; an additional 256 PE's are scheduled to be installed in the early summer of 1994.

Before presenting the current status of several of the porting projects we should comment on some general themes. First, performance figures given below should be understood in context. The CRAFT FORTRAN programming model is not yet available to us. All parallel implementations have

been done using explicit message passing. The compiler does not yet produce fully optimized code nor have the applications themselves been fully reorganized to exploit the power of the T3D at this time. The PE's currently have 16 MB of memory which has made it necessary to compensate in ways which may adversely affect performance. For the heterogeneous programs, performance has been impacted by the relatively slow speed of the C90 to T3D connection (the I/O gateways); this is especially true for Heterogeneous PVM but also for DHSC. The hardware is capable of 200 MB/sec but we have realized process-to-process throughput of only about 10 MB/sec. We expect this to improve substantially as Heterogeneous PVM is expanded and improved.

2 CHARMM and the Molecular Dynamics Kernel

CHARMM (Chemistry at HARvard Macromolecular Mechanics) is a widely used program to model and simulate molecular systems using an empirical energy function. In the 10 years since CHARMM was developed by Brooks [1] and co-workers, a great deal of work has gone into developing optimized versions for a large number of computer systems including vector [2] and parallel [3], [4], [5] machines. Because of its heavy usage at PSC and the nature of the algorithms used by CHARMM we felt that it was an excellent candidate for porting to the T3D.

The principal focus of most CHARMM-based research done by PSC users is simulating poly-peptides and proteins in an aqueous environment. As a starting point we developed a specialized version of CHARMM for this problem and are currently running production calculations while we explore algorithms and methods for developing a full-featured version of CHARMM. This initial port extends ideas in heterogeneous computing previously explored at the PSC [6] using a Cray Y-MP and a Thinking Machines Corporation CM-2 and CM-5. This heterogeneous computing approach couples a highly-optimized code to simulate molecular solvents [7] over a high-speed channel using either DHSC routines or network PVM [8].

In previous computational experiments in distributed computing we were able observe good performance in small demonstration calculations. But in scaling these calculations up for production we faced a large number of technical problems, including those related to data format conversion. With the arrival of the T3D and its pairing with the C90, we had a heterogeneous computing system from a single vendor and were hopeful that these issues could be resolved. At present, many of those issues have been resolved and we are currently running production calculations. For a wide range of benchmark problems we currently see speedups of 2 to 3 in run time using a single C90 CPU and 32 T3D PE's over the same calculation done on a single C90 CPU alone.

3 GAMESS

GAMESS (General Atomic and Molecular Electronic Structure System) [9] [10] is a quantum chemistry code currently being developed at Iowa State University. It is written in standard Fortran 77 and runs on a variety of platforms. The program exhibits poor performance on Cray vector hardware. However, as distributed by the author, it provides support for message-passing parallelism. This is accomplished through calls to the TCGMSG message passing library [11]. Because most computation is done in parallel regions and these regions are scattered through the code, GAMESS is better suited to a standalone T3D implementation than it is to the heterogeneous C90/T3D platform.

Because PVM is the natural message-passing mechanism on the Cray T3D, the TCGMSG library was replaced by a set of wrappers to call the appropriate PVM routines. The code would not run under the T3D emulator, so it was necessary to do development work on the PSC workstation cluster. Once the T3D hardware arrived and the software stabilized, GAMESS ran. However, because of current memory limitations on the T3D, it is limited to direct (in memory) calculations with about 200 basis functions. This is marginally enough to handle interesting problems.

GAMESS running on the T3D scales well with the number of processors. Communications accounts for about two percent of the total time and load balancing is at the five-percent level. On a 110 basis-function direct SCF gradient calculation with 32 PEs, the two-electron gradient routine is over 99.5% parallel and the overall efficiency is just over 50%. Direct calculations run in about the same time on four PEs on the T3D as on one processor of a C90.

4 Amber 4

Amber 4[12] is a suite of programs designed for molecular modeling and simulations. It is a robust package prominent in the toolkits of computational chemistry and biology researchers, and its methods, tailored to the study of macromolecules in solution, have found widespread applicability in modeling complex biochemical processes and in the pharmaceuticals industry. Amber's genesis was in the application of classical mechanics (*i.e.* integration of Newton's equations of motion) to large molecular assemblies using fitted potentials to determine low-energy conformations, model biological processes, obtain bulk properties, *etc.* New versions of Amber have since introduced capabilities to treat nuclear magnetic resonance data, support free energy perturbation calculations, and otherwise implement the functionality required by its research audience.

The Amber 4 package consists of sixteen individual programs, loosely categorized as preparatory programs (prep, link, edit, and parm), energy programs (minmd, gibbs, sander, nmode), and utility programs (ana1, mdana1, nmana1, lmana1, nucgen, pdbgen, geom, and a set of tutorial

shell scripts). The preparatory programs address the construction of Amber 4 input. They generally consume a small amount of computational resources and are run a small number of times as the first step of any given calculation. The energy programs perform the real work of minimizing structures and propagating trajectories through time. Computationally they are the most demanding programs in the Amber 4 system, and they are often run multiple times to model different environments and initial conditions. The four energy programs together, including library routines shared between them, comprise only 53% of Amber 4 package's source code. The utility programs analyze configurations, compute various properties from the system's normal modes, and interact with Brookhaven Protein Data Bank[13, 14] (PDB) files. As with the preparatory programs, the utility programs consume an relatively insignificant portion of computational resources when compared with the energy programs.

4.1 Parallel implementations

The clear association of heavy CPU requirements with the energy programs suggests them as ideal candidates for implementation on distributed and parallel computers. Distributing even moderate-sized programs such as these can be laborious, so the PSC was fortunate to receive distributed versions of *minmd* and *gibbs* based on PVM[8] from Professor Terry P. Lybrand and his research group at the University of Washington. (Work is also underway at the University of Washington on *sander*.)

The PSC's initial choice for conversion to the Cray T3D was *minmd* because its relevance to computational chemistry and biology, the number of CPU cycles it consumes, its size, and the time frame in which the PVM version was obtained. *Minmd* performs minimization, in which the atoms' positions are iteratively refined to minimize the energy gradient, and molecular dynamics, in which the atoms' coordinates are integrated in time according to Newton's equations of motion. Typical system sizes in contemporary research range from on the order of 10^2 to upwards of 10^5 atoms. Realistic simulations can entail up to on the order of 10^5 integration time steps, rendering the integration phase of the simulation dominant and also daunting.

Lybrand's PVM-based version of *minmd*, subsequently referred to as the *distributed* implementation, embodies a host/node model to partition a simulation across a networked group of workstations. One process, designated as the *host*, spawns a specified number of *node* processes to compute non-bonded interactions. All other aspects of the calculation are performed on the host, which efficiently overlaps its own computation with communication to and from the nodes.

Work on two distinct implementations *minmd* is well underway: a *standalone* implementation which runs solely on the Cray T3D, and a *heterogeneous* version which distributes work between the Cray C90 and the Cray T3D. The standalone version employs Cray T3D PVM to exchange data between processing element (PE) 0 and all other PE's. There

is occasional synchronization between the nodes, but no explicit node-node data transfer. The heterogeneous version of *minmd* uses CRI Network PVM (also known as "Hetero PVM") to communicate between the Cray C90 (host) and the T3D PE's (nodes). Again, no node-node data transfer is necessary.

The standalone and heterogeneous implementations of *minmd* each have their own advantages and disadvantages. The standalone version offers the greatest potential performance because of the low-latency, high-bandwidth I/O available on the Cray T3D hardware. Its principal disadvantage is that conversion from distributed host/node code to standalone code is tedious and error-prone because two separate sets of source code must be merged. This results in a high maintenance cost for a package such as Amber 4 which is constantly evolving. The heterogeneous implementation currently suffers from the low efficiency of the C90-T3D communications mechanism, but it is very easily derived from the distributed source code. The changes to PVM are trivial, and the only extensive changes required concern file I/O and the processing of command line arguments. Communications rates between the C90 and the T3D are expected to improve with time, so for now development and instrumentation of both implementations of *minmd* will continue.

Preliminary timing data has already been obtained for distributed *minmd* running on the PSC's DEC Alpha workstation cluster and for the heterogeneous *minmd* running between the Cray C90 and T3D. Debugging of the standalone Cray T3D implementation of *minmd* is in its final stages, and timings are expected shortly.

4.2 Acknowledgements

The initial PVM-based implementations of *minmd* and *gibbs* were developed and provided by Professor Terry P. Lybrand and his research group at the University of Washington.

5 MaxSegs

MaxSegs [15] is a program written by the PSC for genetic sequence analysis. *MaxSegs* is designed to take a experimental DNA/RNA or protein query sequence and compare it with a library of all categorized DNA/RNA or protein sequences. Searching categorized sequences with an experimental sequence is useful because it helps the researcher locate sequences that might share an evolutionary, functional, or a biochemical relationship with the query sequence¹. The *MaxSegs* program is written in standard FORTRAN-77 and is highly optimized to run on Cray vector supercomputers. For a typical protein the *MaxSegs* program operates at about

¹There are currently about 40,000 categorized protein sequences ranging in length from 2 to 6000 characters. The average size of a typical protein sequence is approximately 300 residues. There are approximately 170,000 DNA/RNA sequences with lengths ranging from 100 to 200,000. The length of a typical DNA sequence is about 1000.

230 million vector operations per second on the PSC's C90².

MaxSegs was also one of the first programs distributed between the Cray YMP and the Thinking Machine Corporation's CM-2 supercomputer at the Center [16]. This project helped to show that for large problems, two supercomputers could indeed cooperate to achieve results faster than either supercomputer alone could achieve. The CM2 code was implemented using data parallel methods; each virtual processor on the CM2 received a unique library sequence residue to compare with a broadcast query sequence residue. In this implementation, many library sequences could be compared with a query sequence simultaneously. In addition to comparing many library sequences with a query sequence at once, this implementation also requires the use of very little per-processor memory. The disadvantages of this implementation include an enormous amount of nearest neighbor communication, a startup and finishing penalty in which nodes process zeros and the pre-processing of enormous, frequently updated sequence libraries³. Although programming in this style has a number of disadvantages, the results have been impressive enough to allow sequence analysis software implemented on SIMD machines, to gain acceptance within the sequence analysis community.

Both published research [17], [18] and unpublished research by the biomedical group at the PSC have shown that if node processors have sufficient memory, a MIMD style of programming can be applied to the sequence comparison problem yielding performance superior to the performance reported on machines using data parallel approaches. In this style each processor is given the experimental query sequence and unique library sequences to compare. The results of these comparisons are collected by a single host processor. One advantage to this implementation is that superior load balancing can be achieved, without having to pre-sort the frequently updated sequence databases. The increased load balancing capabilities also make this implementation suitable for a wide selection of sequence comparison algorithms, such as Gribskov's profile searching algorithm [19]. In addition to providing superior load balancing overall communication is also reduced; communication only occurs when the sequences are sent to the processors, and the results of the comparisons are collected back at the host. The disadvantages of this method are that the communication patterns are irregular, there is the potential for a bottleneck to occur at the host processor, and the nodes must have sufficient memory to perform the comparison.

We have decided to implement the code on the T3D using the MIMD approach in two different ways. The first way uses the C90 as the host processor, directing the T3D to perform the work of comparing the sequences. The second method uses PE0 on the T3D as the host processor, leaving the remaining T3D processors to compare the sequences. Preliminary re-

²The C90 version of MaxSegs is available upon request.

³The startup and finishing penalties result from the recursive nature of the sequence comparison algorithms; to improve efficiency on a SIMD machine, categorized sequences must be sorted according to size. See [16].

sults indicate that the overall communication speed between the T3D and the C90 is currently insufficient to consider the first approach as a viable alternative to using the native C90 code. However, the second approach is very promising, preliminary results indicate that 32 T3D processors take only 25% more time than a single C90 CPU and that 64 T3D processors can match the performance of a single C90 cpu. These performance results are on preliminary code, and results indicate that communication, rather than computation is the main bottleneck. Using some simple communication reduction techniques, we expect the results to improve dramatically.

5.1 Acknowledgements

This work was developed in part under the National Institutes of Health grants 1 p41 RR06009 (NCR), and R01 LM05513 (NLM) to the Pittsburgh Supercomputing Center. Additional funding was provided to the Pittsburgh Supercomputing Center by the National Science Foundation grant ASC-8902826.

6 Conclusion

Based on our experience we would say that the problem of porting "dusty decks", or, more accurately, large pre-existing packages to a modern, high performance MPP platform is difficult but possible. The difficulty varies depending on the structure of the code, the intrinsic nature of the algorithms, the existence of other message passing versions, and several other factors. In the best cases, the effort is clearly worthwhile. We have seen impressive performance even in the absence of obvious optimizations of both the compilers and the applications programs themselves. It seems clear that in many cases the throughput, measured in the amount of scientific research accomplished per unit time, realized at the PSC will be substantially increased by the availability of these packages either on the T3D or on the heterogeneous platform (C90/T3D).

References

- [1] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. Charmm: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comp. Chem.*, 4:187-217, 1983.
- [2] J. E. Mertz, D. J. Tobias, C. L. Brooks, III, and U. C. Singh. Vector and parallel algorithms for the molecular dynamics simulation of macromolecules on shared memory computers. *J. Comp. Chem.*, 12:1270-1277, 1991.
- [3] B. G. J. P. T. Murray, P. A. Bash, and M. Karplus. Molecular dynamics on the connection machine. Tech-

- nical Report CB88-3, Thinking Machines Corporation, 1988.
- [4] S. J. Lin, J. Mellor-Crummey, B. M. Pettitt, and Jr. G. N. Phillips. Molecular dynamics on a distributed-memory multiprocessor. *J. Comp. Chem.*, 13:1022-1035, 1992.
- [5] B. R. Brooks and Milan Hodoscek. Parallelization of charmm for mimd machines. *Chemical Design Automation News*, 7:16-21, 1993.
- [6] C. L. Brooks III, W. S. Young, and D. J. Tobias. Molecular simulations on supercomputers. *Intl. J. Supercomputer App.*, 5:98-112, 1991.
- [7] W. S. Young and C. L. Brooks III. Optimization of replicated data method for molecular dynamics. *J. Comp. Chem.*, In preparation.
- [8] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. PVM 3 user's guide and reference manual. Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1993.
- [9] M. Dupuis, D. Spangler, and J. J. Wendoloski. *National Resource for Computations in Chemistry Software Catalog, Program QG01*. University of California, Berkeley, 1980.
- [10] Michael W. Schmidt, Kim K. Baldrige, Jerry A. Boatz, Steven T. Elbert, Mark S. Gordon, Jan H. Jensen, Shiro Koseki, Nikita Matsunaga, Kiet A. Nguyen, Shujun Su, Theresa L. Windus, Michel Dupuis, and John A. Montgomery, Jr. General atomic and molecular electronic structure system. *J. Comp. Chem.*, 14(11):1347-1363, 1993.
- [11] R. J. Harrison, now at Pacific Northwest Laboratory, v. 4.03, available by anonymous ftp in directory pub/tcgmsg from host ftp.tcg.anl.gov.
- [12] David A. Pearlman, David A. Case, James C. Caldwell, George L. Seibel, U. Chandra Singh, Paul Weiner, and Peter A. Kollman. *AMBER 4.0*. University of California, San Francisco, 1991.
- [13] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer, Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol.*, 112:535-542, 1977.
- [14] E. E. Abola, F. C. Bernstein, S. H. Bryant, T. F. Koetzle, and J. Weng. Protein Data Bank. In F. H. Allen, G. Bergerhoff, and R. Sievers, editors, *Crystallographic Databases - Information Content, Software Systems, Scientific Applications*, pages 107-132, Bonn/Cambridge/Chester, 1987. Data Commission of the International Union of Crystallography.
- [15] M. S. Waterman and M. Eggert. A new algorithm for best subsequent alignments with applications to trna-rRNA comparisons. *J. Mol. Biol.*, 197:723-728, 1987.
- [16] H. Nicholas, G. Giras, V. Hartonas-Garmhausen, M. Kopko, C. Maher, and A. Ropelewski. Distributing the comparison of dna and protein sequences across heterogeneous supercomputers. In *Supercomputing '91 Proceedings*, pages 139-146, 1991.
- [17] A. Deshpande, D. Richards, and W. Pearson. A platform for biological sequence comparison on parallel computers. *CABIOS*, 7:237-347, 1991.
- [18] P. Miller, P. Nadkarni, and W. Pearson. Comparing machine-independent versus machine-specific parallelization of a software platform for biological sequence comparison. *CABIOS*, 8:167-175, 1992.
- [19] M. Gribskov, R. Lüthy, and D. Eisenberg. Profile analysis. In R. Doolittle, editor, *Methods in Enzymology Volume 183*, 1990.

Some Loop Collapse Techniques to Increase Autotasking Efficiency

Mike Davis

Customer Service Division

Cray Research, Inc.

Albuquerque, NM

Abstract

Loop collapsing has limited application as a technique to improve the efficiency of vectorization; however, when applied to a nest of loops in which Autotasking is taking place, the resulting transformed loop structure can perform much more efficiently. This paper describes the loop structures that can be collapsed, discusses the techniques for collapsing these loops, and measures the efficiencies of the transformations.

1.0 Introduction

The Cray Research Autotasking Compiling System recognizes several forms of vectorizable, parallelizable work in Fortran code. These forms fit the general Concurrent-Outer-Vector-Inner (COVI) model, where outer loop iterations are executed concurrently on separate CPUs and inner loop iterations are executed in vector mode on each CPU. The kinds of iterative structures that cannot be recognized as parallelizable or that cannot be efficiently parallelized have been classified into groups according to their characteristics [1,2]. The descriptions of some of these groupings are as follows:

- a. The parallelized loop contains an insufficient amount of work over which to amortize the cost of initiating and terminating parallel processing (BPEP);
- b. The number of iterations in the parallelized loop is not sufficiently high to permit the use of all (or a large majority) of the machine's CPUs (LI);
- c. The parallel efficiency of the parallelized loop is limited by an ineffective work distribution scheme (LI);
- d. The amount of work being done on each iteration of the parallelized loop is so large that delays in task scheduling can result in significant reductions in achieved speedup (LI);
- e. The amount of work being done on each iteration of the parallelized loop varies greatly from one iteration to the next,

causing high overhead to distribute and synchronize tasks (VW);

- f. The parallel region is itself inside a loop that also contains a significant amount of serial work; this kind of code can potentially result in high overhead as the operating system repeatedly gives CPUs to the process for execution of the parallel region, then takes them away during execution of the serial region (RESCH).

The techniques described in this paper address these types of structures. Section 2 introduces the concept of iteration space, and how looping structures can be described by the shape of their iteration spaces. Section 3 describes coding techniques for collapsing nests of loops and gives intuitive explanations of why the transformations provide performance benefits. Section 4 covers the results of performance-testing the various collapse transformations. Conclusions and areas of future work are presented in Section 5.

2.0 Geometric Characterization of Iterative Structures

The number of different looping structures that can be constructed is literally infinite; the number that have actually been constructed is probably as large as the number of computer applications. But the vast majority of looping constructs can be grouped into a handful of categories. For the purposes of this paper, the best system for classifying loops is the geometric system. In this system, iterative structures are described by the shape of the iteration space. The shape will have as many dimensions as the iterative structure has loop nests. The shape is constructed by projecting it outward by one dimension for every loop in the loop nest. The following helps to illustrate this process:

The outermost loop in the structure (call it L1) is represented by a line; its length, in geometric units, is equal to the trip count of the loop (N1); this is represented in Figure 2.1.

Copyright (C) 1994, Cray Research, Inc. All Rights Reserved.

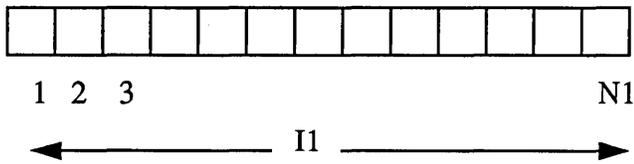


Figure 2.1

The next outermost loop (L2), inside of L1, is represented by projecting the shape to two dimensions. The width of the shape at a point I1 units along the edge formed by L1 is equal to the trip count of L2 (N2) on iteration I1 of L1; this is shown in Figure 2.2.

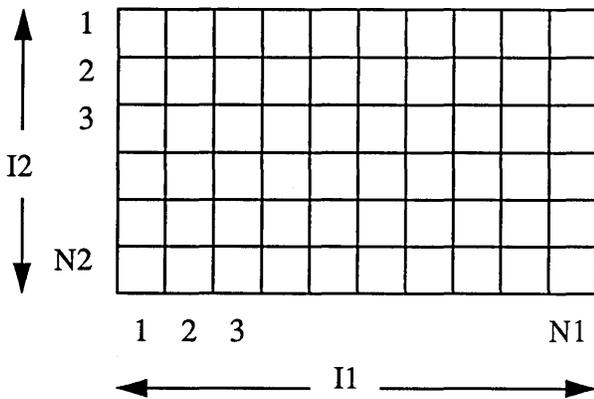


Figure 2.2

The next outermost loop (L3), inside of L2, is represented by projecting the shape to three dimensions. The depth of the shape at a point I1 units along the edge formed by L1 and I2 units along the edge formed by L2 is equal to the trip count of L3 (N3) on iteration I1 of L1 and I2 of L2; this is depicted in Figure 2.3. Quite often, the shape of the iteration space matches the shape of the data upon which the iterative structure operates, but this is not always necessarily the case. Some examples will help reinforce the concept of representing iterative structures geometrically

2.1 Linear Iteration Space

In a simple iterative structure consisting of only one loop, the iteration space is said to have a linear shape. The length of the line is equal to the trip count of the loop.

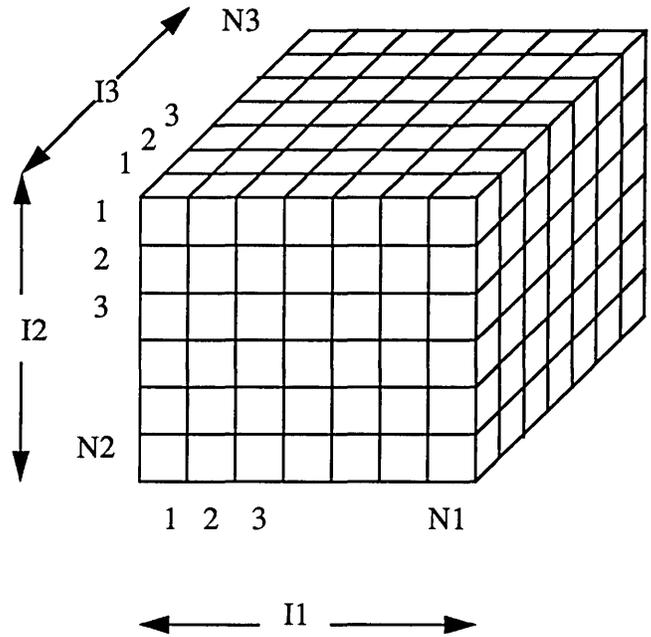


Figure 2.3

2.2 Rectangular Iteration Space

Consider the iterative structure of Figure 2.2. The iteration space of this structure has a rectangular shape. Its length is equal to N1, and its width at all points along its length is equal to N2.

2.3 Triangular Iteration Space

A structure whose iteration space is right-triangular would appear as shown in Figure 2.4. There are several different variations of the triangular iteration space, and each variation corresponds to a triangle with different spatial orientation and a specification of whether or not the triangle includes the "main diagonal." These variations can be distinguished by the form of the DO statement for the inner loop, as shown in Table 2.1.

2.4 Nevada Iteration Space

A more generalized expression for both the rectangular and triangular iteration spaces is that of the "Nevada" iteration space. This shape has both rectangular and triangular components, such that it is shaped like the state of Nevada. The coding structure that corresponds to this shape is shown in Figure 2.5. In this structure, the shape has a length of N1 and a width that varies from N2+N2D to N2+N1*N2D. N2D represents the magnitude

of the slope of the hypotenuse of the triangular component of the shape. If $N2$ is zero, then the shape degenerates into that of a triangle; if $N2D$ is zero, then the shape degenerates into that of a rectangle.

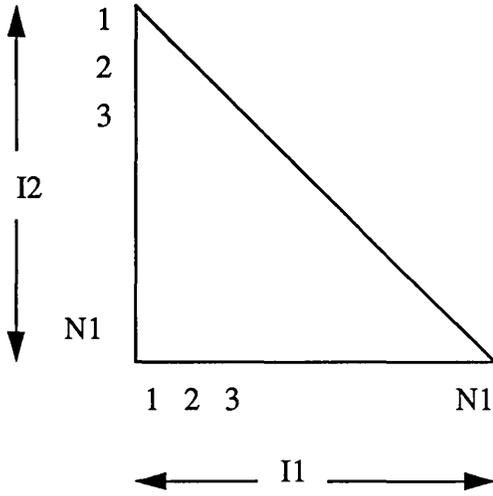


Figure 2.4

along its length is dependent upon the contents of some data structure that has been built prior to entering the coding structure. The histogram iteration space is shown in Figure 2.5. Here, the identifier $N2$ is an integer array of extent $(1:N1)$, and each element of $N2$ specifies the trip count of the inner loop. Geometrically, this corresponds to the notion that for a given "bar" $I1$, the "bar height" is equal to $N2(I1)$.

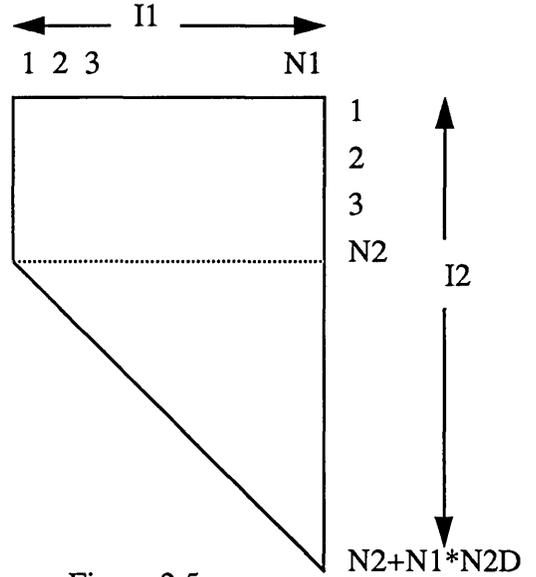


Figure 2.5

I2 index values	Orientation	Inc. Main Diag
1, I1	Upper Right	Yes
1, I1-1	Upper Right	No
I1, N1	Lower Left	Yes
I1+1, N1	Lower Left	No
N1+1-I1, N1	Lower Right	Yes
N1+1-I1, N1-1	Lower Right	No
1, N1+1-I1	Upper Left	Yes
1, N1-I1	Upper Left	No

TABLE 2.1: Variations of Triangular Iteration Space

2.5 Histogram Iteration Space

Another shape commonly encountered is that of a histogram. For this structure, the width of the shape at a given point

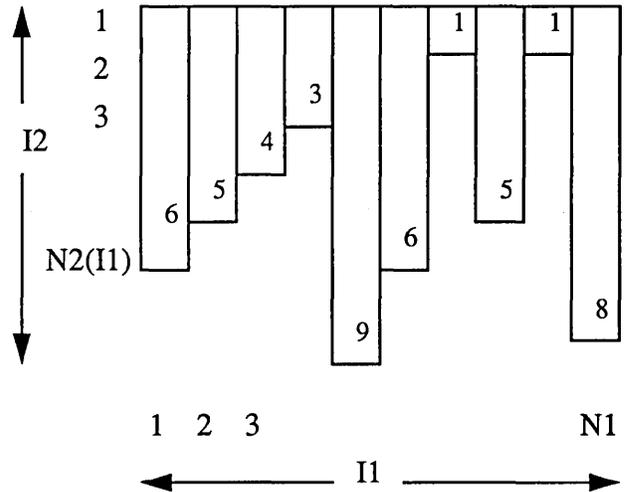


Figure 2.6

2.6 Porous Iteration Space

There is another characteristic of iterative structures that is worth considering for the purposes of this study. Recall from the previous section that the iterative structures that pose problems for Autotasking include those that have low trip counts and low amounts of work in the body. So far we have focused primarily on characterizing loops based on their trip counts. Now we consider how to describe loops in terms of the amounts of work contained within. Consider the iterative structure of Figure 2.7. Notice that the iteration space is rectangular, but the work is confined to a triangular subspace.

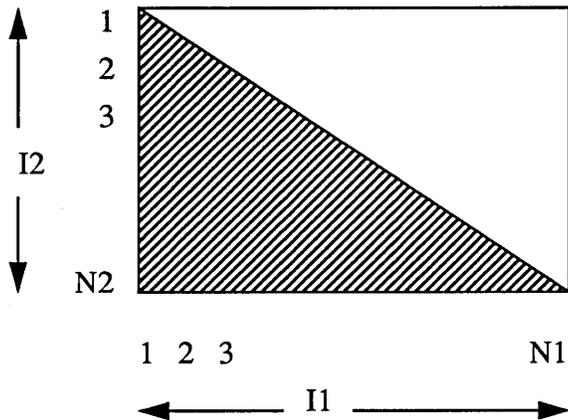


Figure 2.7

Now consider a more generic case, as depicted in Figure 2.8. Here, work is done only on iterations where the control variables suit some condition. The shape of the iteration space is rectangular, but its interior is porous; that is, certain cells of the space have substance while others do not. An iterative coding structure that executes varying amounts of work from one iteration to the next is analogous to a shape that varies in density from one cell to the next. In this work, we will restrict our study to structures that execute either some work or no work depending on the iteration, much like the one in Figure 2.8.

3.0 Loop Collapse Techniques

Loop collapsing is one of several loop transformation techniques for optimizing the performance of code. Some of the others include loop splitting, loop fusion, loop unrolling, and loop interchange. Each technique is suitable for its own class of loop structures, but loop collapsing is done primarily to increase the number of iterations that can be made available to the optimizing stage of

the compiler at one time. Collapsing nested loops can improve the vector performance of a loop structure because it increases the trip count of the vector loop, and hence increases the vector length [3,4]. For Autotasking, a collapsed nest of loops can perform better because the entire iteration space is handled in one parallel region, rather than just individual sections; furthermore, the programmer has more control over the granularity of the parallelism in a collapsed loop.

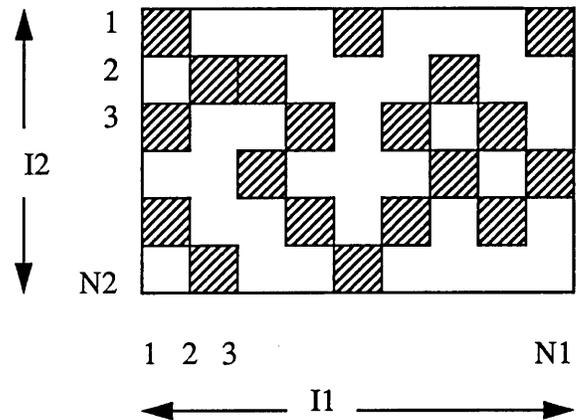


Figure 2.8

3.1 Collapsing the Rectangle

Probably the simplest loop structure to collapse is one with a rectangular shape, such as the one below:

```
DO I1 = 1, N1
  DO I2 = 1, N2
    work
  END DO
END DO
```

Suppose that $N1 = 17$ and $N2 = 3$. If we chose to direct the compiling system to autotask the loop varying $I1$, and we wanted to use all 8 CPUs of a CRAY Y-MP, then we could potentially suffer a high **LI** overhead when 7 CPUs had to wait for the 8th CPU to finish the 17th iteration. On the other hand, if we direct the compiling system to autotask the loop varying $I2$, then we could only make effective use of 3 CPUs in the parallel region (**LI** overhead again), plus we would suffer a high cost for initiating and terminating the parallel region 17 times (**BPEP** overhead).

If we collapse the two loops into one, the resulting code appears as shown below:

```

DO I12 = 0, N1*N2-1
  I1 = I12 / N2 + 1
  I2 = MOD (I12, N2) + 1
  work
END DO

```

Here, the amount of work to be done by the structure essentially has not changed. The trip count has been increased to $17 * 3 = 51$. If we let C_{work} be the cost of executing the work inside the loop body, then the worst case **LI** cost is $C_{work}/51$, because out of 51 iterations to do, we might have to wait for one iterations' worth of work to be completed; by comparison, in the previous case, if the outer loop is Autotasked, the worst case **LI** cost is $3C_{work} / 17$, because out of 17 outer-loop iterations to do, we might have to wait for 3 inner-loop iterations' worth of work to be completed (each task does an entire DO I2 loop). The difference in **LI** cost is therefore a factor of 9.

Another point worth emphasizing here, on the subject of load balancing, is that the collapsed structure offers more flexibility in terms of specifying work distribution parameters to the Autotasking compiling system [5], thus further increasing the parallel efficiency of the collapsed structure relative to the original.

One of the dangers of collapsing loops is that the scope of the data dependency analysis must be widened to include the bodies of the outer and inner loops. If, for example, we had a structure like the one below,

```

DO I1 = 1, N1
  DO I2 = 1, N2
    TEMP = A(I2-1,I1)
    work
    A(I2,I1) = TEMP
  END DO
END DO

```

Autotasking the loop varying I1 is safe, but Autotasking the loop varying I2 is not, because of the data dependency involving A. This data dependency would persist through the collapse of the loops, rendering the collapsed loop unparallelizable. In a case like this, it might be worthwhile to code the transformed loop so that the value of I1 varies fastest, if the logic within the body of the loop allows this; otherwise, it might be better just to leave the structure alone, since the loop varying I1 can Autotask.

Another potential danger occurs when collapsing a loop structure in which the inner loop has a trip count that is lower than the number of CPUs that could be working concurrently on the collapsed loop. In this case, there might be more than one task working on an iteration of the structure with the same value for the inner loop index. This may or may not be a problem, depending on the contents of the loop body. To illustrate, consider the following collapsed rectangle structure:

```

DO I1 = 1, N1
  DO I2 = 1, N2
    work

```

```

SUM(I2) = SUM(I2) + A(I2,I1)
  END DO
END DO

```

. In this example, if N2 were equal to 4 and the collapsed structure were Autotasked across more than 4 CPUs then more than one task will execute concurrently with the same value for I2. Thus, a race condition on SUM(I2) could occur. Techniques to protect against this danger include installing **GUARD** directives in the loop body or interchanging the loops before collapsing.

3.2 Collapsing the Triangle

Collapsing a loop structure that corresponds to a triangular iteration space is a little more complicated. First we note that the number of cells NC_{tri} in the triangular iteration space of length N1 is given by

$$NC_{tri}(N1) = \sum_{i=1}^{N1} i = \frac{N1 \times (N1 + 1)}{2}$$

So we define a statement function **NC_TRI** which will aid in the readability of the transformed code. Given the following original loop structure:

```

DO I1 = 1, N1
  DO I2 = 1, I1
    work
  END DO
END DO

```

The transformation would then look like this:

```

DO I12 = 0, NC_TRI(N1) - 1
  ISEEK = N1 - 1
  DO WHILE (NC_TRI(ISEEK) .GT. I12)
    ISEEK = ISEEK - 1
  END DO
  I1 = ISEEK + 1
  I2 = I12 - NC_TRI(ISEEK) + 1
  work
END DO

```

The sole purpose of the code at the top of the collapsed loop is to determine, given the collapsed loop index I12, the values of the "original" loop indices I1 and I2. The first impulse of many programmers would be to create counters that get incremented on every iteration of the loop varying I12, and occasionally zero one of the counters when a new strip of the triangle is begun. This kind of logic is probably easier to understand, but in order for it to be run Autotasked, it would have to be **GUARDED** to ensure that only one task at a time updates the counters. The logic shown here makes use of the collapsed loop index I12 and a

private variable ISEEK; its primary advantage is that it requires no such protection as a critical section of the loop.

There are a few noteworthy aspects of this structure. First of all, observe that on the first few iterations of the outer loop in the original structure, the trip count of the inner loop is going to be low. Therefore, you are guaranteed to encounter the situation described above in the discussion of collapsing rectangular structures, namely several tasks running concurrently with the same value for I2. Furthermore, one of the techniques to circumvent this potential problem, that of interchanging the loops, is not an option here, because the inner loop trip count depends on the outer loop index: you can't iterate from 1 to I1 on the outside because you don't know what I1 is yet!

The second thing worth noting is that when the outer loop is Autotasked in the original version of the code, the iterations will contain variable amounts of work (VW); this corresponds to the variation in the height of the triangle at various points along the base. This phenomenon will produce added overhead in this kind of loop. The best way to avoid this situation, if the loops cannot be collapsed, is to arrange the iterations of the outer, Autotasked loop so that the amount of work performed on each iteration decreases as the iteration number increases. This VW problem occurs in essentially all non-rectangular iteration spaces.

3.3 Collapsing the Nevada

For the Nevada-shaped iteration space, it is best to make use of a statement function to compute the indices, much like that used for the triangle space discussed above. When the statement function is used, the collapsed loop looks very much like that for the triangle case. The number of cells in a Nevada structure is:

$$NC_{nev}(N1, N2, N2D) = N1 \times N2 + N2D \times NC_{tri}(N1)$$

The function computes the number of cells in a Nevada-shaped space with a rectangular component of size N1 by N2 and a triangular component whose base is N1 and whose hypotenuse has a slope N2D. The original Nevada iterative structure looks like this:

```
DO I1 = 1, N1
  DO I2 = 1, N2 + I1 * N2D
    work
  END DO
END DO
```

The transformation to collapse the Nevada space to a linear space appears below.

```
DO I12 = 0, NC_NEV(N1, N2, N2D) - 1
  ISEEK = N1 - 1
  DO WHILE ( NC_NEV( ISEEK, N2, N2D ) &
    .GT. I12)
    ISEEK = ISEEK - 1
  END DO
  I1 = ISEEK + 1
  I2 = I12 - NC_NEV( ISEEK, N2, N2D ) + 1
  work
END DO
```

In this case, so long as N2 is large enough, there is no need to worry about the possibility of two concurrent tasks having the same value for their inner loop index I2.

3.4 Collapsing the Histogram

Considering now the histogram iteration space, the same kind of transformation technique can be applied, but first, a special data structure must be built to assist in the collapse. The data structure is an integer array of extent (0:N1), where N1 is equal to the number of bars in the histogram, and the value of each element of the array is equal to the height of the histogram bar corresponding to the element's index, plus the value of the preceding element (the value of the zero-th element is zero). Hence, the data structure represents a running sum of the number of cells in the histogram up to a certain point:

$$NC_{hist}(N1) = \sum_{i=1}^{N1} N2(i)$$

We can call this data structure NC_HIST. The code to construct the NC_HIST data structure, prior to entering the loop structure, looks like this:

```
NC_HIST(0) = 0
DO I1 = 1, N1
  NC_HIST(I1) = NC_HIST(I1-1) + N2(I1)
END DO
```

Equipped with this data structure, we can now collapse the histogram iterative structure. The original histogram loop structure looks like this:

```
DO I1 = 1, N1
  DO I2 = 1, N2(I1)
    work
  END DO
END DO
```

The collapsed histogram loop structure is as shown below:

```

DO I12 = 0, NC_HIST(N1) - 1
  ISEEK = N1 - 1
  DO WHILE (NC_HIST(ISEEK) .GT. I12)
    ISEEK = ISEEK - 1
  END DO
  I1 = ISEEK + 1
  I2 = I12 - NC_HIST(ISEEK) + 1
  work
END DO

```

As with the triangle iteration space, we will see here the potential for more than one task running at a time with the same value for I2. And as is also the case for the triangle, the loop interchange remedy is not available to us because of the dependency between the loops. But it is worth remembering that this potential problem is only a real problem whenever there is code within the body of the loop that updates some data structure element using I2 as an index. In this case, the only good remedy is to create a critical section in the loop body that prevents I2-indexed data structures from being updated by one task while being referenced by another task.

3.5 Collapsing the Porous Shape

The last iterative structure we will consider in this section is that of the porous shape, representing a structure with conditional work inside. In this scenario, what we would like to do is eliminate VW overhead. We are also interested in eliminating the iteration overhead of distributing to the CPUs iterations that essentially have no work in them. The technique involves essentially skipping the iterations for which the porosity condition holds true, and distribute only those iterations for which real work will be done. To accomplish this, we must build a data structure before entering the parallel region, much like that used for the histogram above:

```

NCONDA = 0
DO I1 = 1, N1
  DO I2 = 1, N2
    IF (CONDA (I1, I2)) THEN
      NCONDA = NCONDA + 1
      ICONDA(1,NCONDA) = I1
      ICONDA(2,NCONDA) = I2
    END IF
  END DO
END DO

```

CONDA is a function that takes as arguments the loop structure indices, and returns a logical result; it is essentially the porosity function. The data structure ICONDA is an integer array; its size in the first dimension must be equal to the nesting depth of the iterative structure; here it is 2. The size of ICONDA in the second dimension must be large enough to represent the completely non-porous iterative structure; in this case, NCONDA could get as large as N1 * N2, so ICONDA must be equipped to store that many entries. (Of course, if the programmer knows a priori what

the degree of porosity of the structure is likely to be, he can dimension ICONDA accordingly). The ICONDA array keeps track of those non-porous cells within the structure where there is work to be done. Its use in the transformation of the porous iterative structure is shown below. First, the original porous loops:

```

DO I1 = 1, N1
  DO I2 = 1, N2
    IF (CONDA (I1, I2)) THEN
      work
    END IF
  END DO
END DO

```

Now, the collapsed porous structure:

```

DO I12 = 1, NCONDA
  I1 = ICONDA(1,I12)
  I2 = ICONDA(2,I12)
  work
END DO

```

Note that, unlike the other loop collapse transformations, this collapsed loop iterates fewer times than the original structure, so the possibility exists that the collapsed loop will not have a high enough trip count to warrant Autotasking. The effects of this condition, and techniques for accounting for it, will be covered in the section on testing.

It is important to keep in mind that the shape of a structure and its porosity are two totally independent characteristics. In fact, the technique for collapsing a porous structure can be applied to any kind of loop nest, regardless of its shape. This makes the porous collapse technique the most general of all the techniques.

4.0 Performance Testing of Collapsed Loops

For each of the five collapse techniques discussed in the previous sections (rectangle, triangle, Nevada, histogram, porous), four test runs were performed. Two of the four test runs compare wall clock times, and two compare CPU times. Of the two wall clock tests, one compares the performance of the collapsed loop against the original with the inner loop Autotasked, and the other compares the performance of the collapsed loop against the original with the outer loop Autotasked. The same two comparisons are done in the two CPU time tests. The test programs were compiled and executed on a 16-CPU Cray Y-MP C90 installed in the Corporate Computing Network at Cray Research, Inc. in Eagan, MN. All tests were executed during SJS time, which is essentially equivalent to dedicated time. CPU timing tests used the SECOND function, and wall clock timing tests used the RTC intrinsic [6].

The CPU timing plots should reveal gains achieved by collapsing, specifically in the area of reducing BPEP overhead; they may also show costs associated with collapsing, specifically

in the areas of executing code to generate supporting data structures or to compute original index values. The wall clock timing plots should reveal gains achieved by collapsing, specifically in the area of reducing **LI** and **VW** overhead.

The bodies of the loops in all cases were the same, essentially:

```
CALL WORK (A(I2,I1))
```

where A is an appropriately-dimensioned array and the subroutine WORK looks like this:

```
SUBROUTINE WORK (A)
A = 2.7181
DO I = 1, 512
  A = EXP (LOG (A))
END DO
END
```

Thus, the WORK routine does nothing useful, but does exercise the hardware and accumulate CPU time. Notice also that memory traffic is minimal in the loop body. This makes the test results essentially unbiased by memory contention issues. In a real loop body, however, memory contention could be a very big issue.

The results of these tests are depicted in the 20 plots that make up the Appendix. Each will be discussed in the sections that follow.

4.1 Speedup from Collapsing the Rectangle

The graphs that describe the results of the rectangle tests are all 3-D mesh plots, with speedup shown on the vertical axis, as a function of the rectangle's length and width. In these tests, the outer loop of the original structure iterates over the width of the rectangle, and its trip count varies from 1 to 30; the inner loop iterates over the length, and its trip count is varied from 1 to 50.

4.1.1 Inner/CPU

In this plot, we see that the speedup of the collapsed structure is high where the length of the rectangle is small, and decreases to an asymptotic value around 1.0 as length increases. In the original version of this test code, the length dimension of the shape is being processed in the inner loop, and the inner loop is the one being Autotasked, so a small value for length means a high ratio of **BPEP** code execution to user code execution.

The plot also shows that the collapsed structure performs slightly better than the original when the width of the shape is large. This can be explained by the fact that an increase in width means, for the original code, increasing **BPEP** overhead. This overhead cost is not present in the collapsed structure.

4.1.2 Inner/Wall

This plot has the same general shape as the previous

Inner/CPU plot, but the scales are much different. In this case, the speedup to be obtained by collapsing the loops is quite dramatic if the shape is short and wide. This speedup is due to improved load balancing.

Notice also that this plot is more "bumpy" than the previous plot. This is presumably due to the fact that the **LI** overhead of the original code is highly dependent on the trip count of the Autotasked loop; further, there is always an inherent slight variability in wall-clock timings.

4.1.3 Outer/CPU

This plot shows that collapsing the rectangle yields essentially no benefit in CPU time over Autotasking the outer loop.

4.1.4 Outer/Wall

In this plot, we see only slight speedups for the collapsed code over the original. There appears to be a significant drop in speedup at width = 16. This is probably because the original code performs most efficiently there, since the trip count is exactly equal to the number of CPUs in the machine.

4.2 Speedup from Collapsing the Triangle

The graphs that describe the results of the triangle tests are all 2-D line graphs, with speedup shown on the vertical axis, as a function of the triangle's base and height. In these tests, base and height were always equal, and they were allowed to vary from 1 to 50.

4.2.1 Inner/CPU

This graph shows speedup to be moderate when the size of the triangle is small, and decreasing as the triangle increases in size.

4.2.2 Inner/Wall

Speedups for this case are quite significant, as shown in this graph. Like the Inner/CPU case above, payoffs are highest for the small shape, and tapering off as the size of the shape increases. This graph is quite jagged, perhaps indicating that the **LI** overhead is, in the original version of this test case, highly dependent upon small changes in the size of the problem.

4.2.3 Outer/CPU

In this case, the speedup is negligible over essentially the entire test space. Notice the scale of the vertical axis.

4.2.4 Outer/Wall

Speedups here are significant for small and moderate-sized problems. The spikes in the graph are evidence of **LI** over-

head in the original version of the code, which are exacerbated by **VW** conditions.

4.3 Speedup from Collapsing the Nevada

The plots that describe the results of the Nevada tests are all 3-D mesh plots, with speedup shown on the vertical axis, as a function of the shape's rectangular dimensions ($N1$ and $N2$) and diagonal slope ($N2D$). In these tests, $N1$ and $N2$ were always equal, and they were allowed to vary from 1 to 50. $N2D$ was allowed to vary from 1 to 20. The test program iterates over the length of the shape in the outer loop, and over the (variable) width in the inner loop.

4.3.1 Inner/CPU

For this case, the speedup obtained by collapsing is negligible.

4.3.2 Inner/Wall

Here, the speedups are moderate (around 1.5) over most of the test space. The plot shows that speedups are quite variable when the iteration space is small, and more steady when the shape is large. The variation in the performance of the small test cases may be due to differences in task scheduling.

4.3.3 Outer/CPU

This is an interesting plot. Speedups take on a stair-step behavior based on the length of the iteration space. The steps occur at length values that are multiples of 16, the number of CPUs in the machine being used. In this test, the outer loop, iterating over the length of the shape, is Autotasked. Presumably, when an Autotasked loop has a trip count that is some integer multiple of the number of CPUs in the machine, the iterations will be distributed evenly and the efficiency will be near optimal. The plot seems to support this hypothesis.

4.3.4 Outer/Wall

Speedups are dramatic in this case, in the region where the rectangular component of the shape (given by $N1$ and $N2$) is small and the slope of the hypotenuse of the triangular component (given by $N2D$) is steep. The Autotasked loop in the original code iterates over $N1$, so when $N1$ is small, the ratio of overhead to useful work is high. Further, when $N2D$ is large, that means that the trip count of the inner loop varies greatly with respect to $N1$; this results in a very high amount of **VW** overhead. As in the Outer/CPU plot discussed above, there is a stair-step effect at length values of 16, 32, and 48; however, these effects are very subtle here.

4.4 Speedup from Collapsing the Histogram

The plots that describe the results of the histogram tests are all 3-D mesh plots, with speedup shown on the vertical axis,

as a function of the length (number of bars) and the variation in bar height. The average bar height is 25, and the bar height range is allowed to vary from 2 to 50. Before a histogram is processed, the length and average bar height are chosen, then the bars are assigned random heights within the allowable range. The efficiency with which the original coding structure can process the histogram space should depend strongly on the variance in the histogram's bar heights. It is important to note that even though this series of tests used random numbers to generate the iteration spaces, the same series of shapes was generated in each test run, because each test started with the same (default) random number seed.

4.4.1 Inner/CPU

Speedups in this case are negligible. Any gains made in reducing the **BPEP** overhead are perhaps being offset by the cost of constructing the `NC_HIST` data structure.

4.4.2 Inner/Wall

This plot shows the wall-clock speedups to gather around the 1.5 value, but to vary between 1.0 and 2.0. The variation seems to be independent of both the length of the histogram and the variation in bar height. The random jaggedness of the plot is most likely due to the randomness of the shape of the iteration space itself. It is interesting to note, however, that the variation seems to be greater in the left rear corner of the plot and smaller in the right front corner.

4.4.3 Outer/CPU

This plot shows that very little is gained in terms of CPU time from collapsing the histogram versus simply Autotasking the outer loop.

4.4.4 Outer/Wall

Speedups in wall-clock time from collapsing the histogram can be quite dramatic, especially where the length of the shape is small and the bar height variation is high. In this test, the outer loop iterates along the length of the shape. Since each bar being processed in an iteration of the outer loop has a random height, the potential for **VW** overhead is very high; and the larger the variation, the higher the overhead. Then there is the **LI** overhead that can arise when parallelizing in the large grain, especially when the length of the histogram is low. These factors together can severely impact the parallel efficiency of the original structure.

4.5 Speedup from Collapsing the Porous Rectangle

The plots that describe the results of the porous rectangle tests are all 3-D mesh plots, with speedup shown on the vertical axis, as a function of the size of the rectangle and the degree of

porosity. The rectangle is actually a square, because length and width are always equal; they are allowed to vary from 1 to 50. The porosity of the shape is varied from 0% to 95%, in intervals of 5%. The “holes” in each shape are placed at random before the processing of the shape; as in the histogram tests described above, the series of shapes that are used in each test are all the same. The porosity of the shape should have a direct impact on the original coding structure’s ability to process the shape efficiently.

4.5.1 Inner/CPU

Speedups in this test are significant only in the cases where the size of the rectangle is small, or when the porosity is very high. The higher the porosity, the more frequently we are executing concurrent iterations for no useful purpose. The collapsed structure has been designed in such a way that this overhead does not occur.

4.5.2 Inner/Wall

As in the Inner/CPU case described above, speedups are greatest where length is small or porosity is high. Since iterations of the inner loop will do either much work or no work, the VW overhead of high-porosity structures is extreme.

4.5.3 Outer/CPU

This plot shows that speedups are negligible in all but the most pathological of cases. The BPEP overhead saved by collapsing is offset by the cost of building the ICONDA data structure.

4.5.4 Outer/Wall

For this test, the speedups were modest across the majority of the test space. The outer loop in the original code iterates across the length of the structure, and each iteration processes one row along the width. Although these rows are porous, they are all equally porous, so the load here is fairly well balanced. The quality of uniform porosity across the rows of the structure is an artifact of the way in which the test was constructed; it should not be presumed to be a quality of all porous structures. In general, it is probably reasonable to assume that the less uniform the porosity, the better the speedup from collapsing will be.

5.0 Conclusions and Future Work

The loop collapse technique can be applied to a wide variety of iterative structures, with performance benefits that range from modest (25%) to remarkable (over 500%). A collapsed loop offers the advantages of small-grain parallelism, such as good load balance, with the advantages of large-grain parallelism, such as low parallel-startup costs. In most cases the code transformations are trivial; the type of transformation needed is governed by the geometric characterization of the iterative structure.

Several common geometries were presented, corresponding transformation strategies developed, and performance characteristics were measured.

There are several areas that remain to be explored, including:

- a. More work needs to be done to characterize loops with varying amounts of work, and the performance impact of splitting these loops into separate structures that each contain fixed body sizes;
- b. It would be interesting to study the degree to which performance is impacted by body size in structures that have low trip counts;
- c. The memory efficiency of collapsed loops versus their un-collapsed counterparts should be studied;
- d. Some thought should be given to the feasibility of making these kinds of collapse transformations automatically, within the compiler;
- e. It is not clear how transformations such as these fit into the MPP Fortran Programming Model [7]. It is possible that a DOSHARED directive with multiple induction variables specified can do the same job as, or perhaps a better job than, loop-collapse transformations on some kinds of loops.

6.0 Acknowledgements

I would like to thank the following people for providing assistance and inspiration for this work:

Peter Feibelman, Distinguished Member of Technical Staff, Surface and Interface Science Department 1114, Sandia National Laboratories. I developed the concepts of collapsing loops of different shapes while working with Peter on optimizing and multitasking his impurity codes;

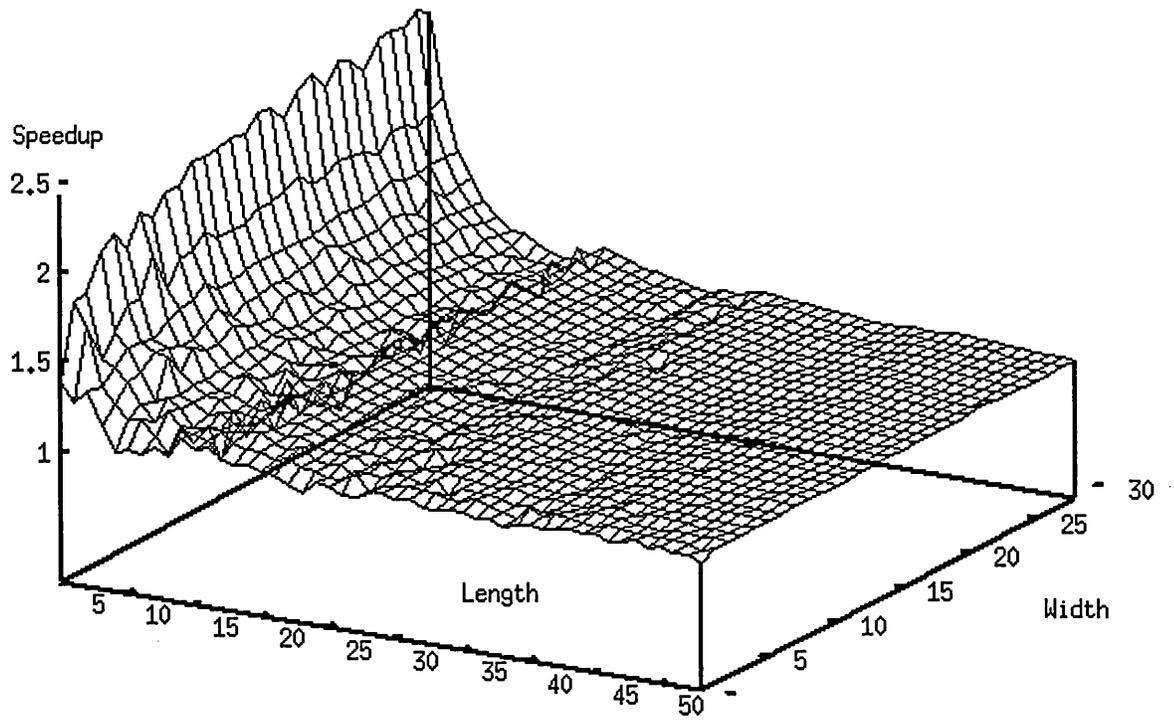
Al Iacoletti, John Noe, Rupe Byers, and Sue Goudy, Scientific Computing Directorate 1900, Sandia National Laboratories. The comments and suggestions from these reviewers helped improve the clarity and structure of several important parts of the paper. All remaining errors, omissions, inaccuracies, inconsistencies, etc., are, of course, my responsibility.

REFERENCES:

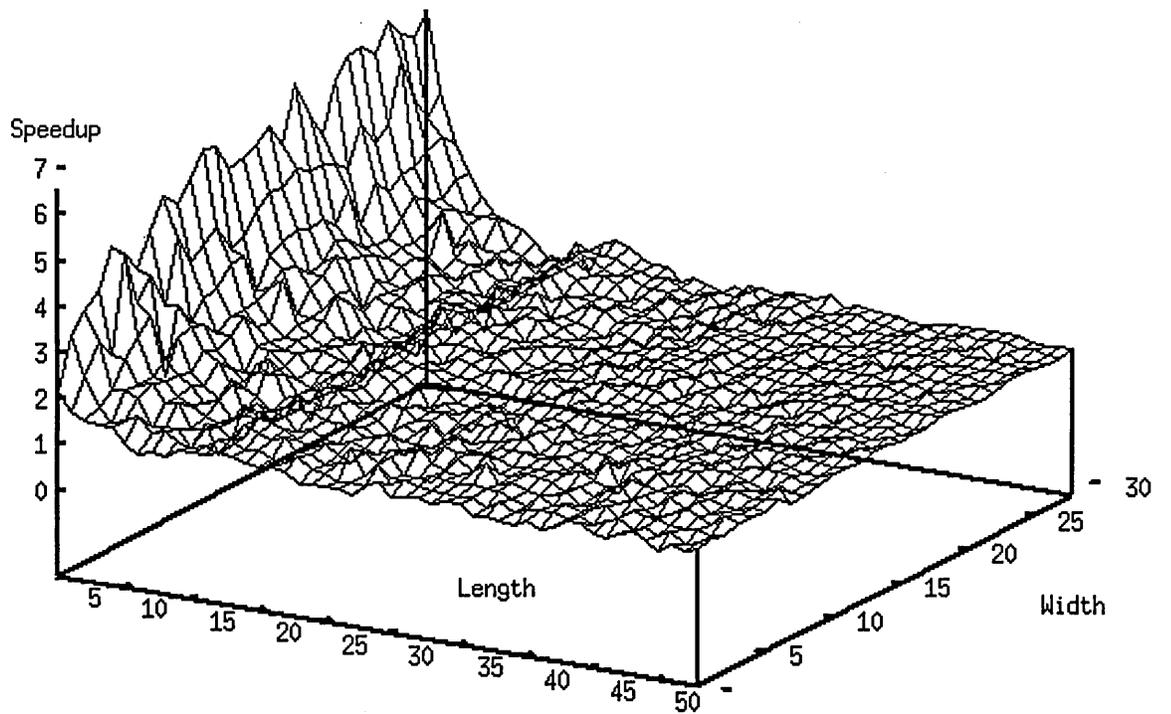
1. Cray Research, Inc., *CF77 Optimization Guide*, SG 3773 6.0, p. 179.
2. Ibid., p. 185.
3. Ibid, pp. 86-87.
4. Levesque J., and Williamson, J. *A Guidebook to Fortran on Supercomputers*, Academic Press (1989), p. 91.

5. Cray Research, Inc., *CF77 Commands and Directives*, SR-3771 6.0, p. 79.
6. Cray Research, Inc., *UNICOS Fortran Library Reference Manual*, SR-2079 7.0, pp. 289-291.
7. Cray Research Inc., *Programming the Cray T3D with Fortran*, SG-2509 Draft, pp. 45-48.

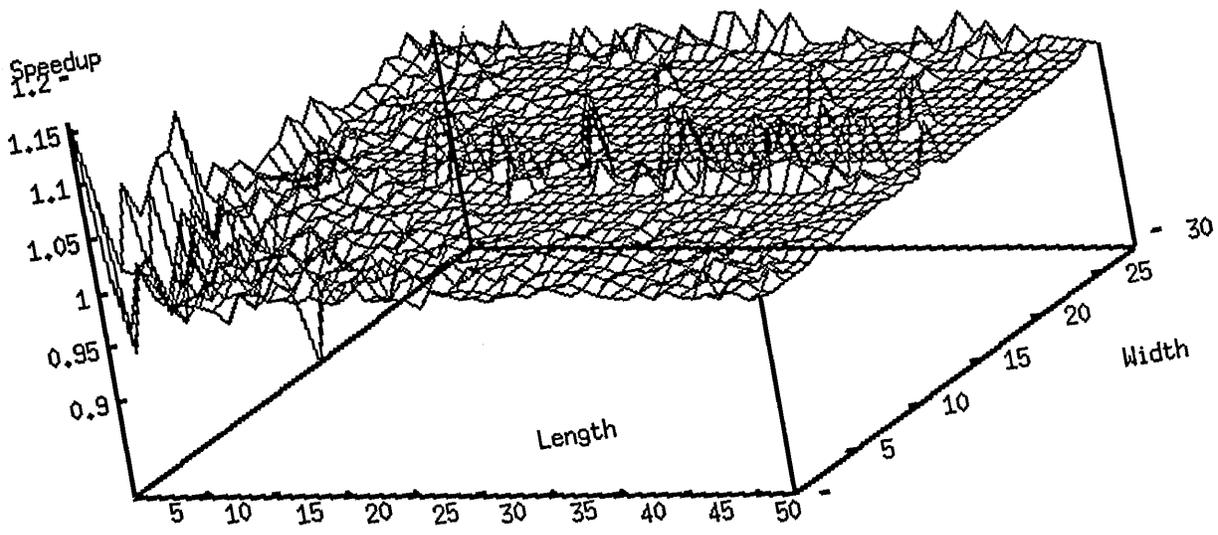
Speedup from collapsing rectangle (inner/cpu)



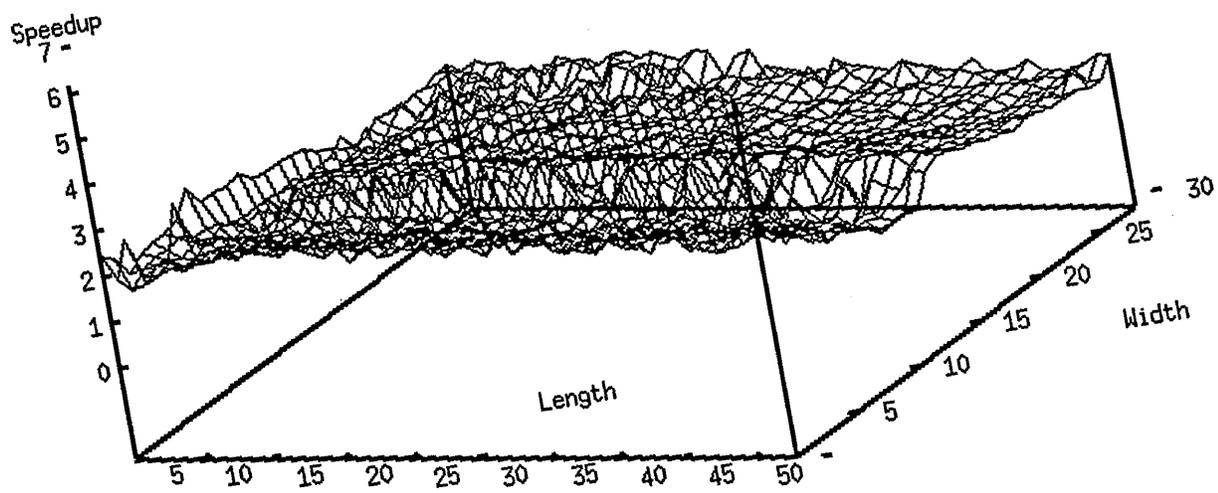
Speedup from collapsing rectangle (inner/wall)

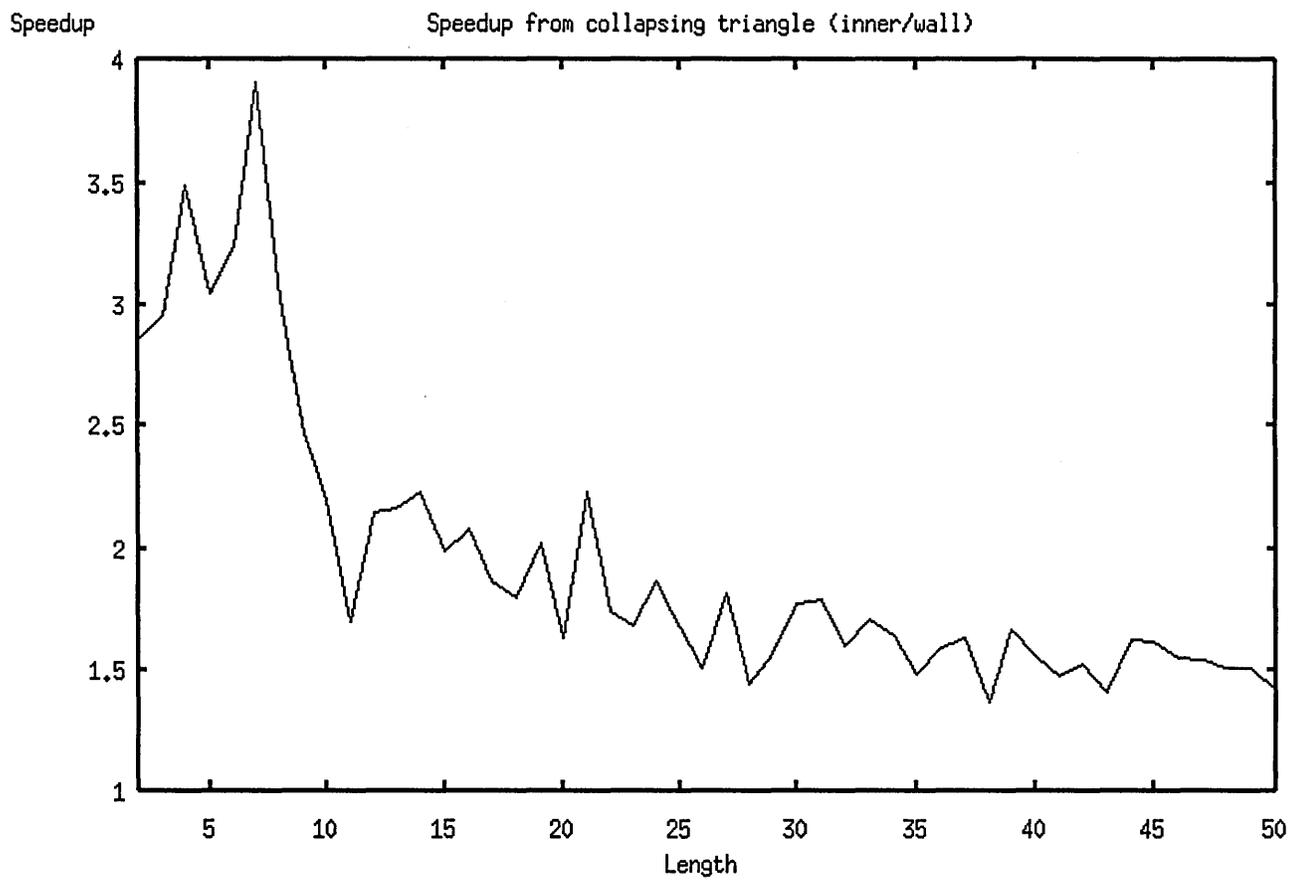
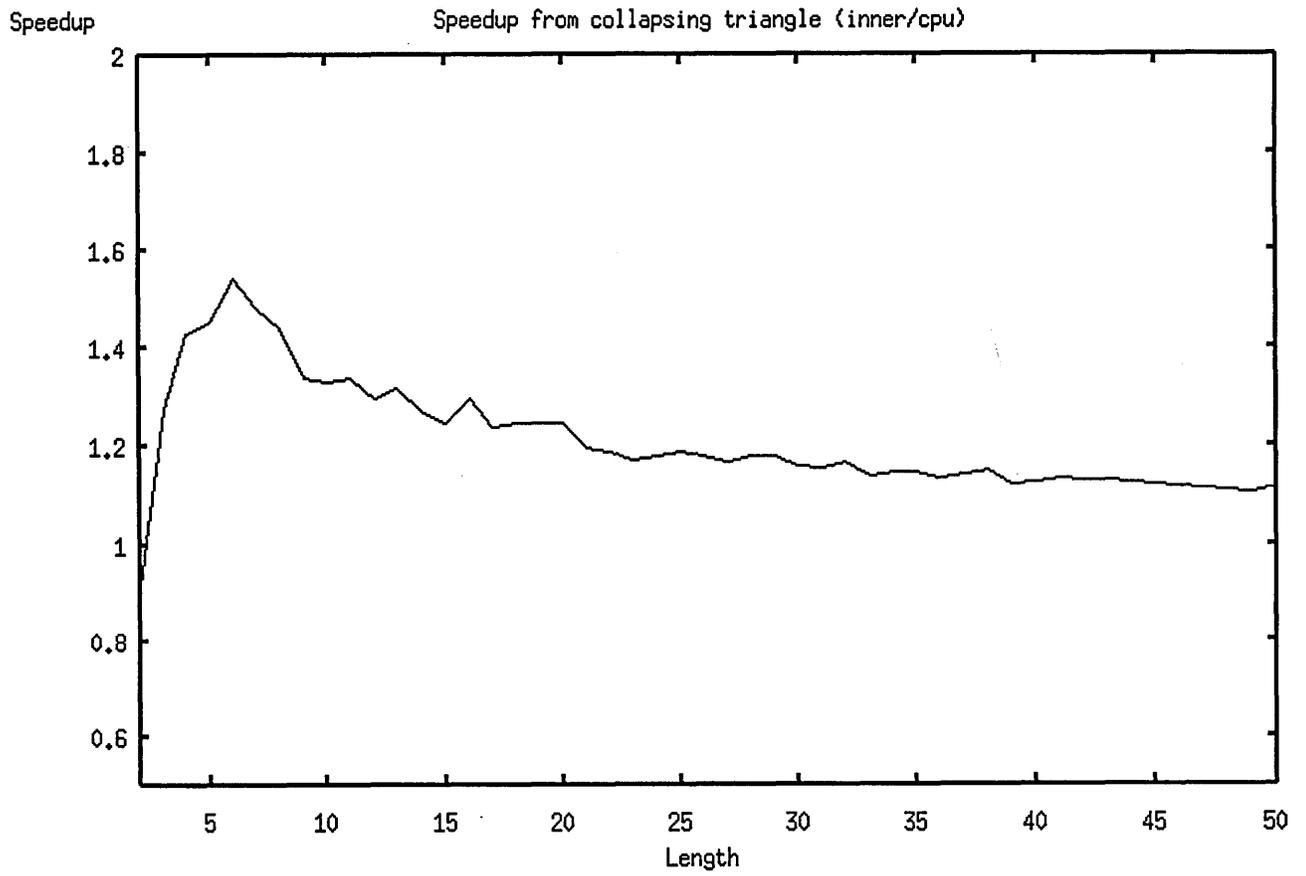


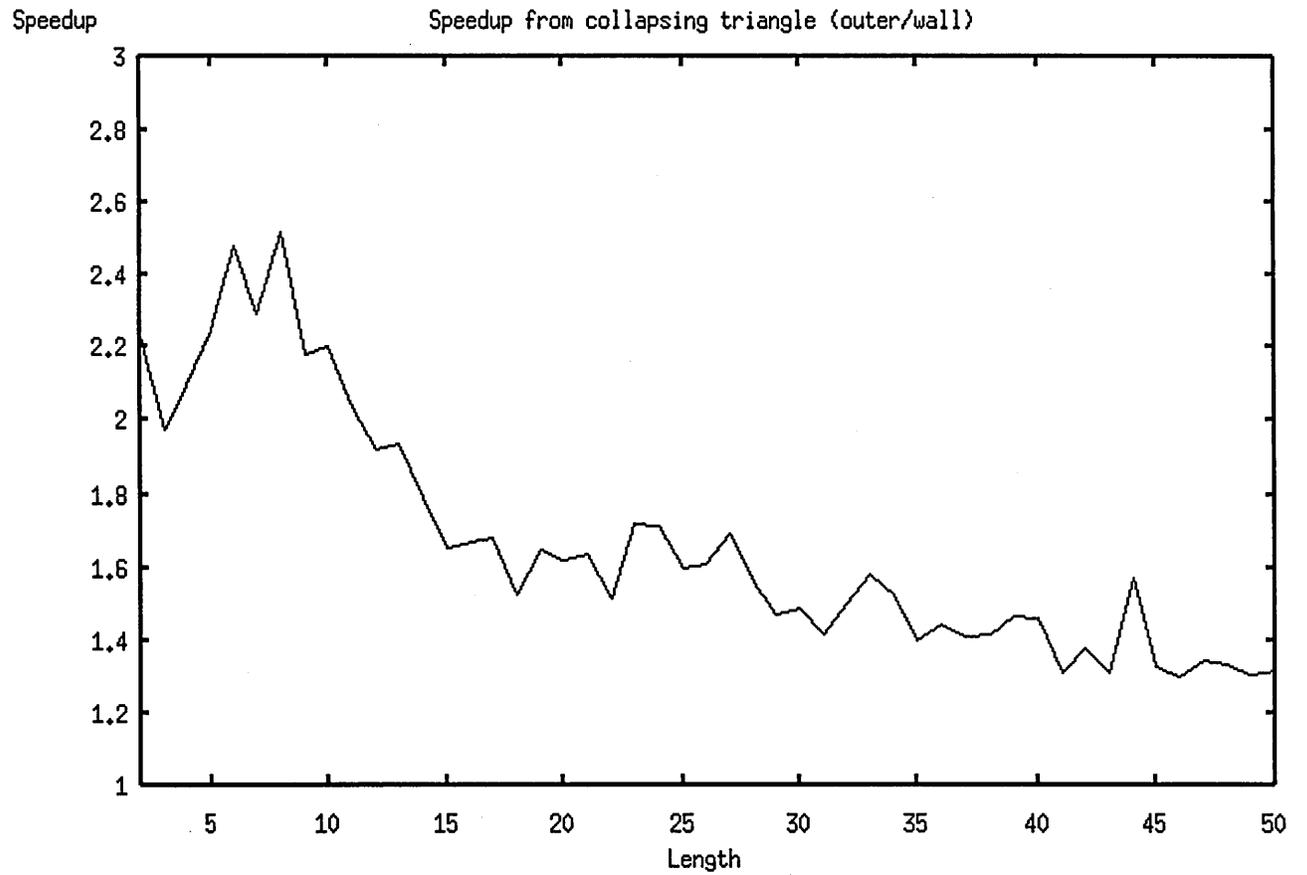
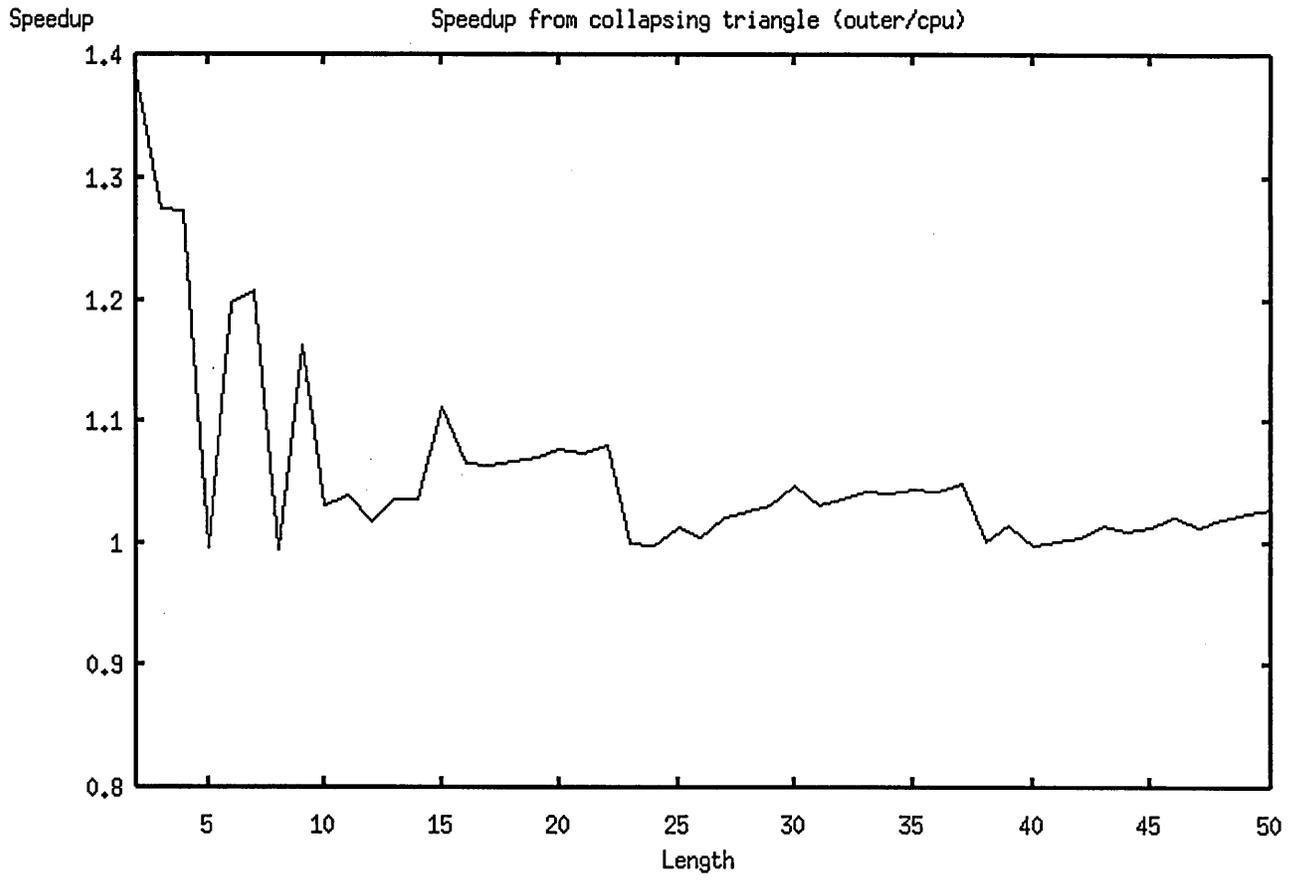
Speedup from collapsing rectangle (outer/cpu)



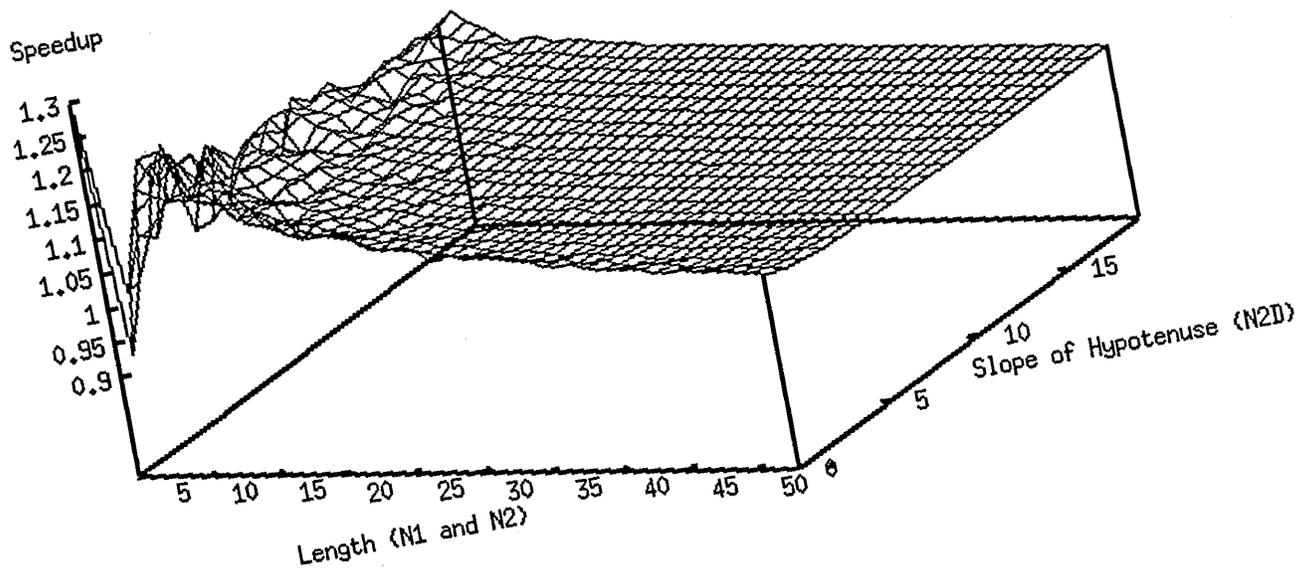
Speedup from collapsing rectangle (outer/wall)



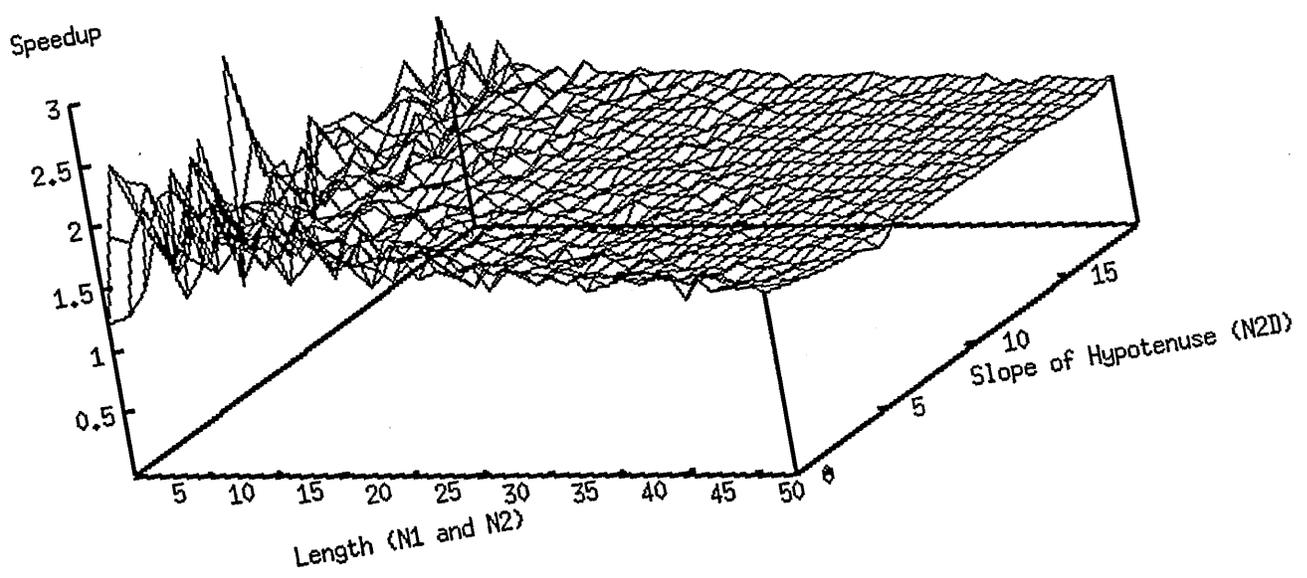




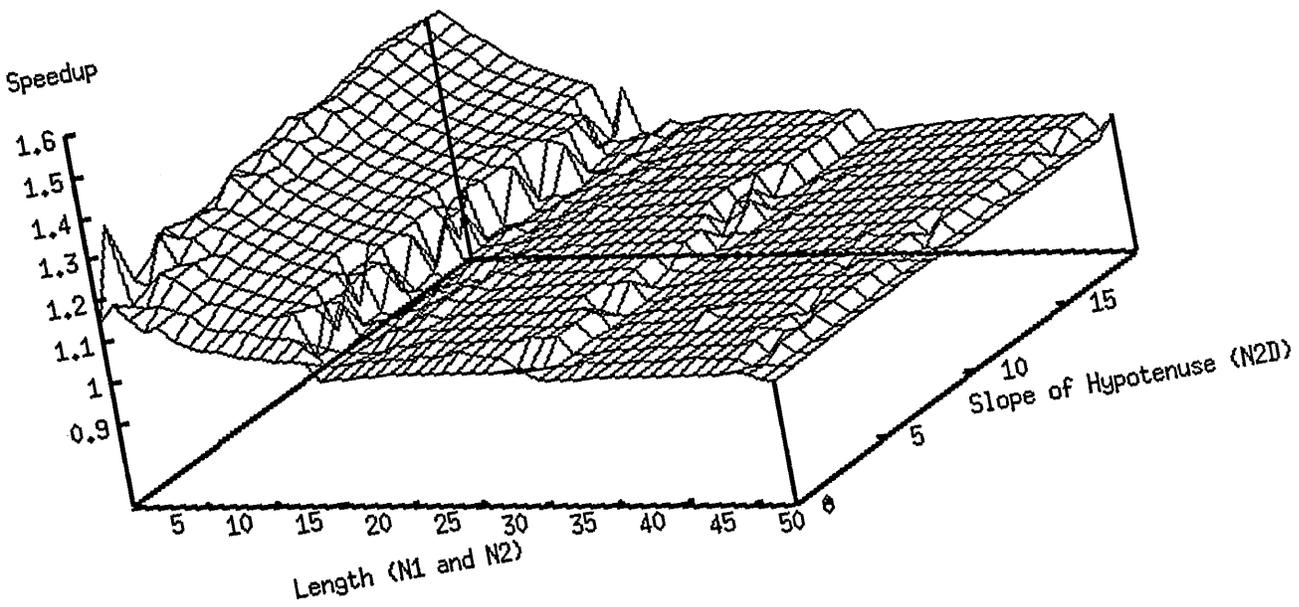
Speedup from collapsing Nevada (inner/cpu)



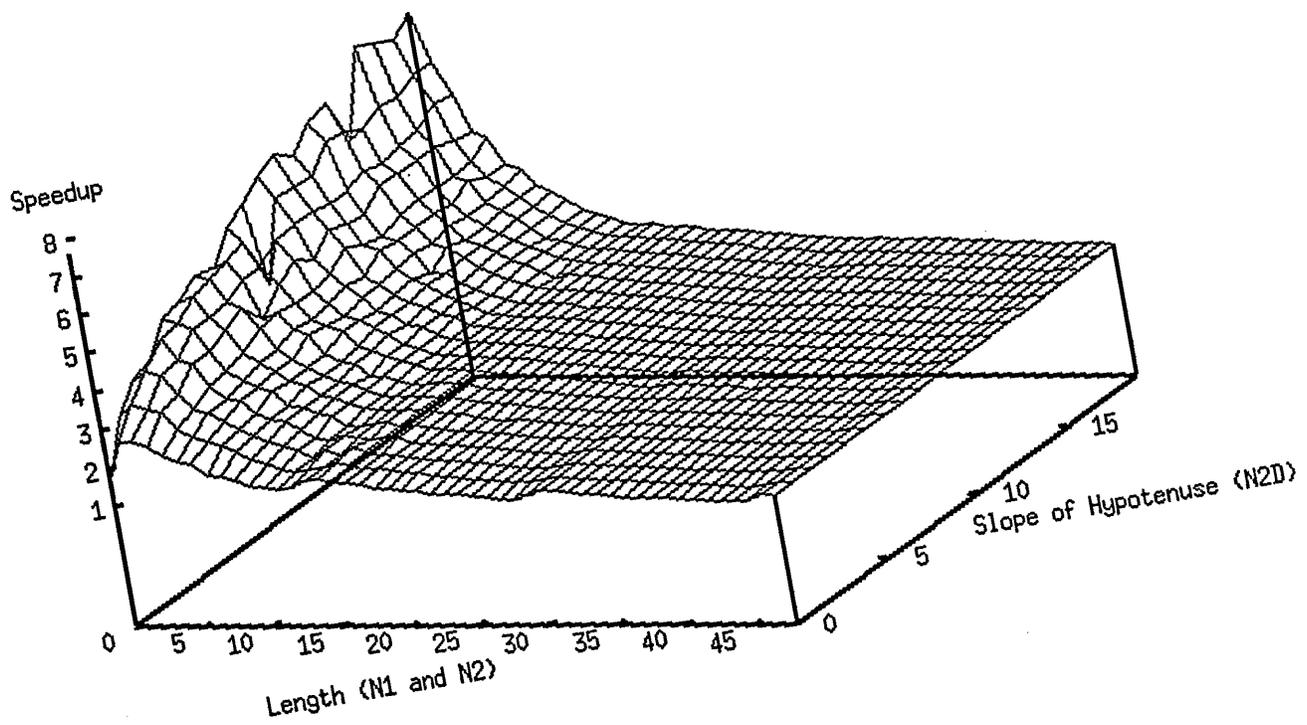
Speedup from collapsing Nevada (inner/wall)



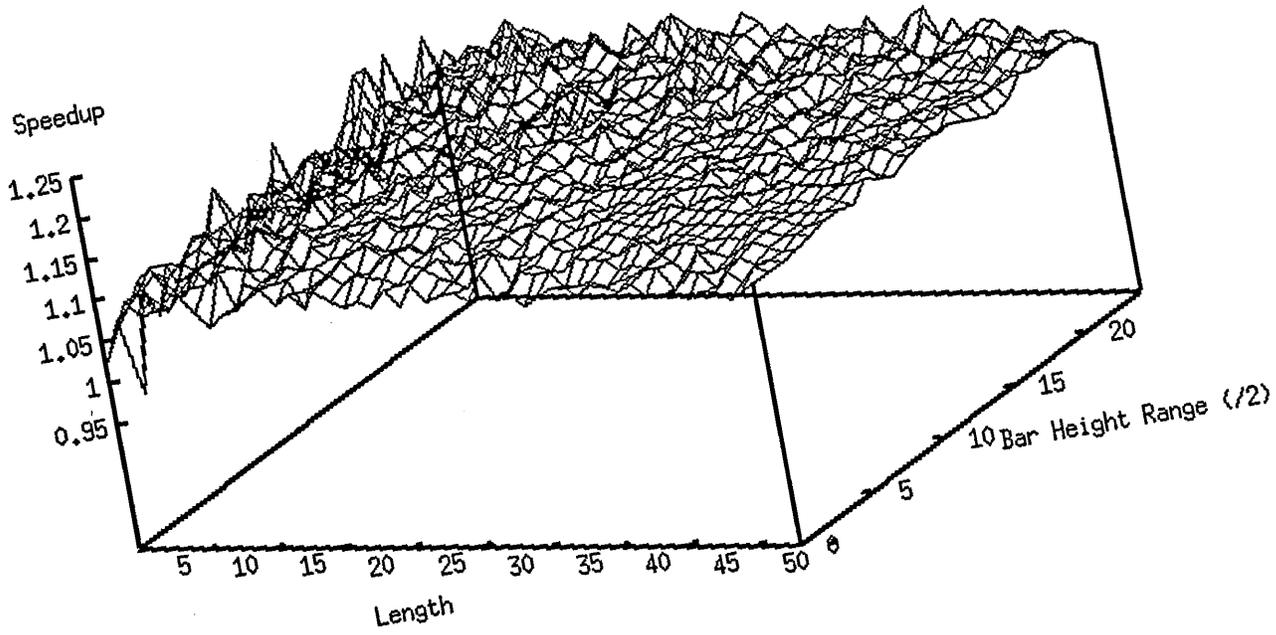
Speedup from collapsing Nevada (outer/cpu)



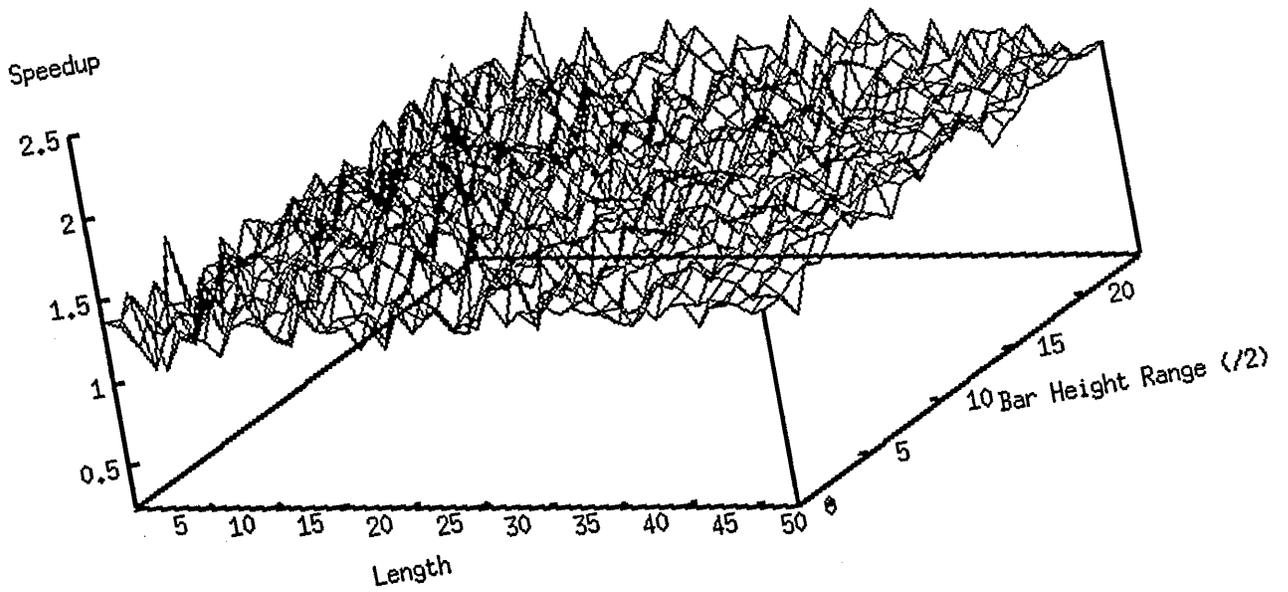
Speedup from collapsing Nevada (outer/wall)



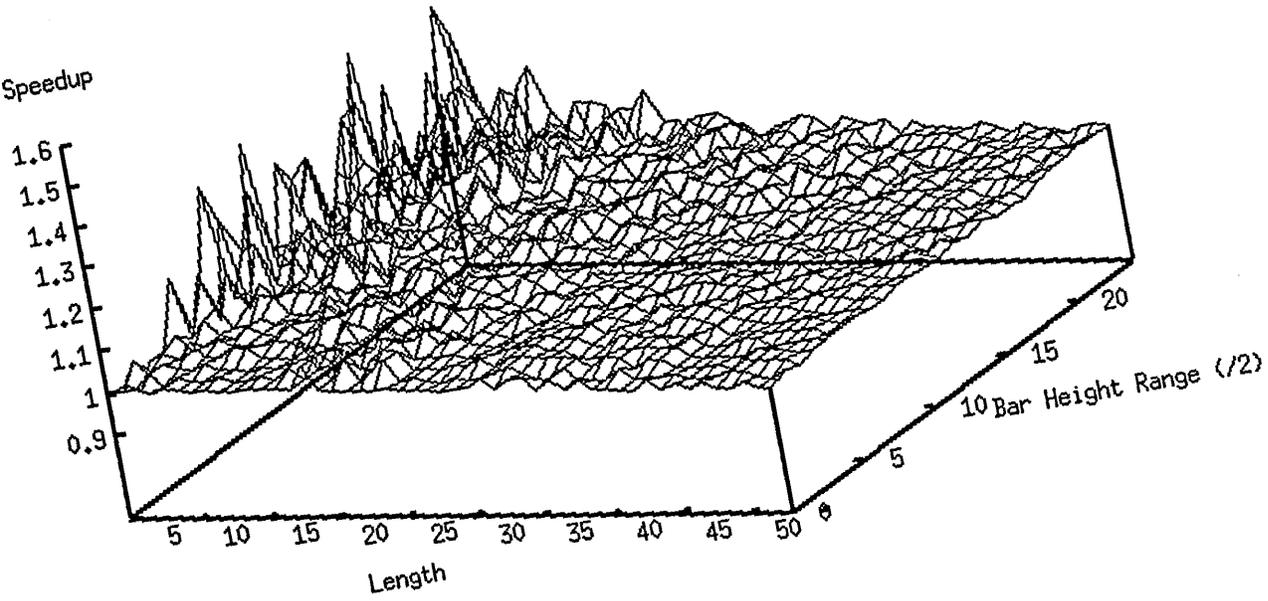
Speedup from collapsing histogram (inner/cpu)



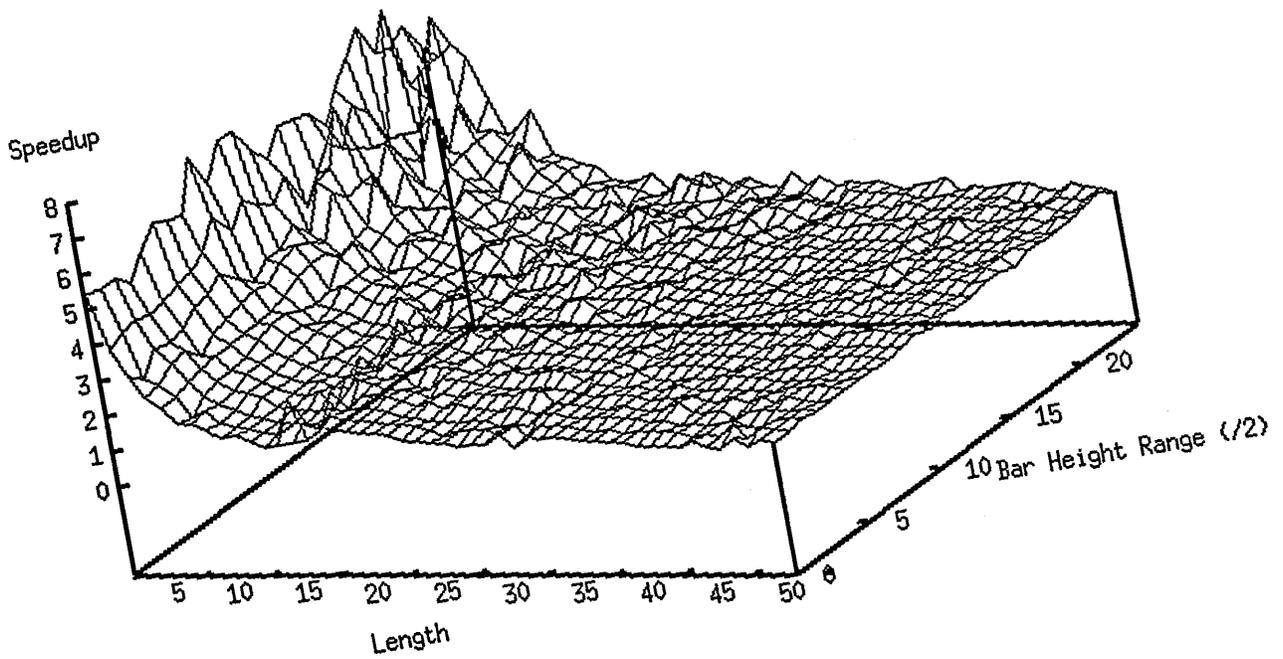
Speedup from collapsing histogram (inner/wall)



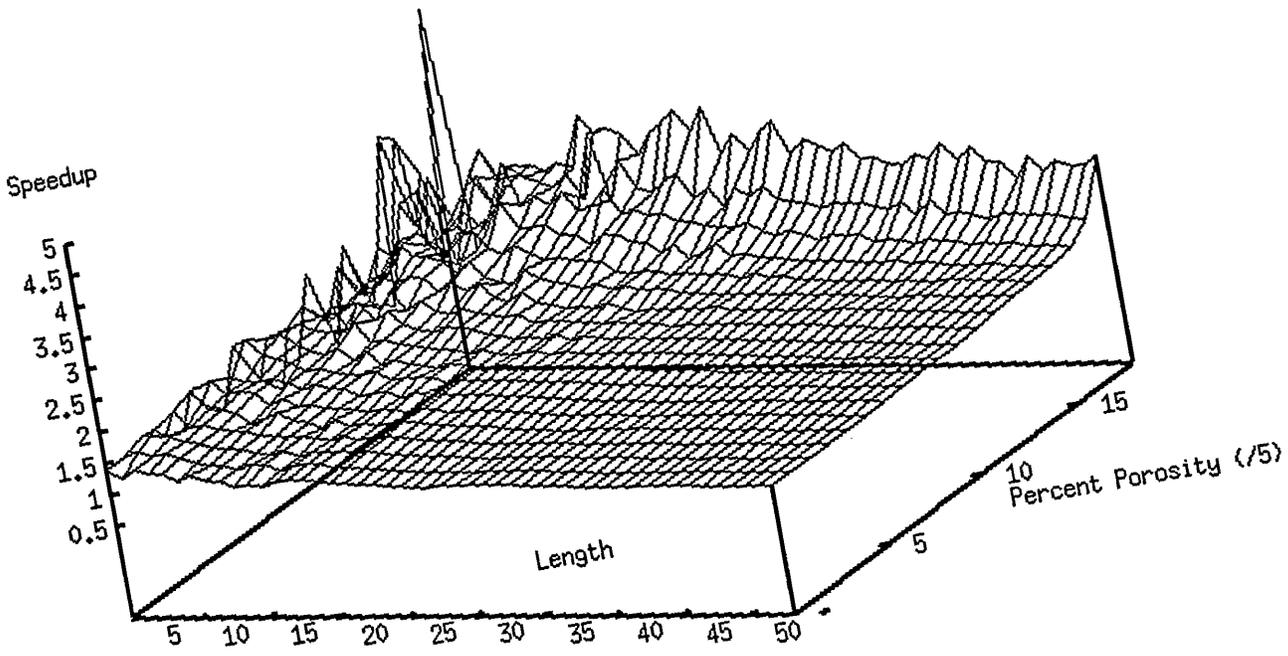
Speedup from collapsing histogram (outer/cpu)



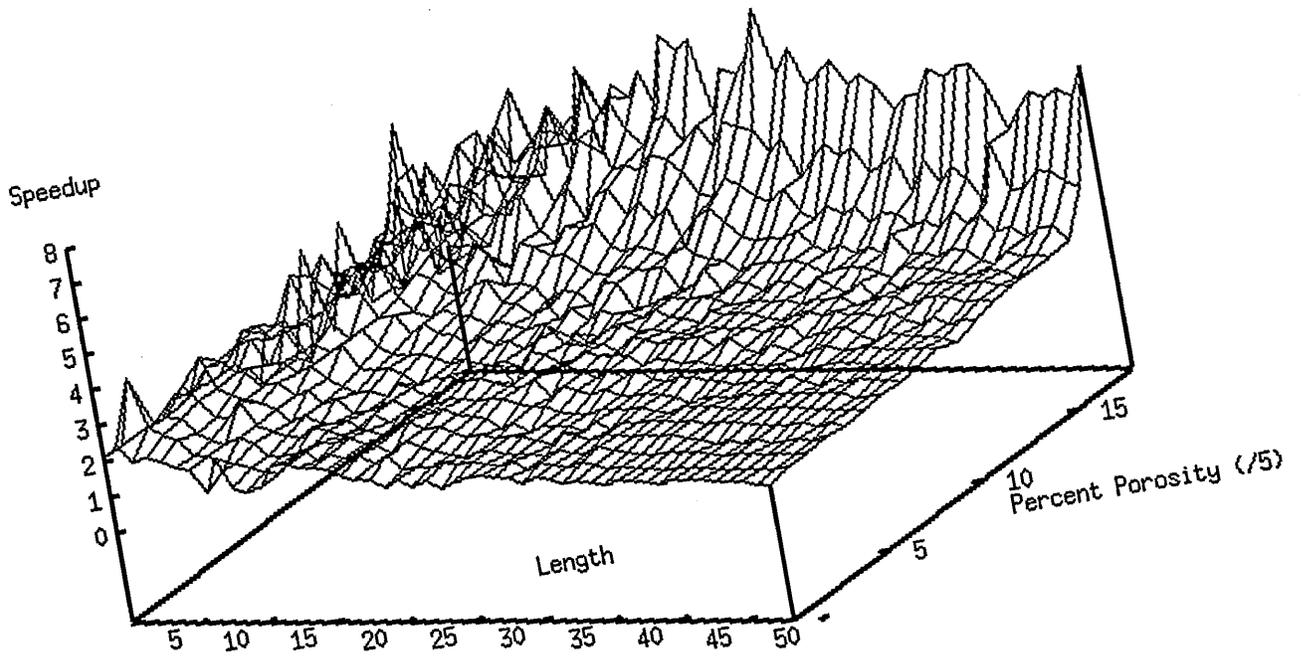
Speedup from collapsing histogram (outer/wall)



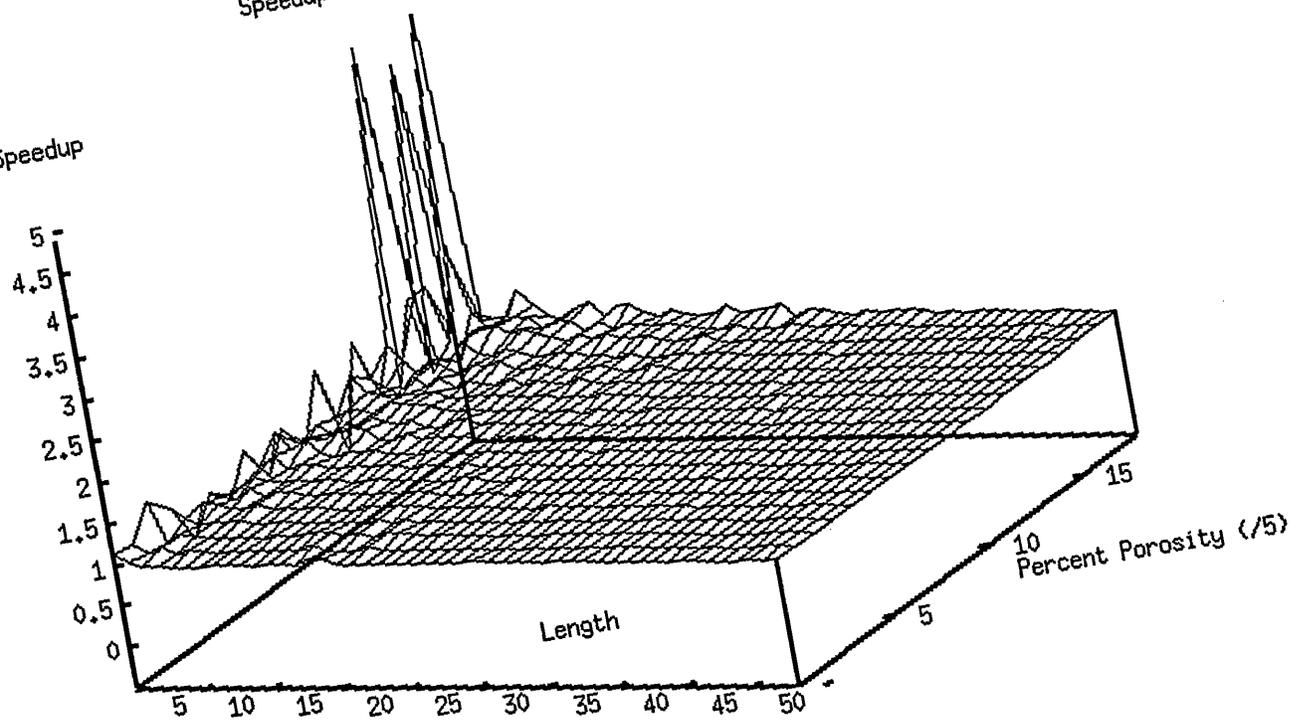
Speedup from collapsing porous rectangle (inner/cpu)



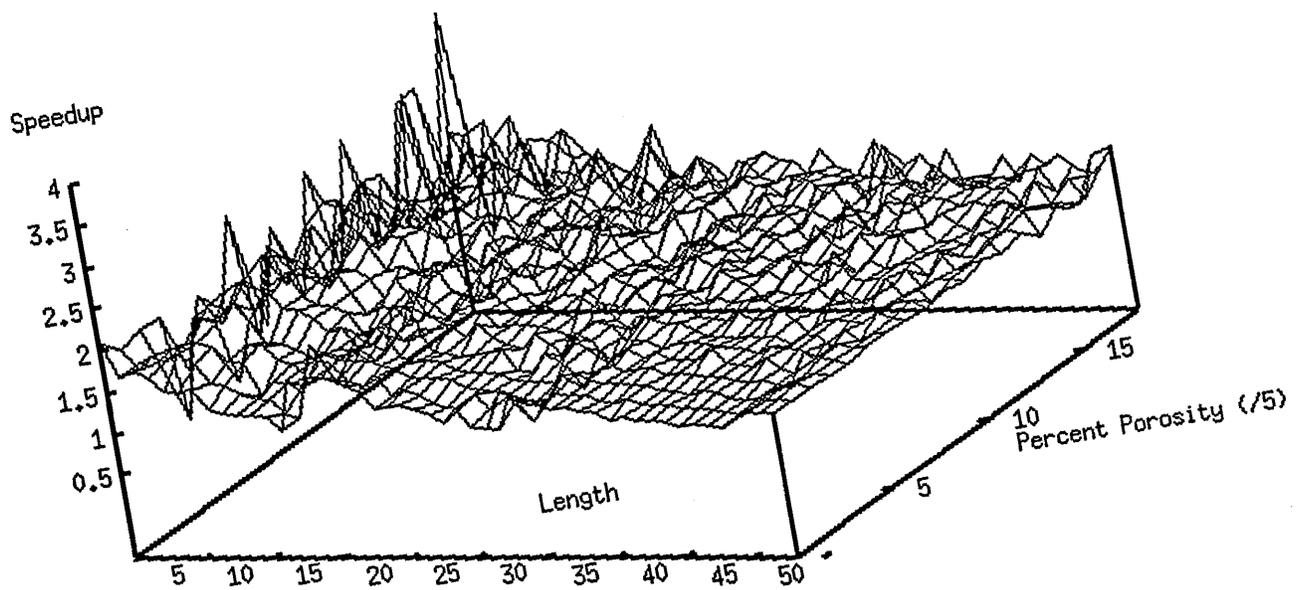
Speedup from collapsing porous rectangle (inner/wall)



Speedup from collapsing porous rectangle (outer/cpu)



Speedup from collapsing porous rectangle (outer/wall)



Collaborative Evaluation Project Of Ingres On The Cray (CEPIC)

C. B. Hale, G. M. Hale, and K. F. Witte

Los Alamos National Laboratory
Los Alamos, NM

Abstract

At the Los Alamos National Laboratory, an evaluation project has been done that explores the utility a commercial database management system (DBMS) has for supercomputer-based traditional and scientific applications. This project studied application performance, DBMS porting difficulty, and the functionality of the supercomputer DBMS relative to its more well-known workstation and minicomputer hosts. Results indicate that the use of a commercial DBMS package with a scientific application increases the efficiency of the scientist and the utility of the application to its broader scientific community.

Introduction

Ingres is a relational distributed database management system that makes it possible to quickly transform data into useful information in a secure environment using powerful database access and development tools. It has the architecture for true client/server performance.

Ingres offered Los Alamos a ninety day trial evaluation of the Cray/Ingres product on a Los Alamos Cray. This included the use of the base product, ABF, C, FORTRAN, Knowledge Management, TCP/IP Interface, Net, Vision, and STAR. To insure the success of the project, they provided Los Alamos with the Premium level of support (required for a Cray platform) during the trial period.

CRI also was very supportive of Los Alamos' interest in putting Ingres on Crays, because they see the potential for entering a new market. Sara Graffunder, senior director of Applications at Cray Research, said that with Ingres functionality on Cray systems, users will be able to apply the world's largest memories and superior computational performance of Cray systems to data management. They wanted Los Alamos to demonstrate that this is true. They offered the one-processor YMP-2E (BALOO) in the Advanced Computing Laboratory (ACL) for Los Alamos to use for the Ingres/Cray trial evaluation. Having a machine dedicated to

this effort allowed us to run basic tests on the product for evaluation without affecting the user community. They provided consulting to go with the installation.

This project was an effective way of using Los Alamos scientific and computational expertise in collaboration with CRI and Ingres to generate interest in the scientific community in database management on supercomputers.

Project Description

Members of Client Services and Marketing (C-6), Internal Information Technology (ADP-2), and Nuclear Theory and Applications (T-2) Groups at Los Alamos National Laboratory (LANL) collaborated with representatives from Cray Research, Inc. (CRI) and Ingres Corporation to successfully complete the Collaborative Evaluation Project of Ingres On The Cray (CEPIC). The project objectives were to determine:

- how easy it is to use Ingres on Cray computers and if Ingres runs in a reasonable time similar to current utilities in an environment that people like and will use;
- whether current standard database applications could be ported to the Cray and the level of effort required to accomplish such porting;

- how well Ingres performs on the Cray in managing data used in and generated by large scientific codes;
- what additional hardware is required to use Ingres on the Cray;
- whether Ingres is compatible with Los Alamos UNICOS.

Project Plan

The CEPIC project plan was to:

- install Ingres on the Cray;
- port an existing traditional database and its three applications from a VAX to the Cray and run compatibility and performance tests;
- create an Ingres database (NUCEXPDAT) and its application (EDAAPPL) to manage the data and results for an existing scientific code (EDA);
- run performance tests on the Cray and other machines;
- install Ingres on machine RHO, a Cray Y-MP8/128 in the Los Alamos ICN
- port the NUCEXPDAT database and the EDAAPPL application to machine RHO and run Los Alamos UNICOS compatibility and performance tests on machine RHO

Installation Of Ingres On A Cray Y-MP

After deciding on the system configuration and Ingres file location, Ingres 6.3 was installed on a Cray Y-MP, named BALOO. It was configured as follows:

Cray CPU: Y-MP2E/116, S/N 1620
 1 Central Processor
 16 Million 64-bit words of central memory
 1 HISP channel to IOS
 1 LOSP channel to IOS

I/O Subsystem (IOS): Model E, serial number 1620

1 I/O Cluster
 1 Million words of buffer memory
 1 HISP channels to mainframe memory
 1 LOSP channel to CPU
 2 HIPPI channels

Disks: 15.68 GBytes on-line storage
 8 DD-60 drives
 1.96 GB (formatted) per drive
 20 MB/s peak transfer rate per drive

Because BALOO was located in the ACL test environment and was being used for non-database work, no database performance tuning was done on the machine.

EDA Physics Code

The Los Alamos code EDA (Energy Dependent Analysis) was chosen for this study because it has data management requirements representative of many of the scientific codes used at the Laboratory. EDA is the most general, and among the most useful, programs in the world for analyzing and predicting data for nuclear reactions among light nuclei. In its present form, the code can be used only on Cray computers, where all of its data files reside. These data files are represented by the boxes shown in Fig. 1; the ones in the upper part of the figure are input files, and those in the lower part are output files (results) of the analysis. Because of the size and complexity of the data files that are used and produced by the program, the data management tasks associated with EDA are quite challenging.

The primary data are the results of experimental measurements (upper left-hand box of Fig. 1) for reactions that can occur between light nuclei. Because this information also has the most complex data structure, we decided to concentrate on these files for the CEPIC demonstration project. An experimental data library containing on the order of 50,000 measured points had already been assembled in the specific form required by the code before the project began. These data entries are classified according to several identifiers, including those for compound system, reaction, energy, observable type, and reference source. At run time, the user selects a subset of these

data to be used for a particular problem. This was being done mainly by grouping data

associated with different compound systems in separate files.

Energy Dependent Analysis

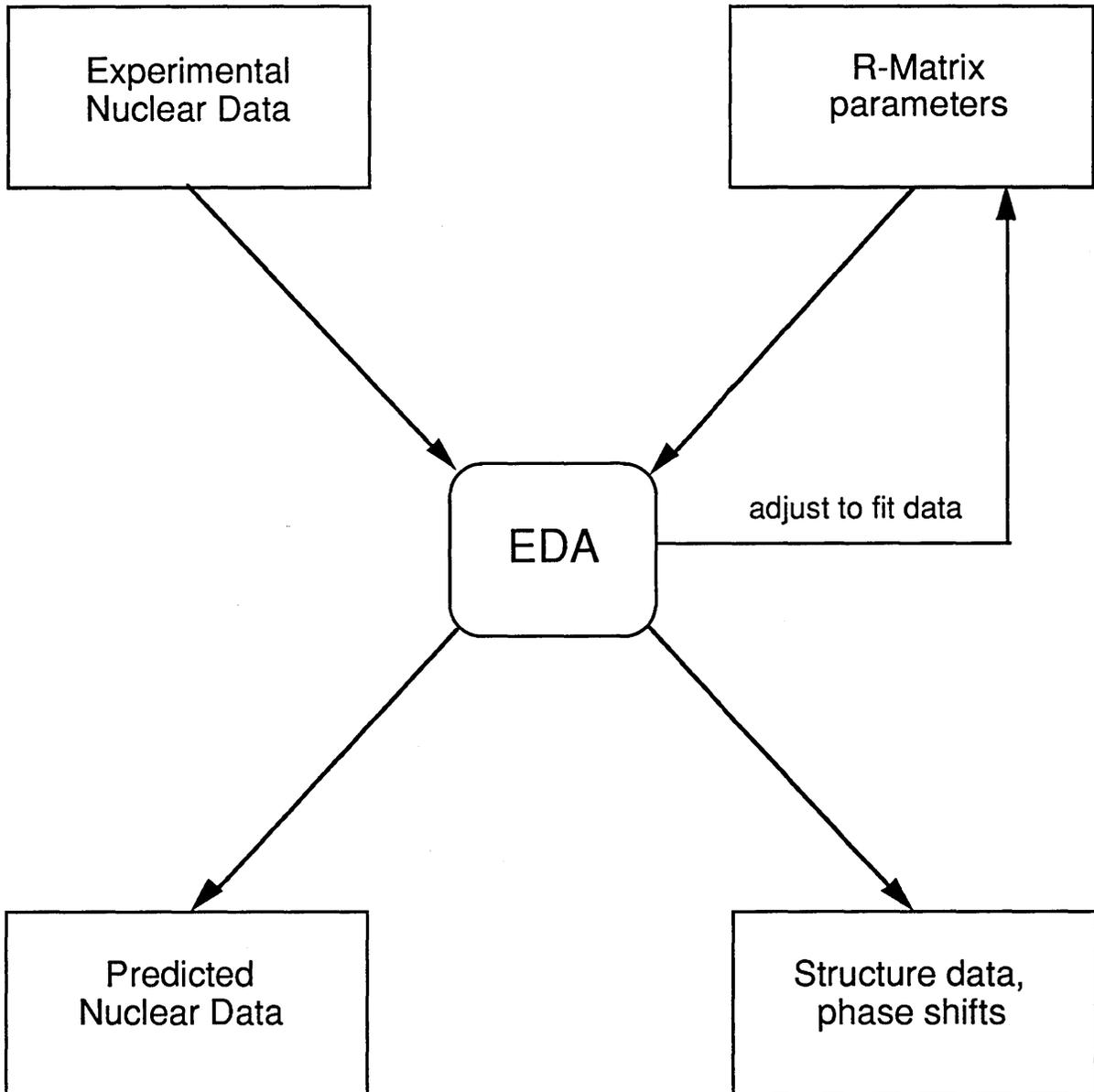


Figure 1. Schematic Of EDA Physics Code Files

We anticipated that putting the experimental information into an Ingres-based data management system would allow far more selectivity in the choice of data (e.g., by energy ranges or data type) than was possible with the existing system. Also, for purposes of experimental data compilation, which is an aspect of the EDA activity that is of great potential interest to outside users, it would provide the capability to sort the entire experimental data file according to any hierarchy of the identifiers listed above. However, we also obtained some unexpected benefits, related to the ease and accuracy with which new data could be entered into the system, and publication-quality

bibliographies of the experimental data references could be generated in a variety of formats.

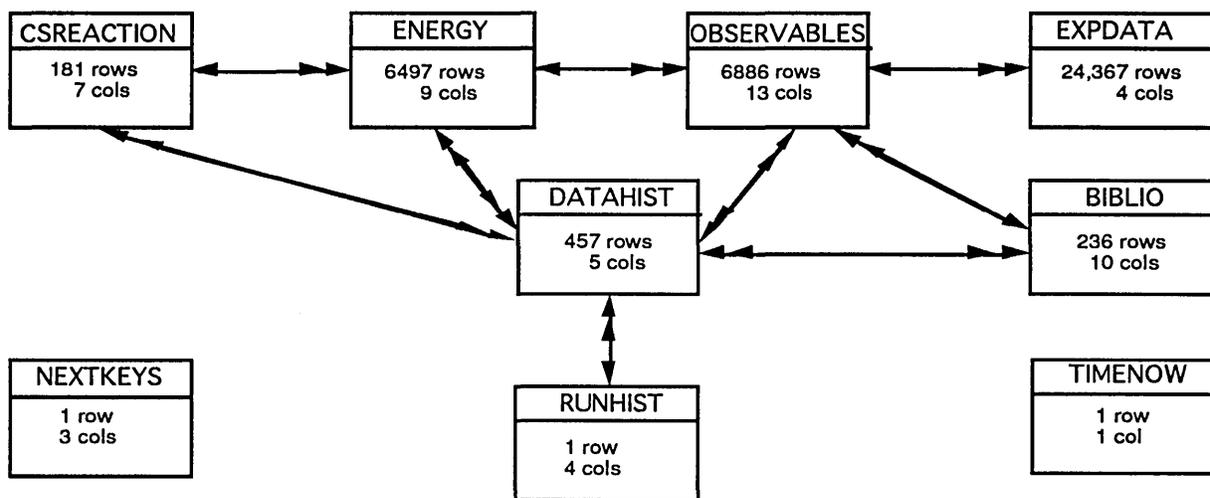
NUCEXPDAT Database

The nine tables and four views in the NUCEXPDAT database are summarized in Table 1. A schematic of the table relationships, showing the number of rows and columns in each table, is given in Fig. 2. Unique identifiers join all tables except NEXTKEYS and TIMENOW. Single and double arrows indicate one-to-many and many-to-many relationships.

<u>NAME</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
BIBLIO	table	Bibliographic data
CSREACTION	table	Compound System Reaction data
DATAHIST	table	Date and data identifiers of EDA runs
ENERGY	table	Energy data
EXPDATA	table	Experimental data (angle, value, error)
NEXTKEYS	table	Next keys for BIBID, RNO, ENO, and OBSID
OBSERVABLES	table	Observables data
RUNHIST	table	History of EDA runs (date, energy range, notes)
TIMENOW	table	Time stamp
BIBLIOVW	view	Join of CSREACTION, ENERGY, OBSERVABLES, and BIBLIO tables used for BIBLIORPT, BIBNPLTRPT, BIBNPRPT, BIBPRLTRPT, BIBPRRPT reports
EDADATAVW	view	Join of CSREACTION, ENERGY, OBSERVABLES, EXPDATA, and BIBLIO tables used for EDADATARPT and EXPDATARPT reports
RENOBBIBVW	view	Join of CSREACTION, ENERGY, OBSERVABLES tables used for BLBIBIDRPT report
RENOBDATVW	view	Join of CSREACTION, ENERGY, OBSERVABLES, and EXPDATA tables used to view all data through QBF

Table 1. NUCEXPDAT Tables And Views

NUCEXPDAT DATABASE



Total: 22.5 Mbytes

Figure 2. Schematic Of NUCEXPDAT Table Relationships

EDAAPPL Application

Shown in Fig. 3 is a flow chart of the EDAAPPL application, while Table 2 describes its frames and procedure. Data

entry and update frames are BIBLIOFRM, DATENTFRM, DATUPDFRM, FIXANGLEFRM, FIXENERGYFRM, RESEFRM, and RESERNGFRM. Report frames end in RPT.

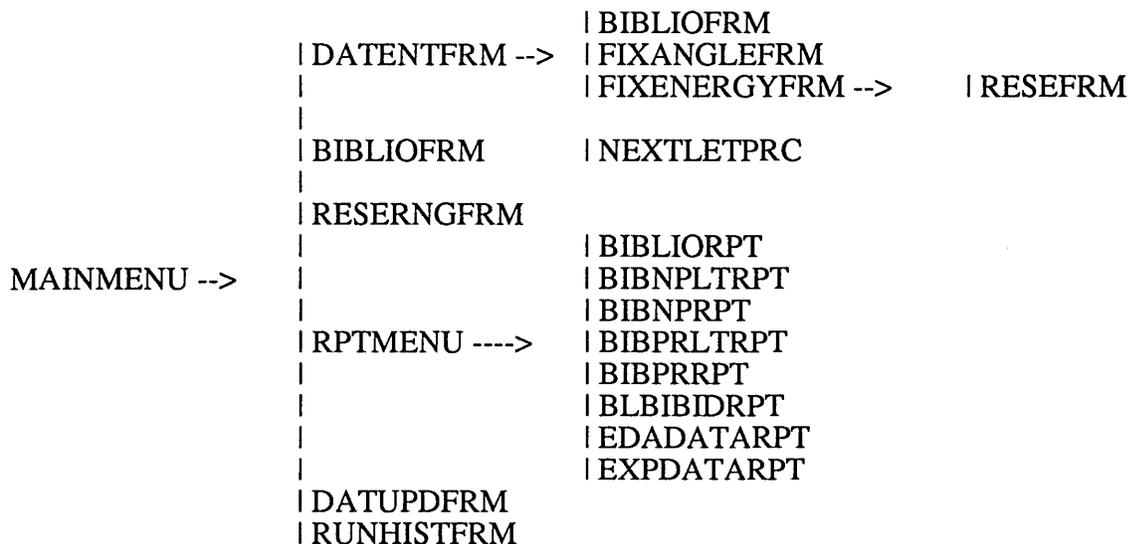


Figure 3. Flow Chart Of EDAAPPL

NAME	TYPE	DESCRIPTION
BIBLIOFRM	user	Add data to the BIBLIO table
BIBLIORPT	report	Generate report of data in BIBLIO table with system, reaction, and observable
BIBNPLTRPT	report	Generate bibliography in Nuclear Physics form with LaTeX command for bold face type
BIBNPRPT	report	Generate bibliography in Nuclear Physics form
BIBPRLTRPT	report	Generate bibliography in Physical Review form with LaTeX commands for bold face type
BIBPRRPT	report	Generate bibliography in Physical Review form
BLBIBIDRPT	report	Generate a list of blank BIBIDs
DATENTFRM	user	Add data to CSREACTION, ENERGY, OBSERVABLES, and EXPDATA tables
DATUPDFRM	user	Update data in CSREACTION, ENERGY, OBSERVABLES, and EXPDATA tables
EDADATARPT	report	Generate EDA input data file
EXPDATARPT	report	Generate file of experimental data with bibliographic references
FIXANGLEFRM	user	Add excitation function data (energy, value, and error)
FIXENERGYFRM	user	Add angular distribution data (angle, value, error)
MAINMENU	user	Select menu choices for application
NEXTLETPRC	procedure	Take a character and return the next greatest one
RESEFRM	user	Update blank resolution data fields in ENERGY for a given compound system, energy, and reaction
RESERNGFRM	user	Add resolution data to ENERGY for an energy range
RPTMENU	user	Select menu choices for report
RUNHISTFRM	user	Keep a history of data files used in EDA runs

Table 2. Description Of EDAAPPL Frames And Procedure

What was done

In the analysis phase, we specified the data, their relationships in the various tables, the kinds of forms needed to cover all possible types of experimental data that would be input to the database, and the reports, including the format of the EDA data file. Then, the database and its application were designed and created to meet these specifications.

Code was written that was based on that part of EDA that processes the input data. This allowed existing EDA data files to be read directly into the database tables. Even though the EDA input code and input file format are extremely complex, in less than two weeks this code was written and used to

load about half of the existing data into the NUCEXPDAT database.

We located, with the help of an earlier bibliographic file, many complete references to the experimental data and put them into the database. We obtained, for the first time, a complete set of references for the data that had been used in an analysis of reactions in the ^5He nuclear system.

Many different types of reports were generated from the database, including EDA run files, bibliographic files in standard-text and TeX format, using the style of either of the two major nuclear physics journals, and annotated listings of experimental data. The run files were checked by using them in actual EDA calculations, and verifying that the answers duplicated the results of previous

runs. By entering qualification data and then pressing just one key, we were able to generate the EDA data file for the ^5He compound system in eleven seconds (clock time).

After installing Ingres on RHO, we unloaded the database and copied the application from BALOO. The entire procedure required less than fifteen minutes and was straightforward to do because both Cray machines were running the same version of Ingres. Operations on the database and using the application gave identical results as on BALOO. We had no problems running Ingres nor did we have to modify any code because of Los Alamos UNICOS. Using Ingres on RHO was frustrating because of delayed responses when there were many users on the machine. Machine time was comparable on the two Crays.

Discussion

Creating an application using ABF was identical to creating one on a VAX or a SUN only much faster. Because compilation and report saving times were so short and because reports using complicated queries of lots of data could be run in real time, application development time was greatly diminished. The EDA code is an extreme case of program complexity for reading input. Therefore, rewriting the relevant code to process existing data files and entering the data in the database in less than two weeks suggests a very short conversion time for codes with more usual data reading requirements.

The EDA experimental data application (EDAAPPL):

- makes the existing experimental data files more uniform, especially in labeling and bibliographic information;
- allows rapid and easy editing of data files (by reaction, energy ranges, data type, etc.);
- simplifies and speeds up the task of entering new data, and provides some error checking;

- produces readable listings of experimental data for outside distribution; and
- greatly facilitates the generation of bibliographies for reports, articles, etc., which reduces the time it takes to document research for publication;
- creates data files so quickly that there is no reason to save these files on CFS to reuse, thereby reducing long-term mass storage requirements.

These capabilities could make significant changes in the way one of the authors (G. H.) does his work. He could spend more time on physics and less on editing, file management, and looking for the right data decks. One of the most tedious parts of writing his papers, compiling the bibliography of experimental data references, can now be done automatically, using the BIBID as a bibitem label in a LaTeX-formatted bibliographic file. The fact that there are no longer funds to hire specialists that enter and help manage data can be off-set in some cases by the ease of using a well-designed, automated DBMS.

If Ingres becomes available on the ICN Cray computers, we would first put the remaining data from the existing data files into the NUCEXPDAT system, along with their associated bibliographical references. Any new data, of course, would be added to the system using the data-entry forms.

Then we would address the other input- and output-file questions. The files containing the parameter values and covariances should be fairly straight-forward to put into a DBMS system, because they are relatively small and have a fixed format. Using these files, the code can predict the results of any measurement and their associated uncertainties. In many applications, large files of these predicted quantities, produced in a complicated and rigid data format, are the desired output of the analysis.

In other cases, resonance parameters and/or phase-shifts that are of more fundamental interest are produced. Some of the requests received for information from EDA analyses

are delayed because of difficulties in retrieving and reconstructing the information used in the analysis at its last stage (especially if it was done some time ago). Implementing the same type of sophisticated data-base system that was used for the experimental data files to manage the many types of information that are produced as output files would benefit all aspects of the EDA work at Los Alamos.

Many of the data management problems solved by using a database management system are similar to those of scientific applications involving gigabytes or terabytes of data. For these applications a DBMS could be used to query the metadata, load the files containing the full data set needed for a particular run, and keep track of runs and their resulting output files. The database could contain metadata of three types: user administrative data; internal data management data; and storage information data. A menu-driven 4GL application could be written to get, list, update, and put files on a storage system and to run analysis codes including those producing images.

Conclusions

Project participants feel that Ingres worked well on the Cray computers in a reasonable time in an environment that people will like and will use.

It was no more difficult or time-consuming to port an existing database and applications to the Cray than it has been to port them to other platforms.

For both scientific and traditional database applications, Ingres looked, acted, felt, and ran the same on the Cray as on other platforms, only it was faster. Because the database and target execution machines are the same, accuracy when going from one machine to another is not an issue.

It was possible to create a useful scientific database application on the Cray in a reasonably short time. Application development time was less because the developer was able to try things without long

waits for compilation of code, reports, and forms. Queries for reports were able to be coded in a single SQL statement instead of in steps because queries, even with aggregates, were fast.

No additional hardware was required to use Ingres on the Cray.

Ingres was compatible with Los Alamos UNICOS.

The availability of a database management system like Ingres on a Cray greatly enhances the efficiency and flexibility of users and producers of large quantities of scientific data on supercomputers.

**PROVIDING BREAKTHROUGH GAINS:
CRAY RESEARCH MPP FOR COMMERCIAL APPLICATIONS**

Denton A. Olson

**Cray Research, Inc.
Eagan, Minnesota**

Abstract

Since announcing plans to build the CRAY T3D massively parallel processor (MPP), Cray Research, Inc. has been approached by numerous customers who have enormous needs for database mining. Many are not our traditional science and engineering customers. These customers want to mine information about, for example, buying trends, from their terabyte-sized databases. They are not asking for help with their payroll or accounting systems. They are not asking for COBOL. They want to build huge decision support systems. They want to search for reports and drawings done ten years and seven million pages ago. These customers are looking for MPP to provide breakthrough gains for competitive advantage. They have hit the computational brick wall with their traditional mainframes and the current MPP offerings. They have come to the supercomputer company for help.

Background

According to Bob Rohloff, Mobil Exploration & Producing Division vice president, Mobil Oil cannot tolerate the "data dilemma of the geoscientist who spends as much as 60 percent of his time simply looking for data", not working with it. (Source: Open Systems Today, July 20, 1992, "Mobil's Collaboration")

One of the reasons Cray Research originally ported database management systems (DBMS) to our computers was to address the concerns of scientists and engineers who spent much of their time just *looking* for data – and possibly having to recreate the data when it cannot be found – rather than *working* with the data. Our customers have asked us to provide database management systems on Cray Research systems that will help address their data management needs.

Historically, Cray Research parallel vector processing (PVP) systems have provided numerous strengths for DBMS processing, including: large memories (e.g., the CRAY Y-MP M90 series with up to 32 GBytes of real memory); fast memory (250 GBytes/sec for CRAY C90); fast and configurable I/O subsystems for connecting to many different kinds of peripherals; UNICOS for UNIX compatibility; network supercomputer protocols required for implementing client-server architectures; and the ability to embed SQL (the ANSI-standard structural query language of most DBMS) in vectorized and parallelized C and FORTRAN codes.

Third party DBMS products now available on Cray Research PVP systems are INGRES, ORACLE, and EMPRESS.

New Market Needs

There is a new class of customer that has approached Cray Research in the past year with needs that are often quite different from those of scientists and engineers. These needs are in the commercial area of 'database mining'—searching the contents of databases looking for relationships and trends that are not readily apparent from the data structure.

These customers want to find sales trends, do marketing analysis, and look for drawings that they have stored a million pages and many years ago. They present a wide spectrum of requirements (see Figure 1) such as decision support; report generation; data visualization; econometric modeling; and traditional scientific/engineering applications.

MPP-DBMS Is Core Requirement

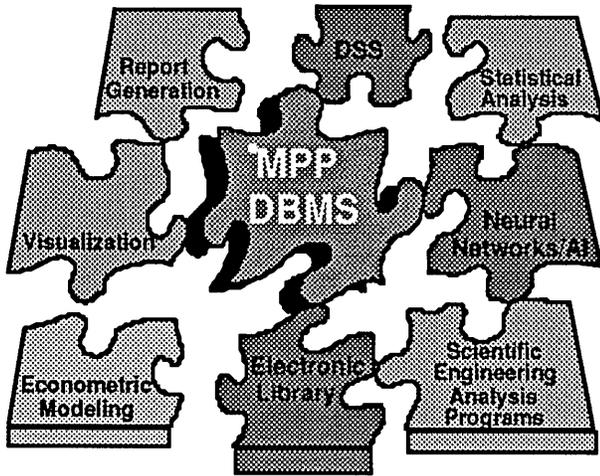
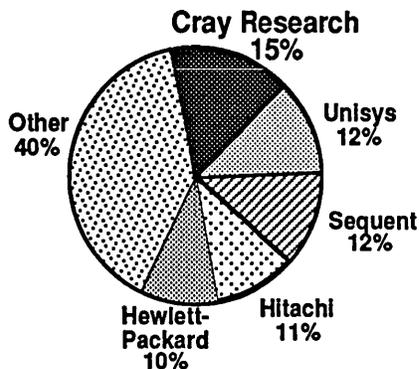


Figure 1

The consistent *core* requirement for these customer inquiries is an MPP database.

These customers have run into performance 'brick walls' with their current systems. They have large AT&T-GIS/Teradata systems and many IBM 3090s and ES-9000s. They came to Cray Research for help — even before we announced the CRAY T3D. These customers are moving to Open Systems, which right now means UNIX. They sought out the high-end suppliers of UNIX systems and, as Figure 2 shows, found Cray Research.

Percentage of the Market for Large Unix Systems



Total estimated 1993 market: \$3.5 billion

Source: InfoCorp. and September '93 Electronic Business Buyer

Figure 2

Cray Research's UNIX market share and experience gives these customers a level of comfort about a major player in this market being able to provide their UNIX needs.

These commercial customers are *not* asking for COBOL or payroll systems or traditional data processing applications. Many are asking Cray Research for help with Decision Support Systems (DSS) not the time-critical OnLine Transaction Processing (OLTP) or Point-Of-Sales (POS) systems. They need fault-resistance not fault-tolerance. There are a limited number of sophisticated users on-line — not the hundreds of users that might be required for banking or reservation systems. There are also significant 'number crunching' components to their applications — in concert with the database processing they want to do.

Requirements for MPP-DBMS

A technical overview of the CRAY T3D is beyond the scope of this paper and is adequately covered in other CUG papers. But briefly, when studying hardware requirements for running databases on MPP platforms, the following features must be examined:

- CPU performance. The 150-MHz Alpha RISC microprocessors from Digital Equipment can provide the required horsepower needed for running scalar, CPU-intensive, MPP-DBMS applications.
- Latency. Low latency is important for MPP-DBMS codes that currently rely on message-passing models. The CRAY T3D provides industry-leading low latency.
- Bandwidth. MPP databases need to move massive amounts of data. The CRAY T3D provides unrivaled bisection bandwidth.
- I/O. The CRAY T3D's Model E IOS subsystem provides a wealth of peripheral capabilities for disks, production tapes and a large spectrum of standard networking protocols.

Conclusion

Cray Research's activities in providing MPP databases for Cray Research systems are part of a Data Intensive Systems (DIS) focus. We are in discussions with numerous customers who have DIS and MPP-DBMS needs. These customers span a wide spectrum of industries from petroleum, to government, to retailers.

Cray Research is studying the hardware and software requirements for MPP-DBMS, and is establishing requirements for our follow-on products. We are also in discussions with various system integrators who can provide great synergy and help us move into the commercial marketplace.

Customers and prospects continue to come to Cray Research for help because it is clear from what they are telling us that there is no MPP-DBMS 'winner' to date.

Asynchronous Double-Buffered I/O Applied to Molecular Dynamics Simulations of Macromolecular Systems

Richard J. Shaginaw, Terry R. Stouch, and Howard E. Alper

*Bristol-Myers Squibb Pharmaceutical Research Institute
P.O. Box 4000
Princeton, NJ 08543-4000*

shaginaw@bms.com stouch@bms.com

ABSTRACT

Researchers at the Bristol-Myers Squibb Pharmaceutical Research Institute use a locally modified Discover (tm Biosym Technologies, Inc.) to simulate the dynamics of large macromolecular systems. In order to contain the growth of this code as problem size increases, we have further altered the program to permit storage of the atomic-neighbor lists outside of central memory while running on our Y-MP2E/232 with no SSD. We have done this efficiently by using BUFFER OUT and BUFFER IN to perform asynchronous double-buffered I/O between neighbor-list buffers and DD-60 disk storage. The result has been an improvement in turnaround for systems currently under study (because of reduced swapping) and enablement of much larger simulations.

1.0 Introduction

Each atom in a molecular dynamics simulation of a very large molecule or of a macromolecular system must sense the attractive/repulsive forces of neighboring atoms in the system. These atomic neighbors include the atoms covalently bonded to the atom in question as well as those not bonded to it but lying within a prescribed cutoff distance. Faced with the choice of either identifying all neighbors each time step or maintaining a periodically updated list of neighbors, researchers ordinarily choose the latter approach.

The size of such a neighbor list is linearly proportional to atom count and geometrically proportional to cutoff length. For researchers interested in treating very large systems as accurately as possible, the physical limitation of computer memory is a handicap. Especially in the Y-MP environment, with expensive central memory and no virtual paging, the hardware limits the size of the problem. On less powerful platforms, these computationally intensive problems require so much time as to be intractable. Moreover, a virtual-memory system without ability to advise the system to pre-fetch pages cannot accommodate an extremely large program.

We have solved this dilemma by using the BUFFER IN and BUFFER OUT statements in FORTRAN on the Y-MP to achieve asynchronous transfer of very

large neighbor lists to high-speed disk as the program creates the neighbor list, and from disk each time step when the neighbor list is needed. This paper details the implementation strategy and our results. Section 2 is a more complete statement of the problem. Section 3 discusses our programming strategy in detail. Section 4 presents the FORTRAN specifics of the implementation. Section 5 is a summary of our results and conclusions.

2.0 Statement of Problem

Numerical simulation of a biological system using molecular dynamics techniques requires repeated calculation (each time step along a numerically integrated trajectory) of the force exerted on each atom in the system by every other atom within a specified cutoff radius. Each atom then moves once per time step in response to these forces. The atoms whose electronic forces affect the motion of a given atom are considered "neighbors" of that atom. Most of an atom's neighbors are not chemically bonded directly to that atom; nevertheless, their position, charge, and motion are vital pieces of information.

All atoms in a system under study are indexed using positive integers. One approach to molecular dynamics involves tracking every atom's neighbors by keeping a list of neighbors' indices in an integer array. In turn, two other integer arrays maintain for each atom

a pointer to its first neighbor's position in the list and another pointer to the last. The program reconstructs the neighbor list at a interval specified by the researcher based on his or her expectations for the movement of atoms into and out of cutoff range.

Obviously, a large macromolecular system contains many atoms; the neighbor list grows approximately linearly with atom count. Moreover, accurate simulation requires a long cutoff distance; the neighbor list grows geometrically with cutoff distance. Therefore the real memory available to a program limits the size and accuracy of any simulation.

At BMSPRI, researchers are interested in the structure and dynamics of lipid bilayers as instantiated in animal cell membranes. The membrane's structure and dynamics control the transport and diffusion of compounds in their vicinity, especially those which cross the membrane into or out of the cell interior. The compounds of interest include drug molecules, toxins, antigens, nutrients, and others. The cell membrane incorporates a variety of embedded proteins, some which function as receptors for a variety of stimuli, and others which act as ion channels.

As they have focused their resources on the lipid bilayer, BMSPRI scientists have increased the size and accuracy of the simulations they use in their research. This increase has led to a non-bond neighbor list approaching 7 million Cray words in size. Added to an already large, complex program, this memory requirement has pushed the size of the program beyond 14 megawords (roughly half of our Y-MP2E/232). Research progress dictates that future simulations must embrace much larger systems with longer cutoffs. In order to achieve this, the researchers have decided to try to reduce the strong dependency of program size on molecular system size and cutoff distance.

3.0 Solution Strategy

Multiple-buffered asynchronous I/O is in common use in graphical animation, in numerical modeling of very large physical systems, and in other computer applications. The basic approach is to create and open at least one file, using specialized I/O library functions or subroutines. These calls permit data to bypass the I/O buffers that are ordinarily part of user data space (in the FORTRAN or C library) and to bypass the buffers that are maintained by the system in kernel space. In other words, these I/O calls permit data

transfer directly between the user program's data space and unstructured files on the storage medium. The programmer uses at least two program arrays as I/O buffers, and the program must include the book-keeping needed to make well-formed I/O requests (I/O transfers that are integer multiples of the disk device's physical block size). Avoiding library and system buffers permits program execution to continue while I/O proceeds. Special calls then permit blocking of execution in order to synchronize data transfers before writing to or reading from the program arrays functioning as buffers, to protect data integrity.

Cray Research supports several techniques for asynchronous I/O. Table 1 outlines these.

Table 1. Cray Research Asynchronous I/O Options.

Technique	Description
READDR/WRITDR	record-addressable random-access I/O
GETWA/PUTWA	word-addressable random-access I/O
AQREAD/AQWRITE	queued I/O
BUFFER IN/OUT	unbuffered, unblocked I/O

As structured at the start of this effort, BMSPRI-modified Discover generates three neighbor lists; two of these are subject to rapid growth with cutoff distance, and so these two are our candidates for disk storage. Our strategy is to store both lists of neighbors on disk at the time of list creation, and then to read up this list each time step in the routines that compute the non-bond energies of the system. We have no need for record- or word-addressable random access, because we know a priori that the energy routines require the data to be read sequentially. Likewise, the random-access capability of the queued I/O calls is unnecessary. We have decided to use BUFFER IN and BUFFER OUT to achieve asynchronous transfer.

For efficiency, the ratio of transfer time to the quantity (transfer time + latency) must be close to unity; therefore the buffer size must be sufficiently large to overwhelm latency. In contrast, for I/O to overlap execution completely, the buffer size must be sufficiently small to permit completion of the transfer before the buffer is needed again.

Our fastest filesystems reside on unstriped logical devices built on DD-60 drives, with one drive per I/O

channel. The fastest user-writable filesystem is one we call /tmp/people, a continuous scratch area of about 5 GB, where every user with an account owns a directory. The worst-case maximum rotational latency for DD-60 devices is 26 milliseconds, according to Cray Research. We have found that unbuffered writes to existing unblocked DD-60 files run at about 19 megabytes per second, while unbuffered reads from the same files proceed at about 16 megabytes per second. This asymmetry may be due to the fact that each read requires a seek operation, while the drives when idle are positioned for writing.

At 16 MB per second, the smallest I/O request size that permits 90% efficiency is 490,734 words. The loop timing in the non-bond-energy routines (under a system load of four simultaneous batch jobs) averages 0.18 seconds, and so the maximum transfer size to achieve overlap is 377,487 words. These limiting values clearly eliminate the possibility of 90% efficient asynchronous I/O in our case. Nevertheless, we have chosen to accept an efficiency level of less than 90% in order to test our strategy. We have chosen a buffer size of 409,600, which is exactly 200 DD-60 sectors in length. This buffer size leads to an efficiency of 88%, but may lead to incomplete overlap on the read side under typical load. In the case of heavy load, overlap on both read and write will be complete.

We have decided to employ two files, each corresponding to one buffer, for each list, in order to maximize overlap in end cases. In other words, we use four files in the current implementation. We use the same buffers for reading and for writing, and for both neighbor lists.

4.0 FORTRAN Implementation

BMSPRI uses the program Discover from Biosym Technologies, Inc, to carry out its lipid bilayer simulations. The Institute holds a source license for Biosym, and researchers have modified the program substantially, to include theory useful in their specific problem area. Two neighbor-generation routines use several criteria to determine the relationship of each atom in the system to every other atom in the system. These routines create two separate integer lists of neighbor indices. Two nonbond-energy routines read through these neighbor lists each time step; these integer lists point into an array containing charge and position data for each atom. Four routines contain all the code modifications made in the double-buffering

effort.

Two COMMON blocks contain six control variables necessary for bookkeeping and the buffers themselves, dimensioned (LBUF,2) where LBUF = LREC + LPADD; LREC = 409,600 and LPADD is a pad-region length, set to 10,000 words.

Each neighbor-generating routine has two loops that iterate across all "atom groups" in the system. The first of these loops is operative in the case where all atoms in the system are permitted to move; the second loop, when some atoms are held fixed in space. The energy routines each contain three exclusive loops in which the neighbor list is read, one atom at a time.

In the list-generation routines, the first action taken is to synchronize both files used by each of the two neighbor lists, using the FORTRAN extension LENGTH. Then the program repositions both files into which it is about to write to the beginning of data. Then we initialize all control variables. During each iteration of the main loop, the program stores each atom's neighbor indices sequentially into the "current" buffer and advances the buffer pointer. At the end of each iteration, the program tests the buffer pointer, and if the buffer has overflowed into the pad region, it initiates a BUFFER OUT for the current buffer. If this is a second or subsequent write, it uses LENGTH to synchronize the write of the alternate buffer. Then it moves the content of the pad region to the start of the alternate buffer. At this point, the program switches the alternate buffer to current status. Then, whether the "buffer full" test has passed or failed, if this is the last loop iteration, the program initiates a BUFFER OUT of the current buffer; otherwise it continues iterating.

The first action in the non-bond energy routines is to synchronize all four files and to initialize local control variables. Then we reposition both files about to be read to the beginning of data. Then the program initiates reads into both buffers and then blocks execution, using LENGTH, to synchronize the read into the current buffer. The program uses buffer-swapping techniques analogous to those in the generation routines to manage the buffers during loop iteration.

To achieve asynchronous I/O, the files in use must be typed as "unblocked" files. The UNICOS command "assign" with the option "-s u" creates a file of type "unblocked". We name these files uniquely to each batch job by including the C-shell process-ID substitution string "\$\$" in each of the four file names. At

the end of the job, we remove all four files.

5.0 Results and Conclusions

Implementation of these code changes has led to a reduction in the size of the executable code for a 30,000-atom case with a cutoff of 12 Angstroms by 3.4 Megawords. Moreover, no longer is program size dependent on cutoff length.

The overhead incurred by BUFFER IN/OUT and the additional bookkeeping in the program has led to an increase in CPU time of 1% to 2%. This will cause in our environment a worst-case increase in turnaround time of one day (out of 6 weeks of wallclock time) for a nanosecond of simulated time. This turnaround delay is acceptable to Institute researchers. Moreover, the improvement in turnaround time due to a reduction in swap-in-queue residency more than compensates for this disimprovement in most cases. On the other hand, at this point this code sustains an increase of I/O wait time from effectively zero to between 3% and 5% of CPU time. We expect this wait time to increase turnaround time to an unacceptable level. Profiling of the effected routines reveals that essentially all of these I/O waits occur on the read side, in the non-bond energy calculation routines. We believe that this reflects the lower speed of a typical read. Writing the contents of a 409,600-word buffers to an unblocked file resident on unstriped DD-60 takes an average of 0.16 seconds; reading a 409,600-word unit of data from the same file into a buffer takes about 0.19 seconds. With our typical system load of four simultaneous batch jobs, our I/O scheme tries to do a read or write every 0.17 to 0.18 seconds, on average. This asymmetry between read and write performance can explain the additional I/O wait time.

Our next modification to this program will be to add a third buffer to accommodate a read-ahead of the data chunk beyond the "next" in the energy routines. This should nearly eliminate the I/O wait overhead. We also plan to experiment further with the queued asynchronous I/O strategies.

A PVM Implementation of a Conjugate Gradient Solution Algorithm for Ground-Water Flow Modeling

by

Dennis Morrow, John Thorp, and Bill Holter

Cray Research, Inc. and NASA Goddard Space Flight Center

SUMMARY

This paper concerns the application of a conjugate-gradient solution method to a widely available U.S. Geological Survey (USGS) ground-water model called *MODFLOW* which was used to solve a ground-water flow management problem in North Carolina.

The performance of the *MODFLOW* model incorporating a polynomial preconditioned conjugate gradient (PPCG) algorithm is presented on the Cray C90, and a PVM implementation of the algorithm on the Cray T3D emulator is outlined. For this large-scale hydrologic application on a *shared memory* supercomputer, the polynomial preconditioned conjugate gradient (PPCG) method is the fastest of several solution methods which were examined. Further work is needed to ascertain how well PPCG will perform on *distributed memory* architectures.

The sections in this paper first introduce the USGS *MODFLOW* model and its application to a North Carolina ground-water flow problem. Next the PPCG algorithm and similar CRAY library routines are discussed, followed by tables of CPU timing results and the ratio of parallel speed-up attained by the *MODFLOW* model on the Cray C90.

The final section discusses the distributed memory implementation of the PPCG algorithm using PVM on the Cray T3D emulator followed by a summary and plans for future work.

ABSTRACT

There is a need for additional computing power for modeling complex, long term, real-world, basin-scale hydrologic problems. Two examples which illustrate the computational nature of ground-water modeling are:

1. the stochastic nature of the model input data may require a sensitivity analysis for each model input parameter and/or a number of monte-carlo simulations
2. optimal wellfield pumping scenarios for minimizing

drawdown or for control of contaminant plumes or saltwater intrusion can require many independent simulations

The need to model larger more complex problems is coupled with a need for applying more efficient parallel algorithms which can take advantage of supercomputer hardware and reduce the wall-clock time to get a solution.

INTRODUCTION

This paper examines replacements for the matrix solution algorithm used in a USGS ground-water model, *MODFLOW*. *MODFLOW* is the short name for the Modular Three-Dimensional Finite Difference Ground-Water Flow Model [10], a publically-available model which is widely used in industry, academia, and government. *MODFLOW* simulates transient ground-water flow and can include the influence of rivers, drains, recharge/discharge wells, and precipitation on both confined and unconfined aquifers.

A WATER RESOURCE MANAGEMENT APPLICATION

The application presented in this paper is a water resources management study conducted by Eimers [2,3] with the *MODFLOW* model to determine the influence of pumping on a 3,600 square-mile study area that is a subset of the ten-vertical-layer aquifer system composing the 25,000 square-mile North Carolina Coastal Plain. The aquifer model consists of 122,400 finite difference cells on a 10 x 120 x 102 grid. This is a transient problem with 120 time steps representing a total simulation time of 87 years. The number of wells in the model ranges from 1,416 to 1,680, although some of these are not actual well sites but are *pseudo-wells* needed to constrain the hydraulic condition at certain political boundaries where there is no corresponding hydrogeologic boundary. Vertical conductance, transmissivity, and storage coefficients can vary by node within each layer but are assumed to have no directional component.

GROUND-WATER SOLUTION ALGORITHMS

Many algorithms are available to solve the ground-water flow equations and there is certainly not one best algorithm for all

CRAY SCIENTIFIC LIBRARY ROUTINES

problems on all computers. MODFLOW constructs a sparse matrix called the *A* matrix from the discrete form of the flow equations, then solves the matrix. The matrix is symmetric, positive definite, and has three off-diagonals. In MODFLOW, 97% of the total CPU time is spent in the *A* matrix solution.

A direct linear equation solver such as banded Gauss elimination computes an explicit factorization of the matrix and in general guarantees an accurate solution. Iterative matrix solvers, such as the *strongly implicit procedure* supplied with MODFLOW, begin with an initial approximate solution and then converge toward the solution, thus improving the approximate solution with each iteration.

Used appropriately, iterative algorithms can be as accurate and much faster than direct methods applied to these kinds of problems. Dubois [1], Hill [5], Jordan [8], Van der Vorst [15] and many others have confirmed the desirable properties of iterative conjugate gradient (CG) solvers on vector computers. Johnson [7] suggested polynomial preconditioners for CG solvers and Oppe, et.al.[12] implemented a general non-symmetric preconditioned CG package on a CRAY.

Specifically for ground-water modeling, Kuiper [9] compared the incomplete Cholesky CG method with the strongly implicit procedure (SIP) described by Weinstein, et.al.[16], and Scandrett [14] extended the work and included timings on the CDC Cyber 205, reporting that PPCG has very good convergence and that the iteration sequence is completely vectorizable. Morrow and Holter [11] implemented a single-CPU vectorized PPCG solver for MODFLOW on the Cyber 205. Saad [13] discussed the steps needed to implement a parallel version of the PPCG algorithm and Holter, et.al.[6] attained 1.85 gigaflops on a CRAY Y-MP8 with a multitasked PPCG solver for a two-dimensional ground-water diffusion problem.

PPCG

The PPCG algorithm provides an efficient, vector-parallel solution of $AX=B$, where *A* is symmetric, banded, and diagonally dominant. It is assumed that *A* has been normalized (by a straightforward procedure) so that all its diagonal elements are equal to unity.

The algorithm utilizes a least squares polynomial approximation to the inverse of *A*, and calls for the repeated multiplication of this inverse and a vector of residuals, *R*. [11]

The steps in the PPCG algorithm can be summarized as follows:

1. Set an initial estimate of the groundwater pressure heads in the aquifer.
2. Compute the vector of residuals
3. Form two auxiliary vectors from the residuals
4. Iteratively cycle through a six-step process which updates the heads and residuals until convergence

As mentioned previously, several variations of preconditioned CG matrix solvers perform well on vector computers. Several of these methods are incorporated into a single routine in the CRAY UNICOS Math and Scientific Library (SCILIB). The routine is called SITRSOL and it is a general sparse matrix solver which allows the selection of a preconditioned conjugate gradient-like method. SITRSOL has many selections for the combination of preconditioner and iteration method. The six options for iterative method (accelerators) are:

1. (BCG) --- Bi-conjugate gradient method
2. (CGN) --- Conjugate gradient method applied with Craig's Method
3. (CGS) --- Conjugate gradient squared method
4. (GMR) --- Generalized minimum residual method
5. (OMN) --- Orthomin/generalized conjugate residual method
6. (PCG) --- Preconditioned conjugate gradient method

For preconditioners, there are also six options:

1. No preconditioning
2. Diagonal (Jacobi) preconditioning
3. Incomplete Cholesky factorization
4. Incomplete LU factorization
5. Truncated Neumann polynomial expansion
6. Truncated least squares polynomial expansion

Not all combinations of preconditioners work with all the selections for accelerators. For instance, Incomplete LU factorization cannot be used with a symmetric matrix in half-storage mode. PCG cannot be used unless the matrix resulting from MODFLOW is always positive definite (it is).

The performance of several of these SITRSOL matrix solution routines is compared for solving a test problem of similar size as the North Carolina ground-water modeling problem (see Table 1.). All the runs were made on one CPU of a YMP-2E.

10x125x125 cell MODFLOW problem				
<u>Algorithm</u>	<u>Pre-conditioner</u>	<u>memory Mwords</u>	<u>algorithm megaflops</u>	<u>CPU seconds</u>
PPCG*	polynomial	1,435	232	190
BCG	least squares	11,400	45	956
CGS	least squares	3,650	106	275
PCG	least squares	3,156	98	232
PCG	Neumann	5,774	127	441
PCG	Cholesky	18,342	16	951
*not a part of SITRSOL				

Table 1. PPCG and SITRSOL timing comparisons.

The column headed *memory* is the CPU memory integral time reported by the UNICOS *ja* command. This indicates an average memory requirement for the duration of execution of the entire code. The numbers can be used to infer a comparative memory requirement for the solution algorithms.

For the SITRSOL solution routines, the least squares preconditioner combined with the PCG iterative method had the lowest time and memory requirement of the five attempted SITRSOL combinations, but PPCG was the overall best. The PPCG algorithm is coded to take advantage of the *specific* sparse-diagonal matrix structure in the MODFLOW model.

SHARED MEMORY PARALLEL-VECTOR STRATEGY

Compiler-level strip mining of DO loops was the data-parallel approach used to implement the PPCG algorithm on multiple processors of the Cray C90. Several modifications to the single-cpu PPCG algorithm were made to accomplish this. Some of these modifications also improved the performance of the single-cpu implementation of the algorithm.

The MODFLOW code has a large scratch array dimensioned in the main program and individual variables are referenced by pointers to locations within the scratch array. In some cases, this degrades data locality. To accomplish data locality and to avoid unnecessary references to memory, MODFLOW variables which were a part of the matrix solution were removed from the large scratch array and recast as arrays which were *local* to the PPCG solution routine. Also, some variables were removed from DATA and SAVE statements in order to have as many variables as possible stored on the stack.

Next, the lengths of the off-diagonals were padded by *zero-filling* to match the length of longest diagonal of the matrix (the main diagonal). This allowed the elimination of some IF tests associated with the shorter diagonals. Also several DO LOOPS of varying lengths could now be collapsed into a single DO LOOP, thus organizing lots of work into a single parallel region. For the North Carolina water management problem, loop lengths were fixed by parameter statements to 122,400 to eliminate the need for execution-time checking of loop lengths prior to strip-mining.

Tables 2. presents the Y-MP-C98 timing results for the entire MODFLOW model solving the North Carolina water management problem (122,400 groundwater cells for the 87 year simulation period) with the *shared memory* implementation of PPCG.

10x120x102 cells (122,400 eqns.) Y-MP- c98					
# CPUs	Wall sec.	CPU sec.	Mflops /CPU	Concurrent avg. cpus	Total Gflops
1	61	57	641	1.0	0.64
2	40	62	589	1.6	0.94
4	31	65	558	2.2	1.2
8	28	67	541	2.4	1.3
8*	15	78	466	5.2	2.4

*dedicated run

All the listed computer runs were made during the day on production systems and none of the results are *benchmark* runs though, as noted, one of the runs was made in the single-job-streaming queue (only one batch job is allowed to run at a time). The reported wall-clock times will vary depending on the number of jobs in the system. The listed CPU times and megaflop rates are fairly independent of system load.

The CPU times increase slightly with additional CPUs, and the reduction in wall clock time illustrates that multitasking can cut the turnaround time on a production system.

DISTRIBUTED MEMORY MESSAGE PASSING STRATEGY

Parallel Virtual Machine (PVM) was used to implement the distributed version of the PPCG algorithm on the T3D emulator running on the multiple processors of the Cray C90.

PVM is a public domain set of library routines originally developed for explicit communication between heterogeneous systems tied to a network [4]. PVM was developed at Oak Ridge National Laboratory, and it has become a de-facto message passing standard.

Message passing is a parallel programming paradigm which is natural for network based or (in this case) distributed memory systems. An additional benefit of the message passing paradigm is that it is portable. A message consists of a user-defined message tag and the accompanying data. Messages may be either point-to-point or global (broadcast). In point-to-point message passing, the sender specifies the task to which the message is to be sent, the data to be sent, and the unique message tag to label the message. Then the message is packed into a buffer and sent to the interconnection network. The receiver specifies the task from which the message is expected, the unique message tag which is expected, and where to store the data which is received from the network.

In actual PVM implementation in FORTRAN, the sending task makes three PVM calls which (1) create the *send buffer*, (2) pack the data into the send buffer, and (3) send the message. Similarly the receiving task makes two PVM calls which (1) receive the message from the network, and (2) unpack the data into the local memory user space.

A general observation is that *message passing is to parallel systems as assembly language is to mainframes*. Considerable modification of the shared-memory version of the PPCG algorithm was necessary to accomplish the implementation with message passing. The explicit nature of message passing can also be tedious (unique message tags, five FORTRAN routine calls to send/receive any piece of data, etc). This discourages frequent communications. Avoiding unnecessary communication is an important part of distributed computing.

The message passing strategy for implementing the PPCG algorithm on the T3D emulator began with equally distributing

the data contained in FORTRAN arrays among the total number of processors. All four diagonals of the *A* matrix (the matrix to be solved) are copied from the master processor (PE0), so that each processor has a local copy of its part of the matrix. This is necessary because the *A* matrix is set up by the MODFLOW model and passed to the matrix solution routine and, to date, only the PPCG solution routine has been implemented in PVM. In the program, the arrays W1, WM1, P, R, SQINV, and B can all be distributed equally across the processors. For example, Figure 1. shows three of these six arrays distributed across four processors.

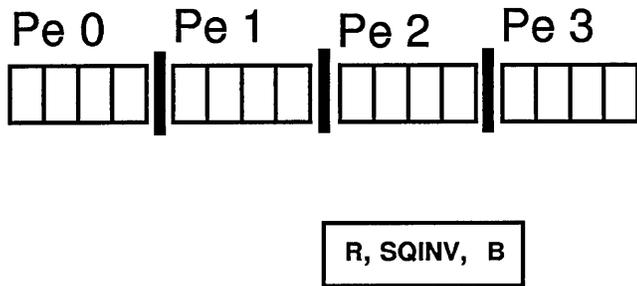


Figure 1. Distributed Linear arrays R, SQINV, and P

A second part of the message passing strategy for implementing the PPCG algorithm was to accomplish communication between processors by making and distributing *offset* copies of the *A* matrix and the temporary work vectors needed in the iterative solution. There are six regular communication patterns which trace back to the three dimensions of the groundwater problem being solved. These six patterns involve offsetting the data by either plus or minus 1, plus or minus *ND*, or plus or minus *NX*, where for this particular groundwater problem, *ND*=12,240 and *NX*=102.

This communication is similar to the End-Off Shift (EOSHIFT) operation where data *does not* wrap around from the last processor to the first. Two of these communication patterns are shown in Figure 2.

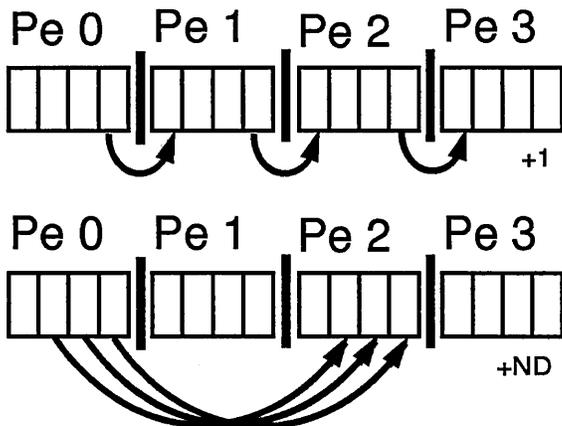


Figure 2. EOSHIFT Operation

Figure 3. shows a complete pattern involving all six offsets. Note that not all offsets will result in off-processor communication. In Figure 3, the -1 offset on PE1 is an *on-processor* communication. The mix between on and off processor communication also will change with the total number of processors.

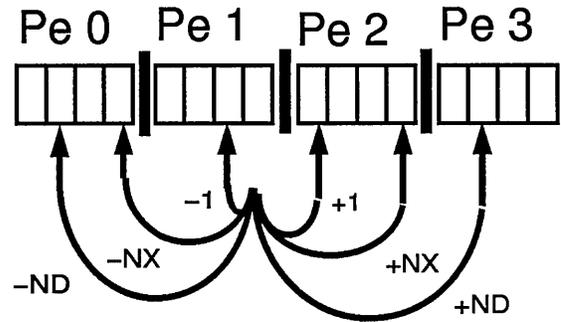


Figure 3. Communication Pattern

Also, there are a number of reduction operations that are accomplished using PVM to communicate the partial sums from each processor back to PE0 using a standard library-like routine.

RESULTS and FUTURE PLANS

For the MODFLOW ground-water modeling application, CPU times for several shared-memory sparse matrix solution algorithms are compared. The PPCG algorithm on a Cray YMP-C98 ran at 2.4 gigaflops and attained a speedup ratio of 5.2. The distributed-memory version of the PPCG algorithm was implemented on the emulator and the next step is to port the code to the CRAY T3D.

CONCLUSIONS

Supercomputers and efficient vector-parallel solution algorithms can speed processing and reduce turn-around time for large hydrologic models, thus allowing for more accurate simulations in a production environment where wall-clock turnaround is important to the ground-water model user.

SUMMARY

A data-parallel *shared memory* implementation and a PVM *distributed memory* implementation of a polynomial preconditioned conjugate gradient solution algorithm for the U.S. Geological Survey ground-water model MODFLOW were presented. The PVM solution is accomplished on multiple processors and can be ported to the T3D.

Performance on the Cray C90 is presented for a three-dimensional 122,400 cell anisotropic ground-water flow problem representing a transient simulation of pumping a North Carolina aquifer for 87 years.

ACKNOWLEDGEMENTS

CRAY, CRAY Y-MP, and UNICOS are federally registered trademarks and Autotasking and CRAY EL are trademarks of Cray Research, Inc. The UNICOS operating system is derived from the UNIX System Laboratories, Inc. UNIX System V operating system. UNICOS is also based in part on the Fourth Berkeley Software Distribution under license from The Regents of the University of California.

REFERENCES

1. Dubois, P.F. et al., "Approximating the Inverse of a Matrix for Use in Iterative Algorithms on Vector Processors", *Computing*, 22, 257-268, 1979.
2. Eimers, J., Lyke, W., and Brockman, A., "Simulation of Ground-water Flow in Aquifers in Cretaceous Rocks in the Central Coastal Plain, North Carolina", Water Resources Investigations Report, Doc I19.42/4:89-4153, USGS Raleigh, NC 1989.
3. Eimers, J. and Morrow, D., "The Utility of Supercomputers for Ground-Water Flow Modeling in the North Carolina Coastal Plain", Proceedings of the ASCE Water Resources Conference, Sacramento, CA May 1989.
4. Grant and Skjellum, "The PVM Systems: An In-Depth Analysis and Documenting Study - Concise Edition", The 1992 MPCY Yearly Report: Harnessing the Killer Micros, LLNL # UCRL-ID-107022-92.
5. Hill, Mary, "Preconditioned Conjugate Gradient (PCG2) -- A Computer Program for Solving Ground-Water Flow Equations", Water Resources Investigations Report # Doc I19.42/4:90-4048, USGS Denver, CO 1990.
6. Holter, B., et.al., "1990 Cray Gigaflop Performance Award".
7. Johnson, O.G. et al., "Polynomial Preconditioners for Conjugate Gradient Calculations", *SIAM J. Numer. Anal.*, 20(2), 362-376, 1983.
8. Jordan, T.L., "Conjugate Gradient Preconditioners for Vector and Parallel Processors", *Elliptic Problem Solvers II*, Academic Press Inc., 127-139, 1984.
9. Kuiper, L.K., "A Comparison of the Incomplete Cholesky-Conjugate Gradient Method With the Strongly Implicit Method as Applied to the Solution of Two-Dimensional Groundwater Flow Equations", *Water Resources Research*, 17(4), 1082-1086, August 1981.
10. McDonald, M.G. and Harbaugh, A.W., "A Modular Three-Dimensional Finite-Difference Ground-Water Flow Model", Open-File Report 83-875, U.S. Geological Survey, 1984.
11. Morrow, D. and Holter, B., "A Vectorized Polynomial Preconditioned Conjugate Gradient Solver Package for the USGS 3-D Ground-Water Model", Proceedings of the ASCE Water Resources Conference, Norfolk, VA, June 1988.
12. Oppe, T., et.al., "NSPCG User's Guide, Version 1.0", Center for Numerical Analysis, The University of Texas at Austin, April 1988.
13. Saad, Y., "Practical Use of Polynomial Preconditionings for the Conjugate Gradient Method", *SIAM J. Sci. Stat. Comput.*, 6(4), 865-881, 1985.
14. Scandrett, C. "A Comparison of Three Iterative Techniques in Solving Symmetric Systems of Linear Equations on a CYBER 205' Supercomputer Computations Research Institute, Florida State Univ., Report # FSU-SCRI-87-44, August 1987.
15. Van Der Vorst, H., "A Vectorizable Variant of Some ICCG Methods", *SIAM J. Sci. Stat. Comput.*, 3(3), 350-356, 1982.
16. Weinstein, H.G., Stone, H.L., and Kwan, T.V. "Iterative Procedure for Solution of Systems of Parabolic and Elliptic Equations in Three Dimensions", *Industrial and Engineering Chemistry Fundamentals*, 8(2), 281-287, 1969.

Decimation of Triangle Meshes

William J. Schroeder

General Electric Company
Scenectady, NY

1.0 INTRODUCTION

The polygon remains a popular graphics primitive for computer graphics application. Besides having a simple representation, computer rendering of polygons is widely supported by commercial graphics hardware and software. However, because the polygon is linear, often thousands or millions of primitives are required to capture the details of complex geometry. Models of this size are generally not practical since rendering speeds and memory requirements are proportional to the number of polygons. Consequently applications that generate large polygonal meshes often use domain-specific knowledge to reduce model size. There remain algorithms, however, where domain-specific reduction techniques are not generally available or appropriate.

One algorithm that generates many polygons is *marching cubes*. *Marching cubes* is a brute force surface construction algorithm that extracts isodensity surfaces from volume data, producing from one to five triangles within voxels that contain the surface. Although originally developed for medical applications, *marching cubes* has found more frequent use in scientific visualization where the size of the volume data sets are much smaller than those found in medical applications. A large computational fluid dynamics volume could have a finite difference grid size of order 100 by 100 by 100, while a typical medical computed tomography or magnetic resonance scanner produces over 100 slices at a resolution of 256 by 256 or 512 by 512 pixels each. Industrial computed tomography, used for inspection and analysis, has even greater resolution, varying from 512 by 512 to 1024 by 1024 pixels. For these sampled data sets, isosurface extraction using *marching cubes* can produce from 500k to 2,000k triangles. Even today's graphics workstations have trouble storing and rendering models of this size.

Other sampling devices can produce large polygonal models: range cameras, digital elevation data, and satellite data. The sampling resolution of these devices is also improving, resulting in model sizes that rival those obtained from medical scanners.

This paper describes an application independent algorithm that uses local operations on geometry and topology to reduce the number of triangles in a triangle mesh. Although our implementation is for the triangle mesh, it can be directly applied to the more general polygon mesh. After describing other work related to model creation from sampled data, we describe the triangle decimation

process and its implementation. Results from two different geometric modeling applications illustrate the strengths of the algorithm.

2.0 THE DECIMATION ALGORITHM

The goal of the decimation algorithm is to reduce the total number of triangles in a triangle mesh, preserving the original topology and a good approximation to the original geometry.

2.1 OVERVIEW

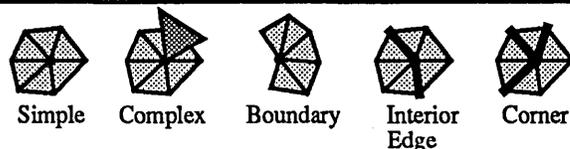
The decimation algorithm is simple. Multiple passes are made over all vertices in the mesh. During a pass, each vertex is a candidate for removal and, if it meets the specified decimation criteria, the vertex and all triangles that use the vertex are deleted. The resulting hole in the mesh is patched by forming a local triangulation. The vertex removal process repeats, with possible adjustment of the decimation criteria, until some termination condition is met. Usually the termination criterion is specified as a percent reduction of the original mesh (or equivalent), or as some maximum decimation value. The three steps of the algorithm are:

1. characterize the local vertex geometry and topology,
2. evaluate the decimation criteria, and
3. triangulate the resulting hole.

2.2 CHARACTERIZING LOCAL GEOMETRY / TOPOLOGY

The first step of the decimation algorithm characterizes the local geometry and topology for a given vertex. The outcome of this process determines whether the vertex is a potential candidate for deletion, and if it is, which criteria to use.

Each vertex may be assigned one of five possible classifications: simple, complex, boundary, interior edge, or corner vertex. Examples of each type are shown in the figure below.



A simple vertex is surrounded by a complete cycle of

triangles, and each edge that uses the vertex is used by exactly two triangles. If the edge is not used by two triangles, or if the vertex is used by a triangle not in the cycle of triangles, then the vertex is complex. These are non-manifold cases.

A vertex that is on the boundary of a mesh, i.e., within a semi-cycle of triangles, is a boundary vertex.

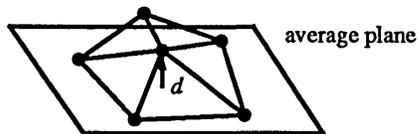
A simple vertex can be further classified as an interior edge or corner vertex. These classifications are based on the local mesh geometry. If the dihedral angle between two adjacent triangles is greater than a specified *feature angle*, then a *feature edge* exists. When a vertex is used by two feature edges, the vertex is an interior edge vertex. If one or three or more feature edges use the vertex, the vertex is classified a corner vertex.

Complex vertices are not deleted from the mesh. All other vertices become candidates for deletion.

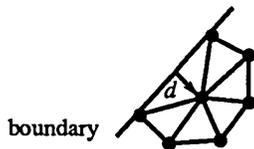
2.3 EVALUATING THE DECIMATION CRITERIA

The characterization step produces an ordered loop of vertices and triangles that use the candidate vertex. The evaluation step determines whether the triangles forming the loop can be deleted and replaced by another triangulation exclusive of the original vertex. Although the fundamental decimation criterion we use is based on vertex distance to plane or vertex distance to edge, others can be applied.

Simple vertices use the distance to plane criterion (see figure below). If the vertex is within the specified distance to the average plane it may be deleted. Otherwise it is retained.



Boundary and interior edge vertices use the distance to edge criterion (figure below). In this case, the algorithm determines the distance to the line defined by the two vertices creating the boundary or feature edge. If the distance to the line is less than d , the vertex can be deleted.



It is not always desirable to retain feature edges. For example, meshes may contain areas of relatively small triangles with large feature angles, contributing relatively little to the geometric approximation. Or, the small triangles may be the result of “noise” in the original mesh. In these situations, corner vertices, which are usually not deleted, and interior edge vertices, which are evaluated using the distance to edge criterion, may be evaluated using the distance to plane criterion. We call this edge preservation, a user specifiable parameter.

If a vertex can be eliminated, the loop created by removing the triangles using the vertex must be triangulated. For

interior edge vertices, the original loop must be split into two halves, with the split line connecting the vertices forming the feature edge. If the loop can be split in this way, i.e., so that resulting two loops do not overlap, then the loop is split and each piece is triangulated separately.

2.4 TRIANGULATION

Deleting a vertex and its associated triangles creates one (simple or boundary vertex) or two loops (interior edge vertex). Within each loop a triangulation must be created whose triangles are non-intersecting and non-degenerate. In addition, it is desirable to create triangles with good aspect ratio and that approximate the original loop as closely as possible.

In general it is not possible to use a two-dimensional algorithm to construct the triangulation, since the loop is usually non-planar. In addition, there are two important characteristics of the loop that can be used to advantage. First, if a loop cannot be triangulated, the vertex generating the loop need not be removed. Second, since every loop is star-shaped, triangulation schemes based on recursive loop splitting are effective. The next section describes one such scheme.

Once the triangulation is complete, the original vertex and its cycle of triangles are deleted. From the Euler relation it follows that removal of a simple, corner, or interior edge vertex reduces the mesh by precisely two triangles. If a boundary vertex is deleted then the mesh is reduced by precisely one triangle.

3.0 IMPLEMENTATION

3.1 DATA STRUCTURES

The data structure must contain at least two pieces of information: the geometry, or coordinates, of each vertex, and the definition of each triangle in terms of its three vertices. In addition, because ordered lists of triangles surrounding a vertex are frequently required, it is desirable to maintain a list of the triangles that use each vertex.

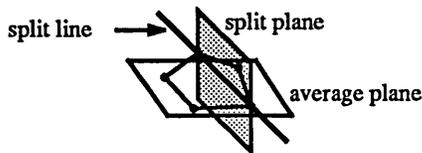
Although data structures such as Weiler’s radial edge or Baumgart’s winged-edge data structure can represent this information, our implementation uses a space-efficient vertex-triangle hierarchical ring structure. This data structure contains hierarchical pointers from the triangles down to the vertices, and pointers from the vertices back up to the triangles using the vertex. Taken together these pointers form a ring relationship. Our implementation uses three lists: a list of vertex coordinates, a list of triangle definitions, and another list of lists of triangles using each vertex. Edge definitions are not explicit, instead edges are implicitly defined as ordered vertex pairs in the triangle definition.

3.2 TRIANGULATION

Although other triangulation schemes can be used, we chose a recursive loop splitting procedure. Each loop to be triangulated is divided into two halves. The division is along a line (i.e., the split line) defined from two non-neighboring vertices in the loop. Each new loop is divided again, until

only three vertices remain in each loop. A loop of three vertices forms a triangle, that may be added to the mesh, and terminates the recursion process.

Because the loop is non-planar and star-shaped, the loop split is evaluated using a split plane. The split plane, as shown in the figure below, is the plane orthogonal to the average plane that contains the split line. In order to determine whether the split forms two non-overlapping loops, the split plane is used for a half-space comparison. That is, if every point in a candidate loop is on one side of the split plane, then the two loops do not overlap and the split plane is acceptable. Of course, it is easy to create examples where this algorithm will fail to produce a successful split. In such cases we simply indicate a failure of the triangulation process, and do not remove the vertex or surrounding triangle from the mesh.



Typically, however, each loop may be split in more than one way. In this case, the best splitting plane must be selected. Although many possible measures are available, we have been successful using a criterion based on aspect ratio. The aspect ratio is defined as the minimum distance of the loop vertices to the split plane, divided by the length of the split line. The best splitting plane is the one that yields the maximum aspect ratio. Constraining this ratio to be greater than a specified value, e.g., 0.1, produces acceptable

meshes.

Certain special cases may occur during the triangulation process. Repeated decimation may produce a simple closed surface such as a tetrahedron. Eliminating a vertex in this case would modify the topology of the mesh. Another special case occurs when "tunnels" or topological holes are present in the mesh. The tunnel may eventually be reduced to a triangle in cross section. Eliminating a vertex from the tunnel boundary then eliminates the tunnel and creates a non-manifold situation.

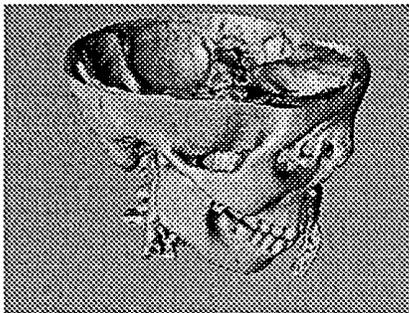
These cases are treated during the triangulation process. As new triangles are created, checks are made to insure that duplicate triangles and triangle edges are not created. This preserves the topology of the original mesh, since new connections to other parts of the mesh cannot occur.

4.0 RESULTS

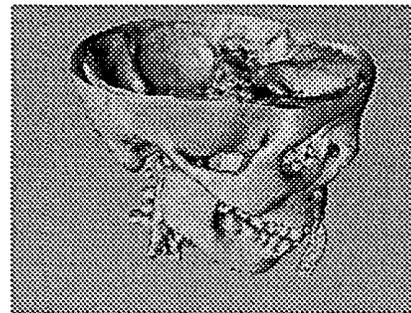
Two different applications illustrate the triangle decimation algorithm. Although each application uses a different scheme to create an initial mesh, all results were produced with the same decimation algorithm.

4.1 VOLUME MODELING

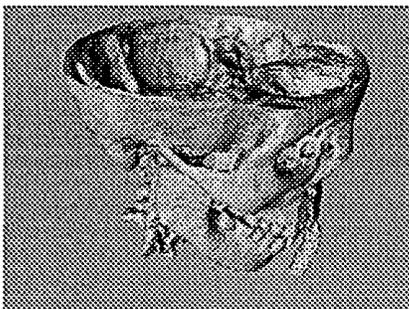
The first application applies the decimation algorithm to isosurfaces created from medical and industrial computed tomography scanners. *Marching cubes* was run on a 256 by 256 pixel by 93 slice study. Over 560,000 triangles were required to model the bone surface. Earlier work reported a triangle reduction strategy that used averaging to reduce the



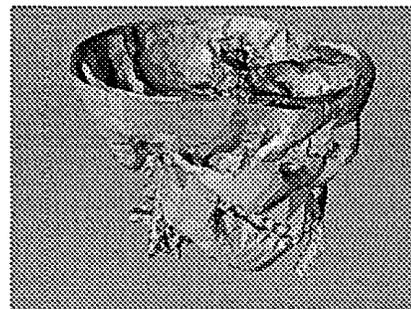
Full Resolution
(569K Gouraud shaded triangles)



75% decimated
(142K Gouraud shaded triangles)



75% decimated
(142K flat shaded triangles)



90% decimated
(57K flat shaded triangles)

number of triangles on this same data set. Unfortunately, averaging applies uniformly to the entire data set, blurring high frequency features. The first set of figures shows the resulting bone isosurfaces for 0%, 75%, and 90% decimation, using a decimation threshold of 1/5 the voxel dimension. The next pair of figures shows decimation results for an industrial CT data set comprising 300 slices, 512 by 512, the largest we have processed to date. The isosurface created from the original blade data contains 1.7 million triangles. In fact, we could not render the original model because we exceeded the swap space on our graphics hardware. Even after decimating 90% of the triangles, the serial number on the blade dovetail is still evident.

4.2 TERRAIN MODELING

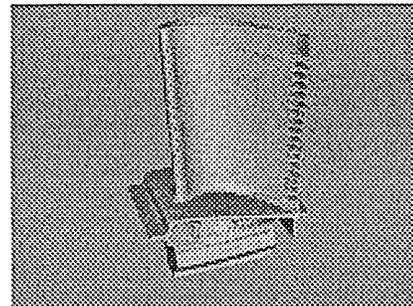
We applied the decimation algorithm to two digital elevation data sets: Honolulu, Hawaii and the Mariner Valley on Mars. In both examples we generated an initial mesh by creating two triangles for each uniform quadrilateral element in the sampled data. The Honolulu example illustrates the polygon savings for models that have large flat areas. First we applied a decimation threshold of zero, eliminating over 30% of the co-planar triangles. Increasing the threshold removed 90% of the triangles. The next set of four figures shows the resulting 30% and 90% triangulations. Notice the transitions from large flat areas to fine detail around the shore line.

The Mars example is an appropriate test because we had access to sub-sampled resolution data that could be compared with the decimated models. The data represents the western end of the Mariner Valley and is about 1000km by 500km on a side. The last set of figures compares the shaded and wireframe models obtained via sub-sampling and decimation. The original model was 480 by 288 samples. The sub-sampled data was 240 by 144. After a 77% reduction, the decimated model contains fewer triangles, yet shows more fine detail around the ridges.

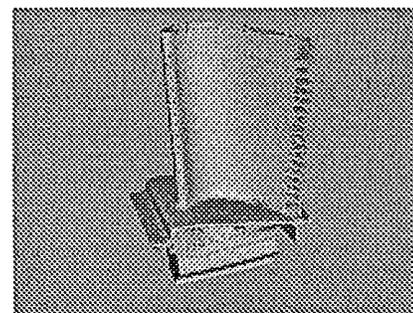
5.0 REFERENCES

- [1] Baumgart, B. G., "Geometric Modeling for Computer Vision," Ph.D. Dissertation, Stanford University, August 1974.
- [2] Bloomenthal, J., "Polygonalization of Implicit Surfaces," *Computer Aided Geometric Design*, Vol. 5, pp. 341-355, 1988.
- [3] Cline, H. E., Lorensen, W. E., Ludke, S., Crawford, C. R., and Teeter, B. C., "Two Algorithms for the Three Dimensional Construction of Tomograms," *Medical Physics*, Vol. 15, No. 3, pp. 320-327, June 1988.
- [4] DeHaemer, M. J., Jr. and Zyda, M. J., "Simplification of Objects Rendered by Polygonal Approximations," *Computers & Graphics*, Vol. 15, No. 2, pp 175-184, 1992.
- [5] Dunham, J. G., "Optimum Uniform Piecewise Linear Approximation of Planar Curves," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 1, pp. 67-75, January 1986.
- [6] Finnigan, P., Hathaway, A., and Lorensen, W., "Merging CAT and FEM," *Mechanical Engineering*, Vol. 112, No. 7, pp. 32-38, July 1990.
- [7] Fowler, R. J. and Little, J. J., "Automatic Extraction of Irregular Network Digital Terrain Models," *Computer Graphics*, Vol. 13, No. 2, pp. 199-207, August 1979.
- [8] Ihm, I. and Naylor, B., "Piecewise Linear Approximations of Digitized Space Curves with Applications," in *Scientific Visualization of Physical Phenomena*, pp. 545-569, Springer-Verlag, June 1991.

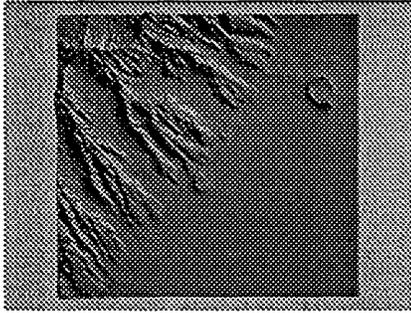
- [9] Kalvin, A. D., Cutting, C. B., Haddad, B., and Noz, M. E., "Constructing Topologically Connected Surfaces for the Comprehensive Analysis of 3D Medical Structures," *SPIE Image Processing*, Vol. 1445, pp. 247-258, 1991.
- [10] Lorensen, W. E. and Cline, H. E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, Vol. 21, No. 3, pp. 163-169, July 1987.
- [11] Miller, J. V., Breen, D. E., Lorensen, W. E., O'Bara, R. M., and Wozny, M. J., "Geometrically Deformed Models: A Method for Extracting Closed Geometric Models from Volume Data," *Computer Graphics*, Vol. 25, No. 3, July 1991.
- [12] Preparata, F. P. and Shamos, M. I., *Computational Geometry*, Springer-Verlag, 1985.
- [13] Schmitt, F. J., Barsky, B. A., and Du, W., "An Adaptive Sub-division Method for Surface-Fitting from Sampled Data," *Computer Graphics*, Vol. 20, No. 4, pp. 179-188, August 1986.
- [14] Schroeder, W. J., "Geometric Triangulations: With Application to Fully Automatic 3D Mesh Generation," PhD Dissertation, Rensselaer Polytechnic Institute, May 1991.
- [15] Terzopoulos, D. and Fleischer, K., "Deformable Models," *The Visual Computer*, Vol. 4, pp. 306-311, 1988.
- [16] Turk, G., "Re-Tiling of Polygonal Surfaces," *Computer Graphics*, Vol. 26, No. 3, July 1992.
- [17] Weiler, K., "Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments," *IEEE Computer Graphics and Applications*, Vol. 5, No. 1, pp. 21-40, January 1985.



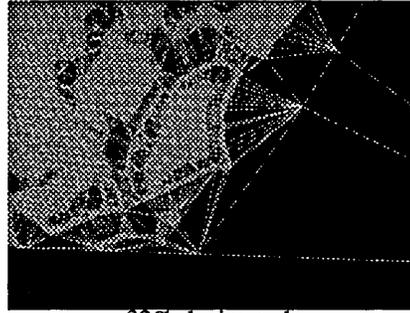
75% decimated
(425K flat shaded triangles)



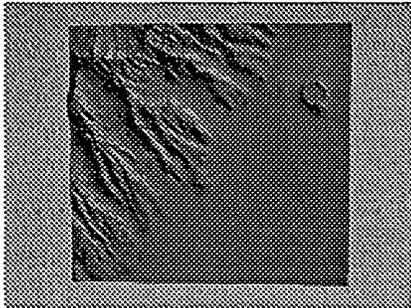
90% decimated
(170K flat shaded triangles)



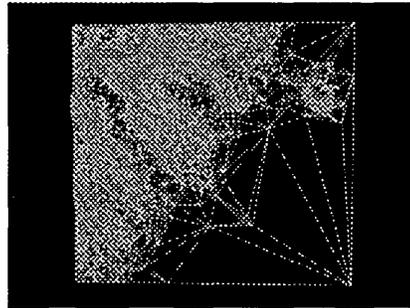
32% decimated
(276K flat shaded triangles)



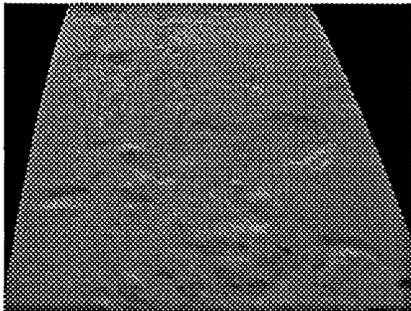
32% decimated
(shore line detail)



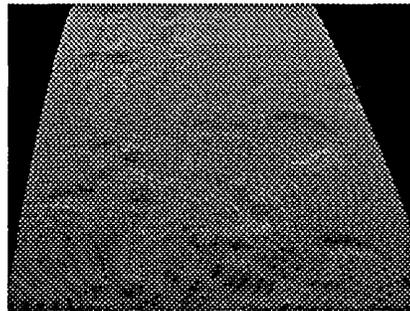
90% decimated
(40K Gouraud shaded triangles)



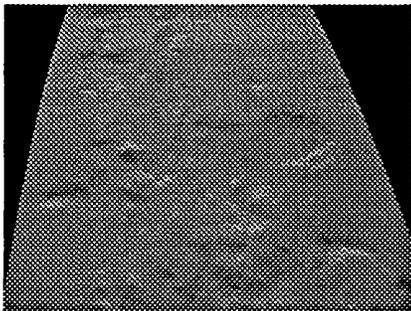
90% decimated
(40K wireframe)



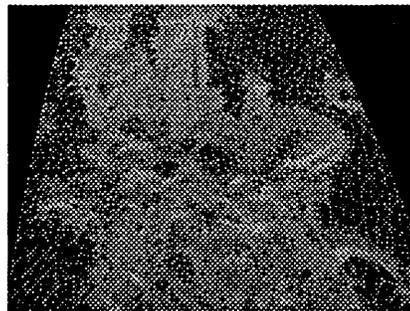
Sub-sampled
(68K Gouraud shaded triangles)



Sub-sampled
(68K wireframes)



77% decimated
(62K Gouraud shaded triangles)



77% decimated
(62K wireframes)

VISUALIZATION OF VOLCANIC ASH CLOUDS

Mitchell Roth

Arctic Region Supercomputing Center
University of Alaska
Fairbanks, AK 99775
roth@acad5.alaska.edu

Rick Guritz

Alaska Synthetic Aperture Radar Facility
University of Alaska
Fairbanks, AK 99775
rguritz@iias.images.alaska.edu

ABSTRACT

Ash clouds resulting from volcanic eruptions pose a serious hazard to aviation safety. In Alaska alone, there are over 40 active volcanoes whose eruptions may affect more than 40,000 flights using the great circle polar routes each year. Anchorage International Airport, a hub for flights refueling between Europe and Asia, has been closed due to volcanic ash on several occasions in recent years. The clouds are especially problematic because they are invisible to radar and nearly impossible to distinguish from weather clouds. The Arctic Region Supercomputing Center and the Alaska Volcano Observatory have used AVS to develop a system for predicting and visualizing the movement of volcanic ash clouds when an eruption occurs. Based on eruption parameters obtained from geophysical instruments and meteorological data, a model was developed to predict the movement of the ash particles over a 72 hour period. The output from the model is combined with a digital elevation model to produce a realistic view of the ash cloud, which may be examined interactively from any desired point of view at any time during the prediction period. This paper describes the visualization techniques employed in the system and includes a video animation of the Mount Redoubt eruption on December 15 that caused complete engine failure on a 747 passenger jet when it entered the ash cloud.

1. Introduction

Alaska is situated on the northern boundary of the Pacific Rim. Home to the highest mountains in North America, the mountain ranges of Alaska contain over 50 active volcanoes. In the past 200 years most of Alaska's active volcanoes have erupted at least once. Alaska is a polar crossroads where aircraft traverse the great circle airways between Asia, Europe and North America. Volcanic eruptions in Alaska and the resulting airborne ash clouds pose a significant hazard to the more than 40,000 transpolar flights each year.

The ash clouds created by volcanic eruptions are invisible to radar and are often concealed by weather clouds. This paper describes a system developed by the Alaska Volcano Observatory and the Arctic Region Supercomputing Center for predicting the movement of ash clouds. Using meteorological and geophysical data from volcanic eruptions, a supercomputer model provides predictions of ash cloud movements for up to 72 hours. The AVS visualization system is used to control the execution of the ash cloud model and to display the model output in three dimensional form showing the location of the ash cloud over a digital terrain model.

Eruptions of Mount Redoubt on the morning of December 15, 1989, sent ash particles more than 40,000 feet into the atmosphere. On the same day, a Boeing 747 experienced complete engine failure when it penetrated the ash cloud. The ash cloud prediction system was used to simulate this eruption and to produce an animated flyby of Mount Redoubt during a 12 hour period of the December 15 eruptions including the encounter of the 747 jetliner with the ash cloud. The animation combines the motion of the viewer with the time evolution of the ash cloud above a digital terrain model.

2. Ash Plume Model

The ash cloud visualization is based on the output of a model developed by Hiroshi Tanaka of the Geophysical Institute of the University of Alaska and Tsukuba University, Japan. Using meteorological data and eruption parameters for input, the model predicts the density of volcanic ash particles in the atmosphere as a function of time. The three dimensional Lagrangian form of the diffusion equation is employed to model particle diffusion, taking into account the size distribution of the ash particles and gravitational settling described by Stokes' law. Details of the model are given in Tanaka [2] [3].

The meteorological data required are winds in the upper atmosphere. These are obtained from UCAR Unidata in NetCDF format. Unidata winds are interpolated to observed conditions on 12 hour intervals. Global circulation models are used to provide up to 72 hour predictions at 6 hour intervals.

The eruption parameters for the model include the geographical location of the volcano, the time and duration of the event, altitude of the plume, particle density, and particle density distribution.

The model has been implemented in both Sun and Cray environments. An AVS module was created for the Cray version which allows the model to be controlled interactively from an AVS network. In this version, the AVS module executes the model on the Cray, reads the resulting output file and creates a 3D AVS scalar field representing the particle densities at each time step.

The raw output from the model for each time step consists of a list of particles with an (x,y,z) coordinate for each particle. The AVS module reads the particle data and increments the particle counts for the cells formed by an array indexed over (x,y,z). We chose a resolution of 150 x 150 x 50 for the particle density array, which equals 1.1 million data points at each solution point in time. For the video animation, we chose to run the model with a time step of 5 minutes. For 13 hours of simulated time, the model produced 162 plumes, amounting to approximately 730 MB of integer valued volume data.

3. Ash Cloud Visualization

The ash cloud is rendered as an isosurface with a brown color approximating volcanic ash. The rendering obtained through this technique gives the viewer a visual effect showing the boundaries of the ash cloud. Details of the cloud shape are highlighted through lighting effects and, when viewed on a computer workstation, the resulting geometry can be manipulated interactively to view the ash cloud from any desired direction.

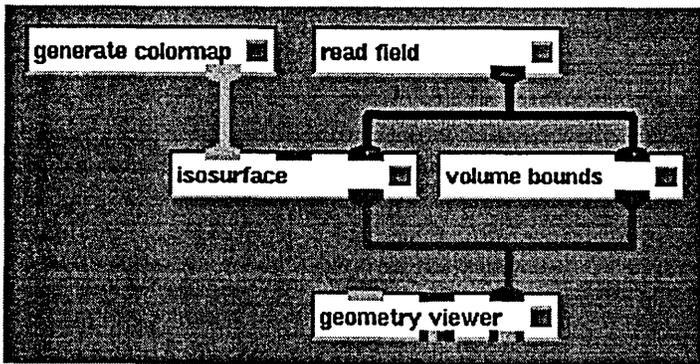


Figure 1. AVS plume isosurface network

At any point in time, the particle densities in the ash cloud are represented by the values in a 150 x 150 x 50 element integer array. The limits of the cloud may be observed using the isosurface module in the network shown in Figure 1 with the isosurface level set equal to 1. As the cloud disperses, the particle concentrations in the array decrease and holes and isolated cells begin to appear in the isosurface around the edges of the plume where the density is between zero and one particle. These effects are readily apparent in the plume shown in Figure 2 and are especially noticeable in a time animation of the plume evolution. To create a more uniform cloud for the video animation, without increasing the overall particle counts, the density array was low pass filtered by an inverse square kernel before creating the isosurface. An example of a filtered plume created by this technique is shown in Figure 8.

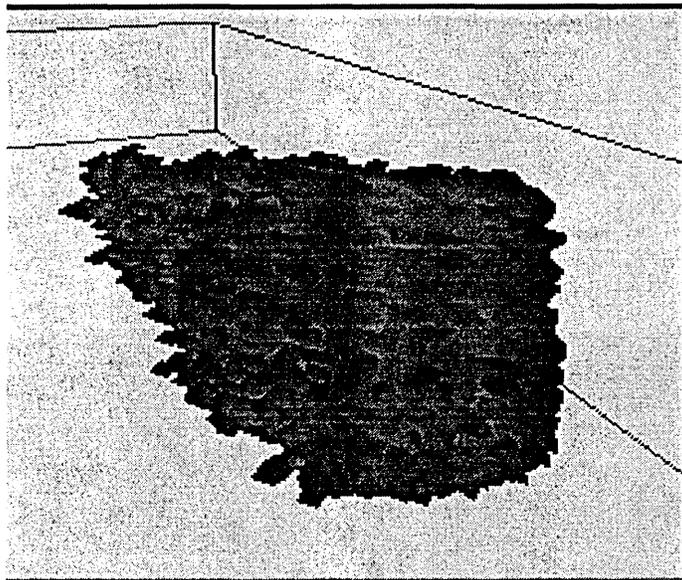


Figure 2. Unfiltered plume data displayed as an isosurface.

4. Plume Animation

The plume model must use time steps of 5 minutes or greater due to limitations of the model. Plumes that are generated at 5 minute intervals may be displayed to create a flip chart animation of the time evolution of the cloud. However, the changes in the plume over a 5 minute interval can be fairly dramatic and shorter time intervals are required to create the effect of a smoothly evolving cloud. To accomplish this without generating additional plume volumes we interpolate between successive plume volumes. Using the field math module, we implemented linear interpolation between plume volumes in the network shown in Figure 3.

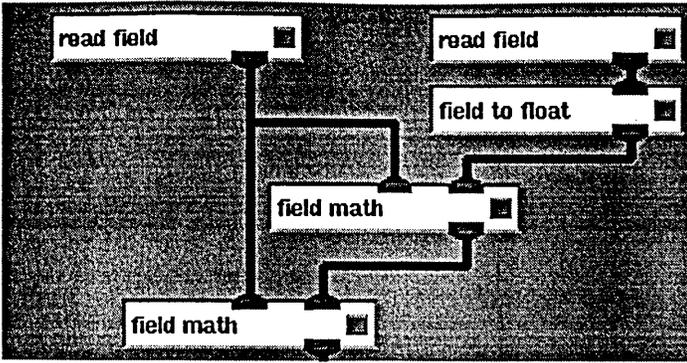


Figure 3. AVS interpolation network.

The linear interpolation formula is:

$$P(t) = P_i + (P_{i+1} - P_i) t, \quad 0 \leq t \leq 1, \quad (1)$$

where P_i is the plume volume at time step i and t is time. The difference term in (1) is formed in the upper **field math** module. The lower **field math** module sums its inputs. Normally, a separate **field math** module would be required to perform the multiplication by t . However, it is possible to multiply the output port of a **field math** module by a constant value when the network is executed from a CLI script and this is the approach we used to create the video animation of the eruption. If it is desired to set the interpolation parameter interactively, it is necessary to insert a third **field math** module to perform the multiplication on the output of the upper module. This can be an extremely effective device for producing smooth time animation of discrete data sets in conjunction with the AVS Animator module.

One additional animation effect was introduced to improve the appearance of the plume at the beginning of the eruption. The plume model assumes that the plume reaches the specified eruption height instantaneously. Thus, the plume model for the first time step produces a cylindrical isosurface of uniform particle densities above the site of the eruption. To create the appearance of a cloud initially rising from the ground, we defined an artificial plume for time 0. The time 0 plume isosurface consists of an inverted cone of *negative* plume densities centered over the eruption coordinates. The top of the plume volume contains the most negative density values. When this plume volume is interpolated with the model plume from time step 1, the resulting plume rises from the ground and reaches the full eruption height at $t=1$.

5. Terrain Visualization

The geographical region for this visualization study is an area in south-central Alaska which lies between 141° - 160° west longitude and 60° - 67° north latitude. Features in the study area include Mount Redoubt, Mount McKinley, the Alaska Range, Cook Inlet and the cities of Anchorage, Fairbanks, and Valdez.

The corners of the region define a Cartesian coordinate system and the extents of the volcano plume data must be adjusted to obtain the correct registration of the plume data in relation to the terrain. The terrain features are based on topographic data obtained from the US Geological Survey with a grid spacing of approximately 90 meters. This grid was much too large to process at the original resolution and was downsized to a 1426×1051 element array of terrain elevations, which corresponds to a grid size of approximately 1/2 mile. As shown in Figure 4, the terrain data were read in field format and were converted to a geometry using the **field to mesh** module. We included a **downsize** module

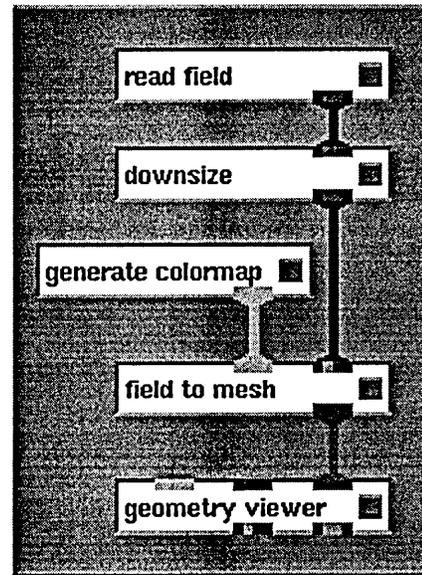


Figure 4. AVS terrain network.

ahead of **field to mesh** because even the 1426×1051 terrain exceeded available memory on all but our largest machines. For prototyping and animation design, we typically downsized by factors of 2 to 4 in order to speed up the terrain rendering.

The colors of the terrain are set in the **generate colormap** module according to elevation of the terrain and were chosen to approximate ground cover during the fall season in Alaska. The vertical scale of the terrain was exaggerated by a factor of 60 to better emphasize the topography.

The resulting terrain is shown Figure 5 with labels that were added using image processing techniques. To create the global zoom sequence in the introduction to the video animation, this image was used as a texture map that was overlaid onto a lower resolution terrain model for the entire state of Alaska. This technique also allowed the study area to be highlighted in such a way as to create a smooth transition into the animation sequence.

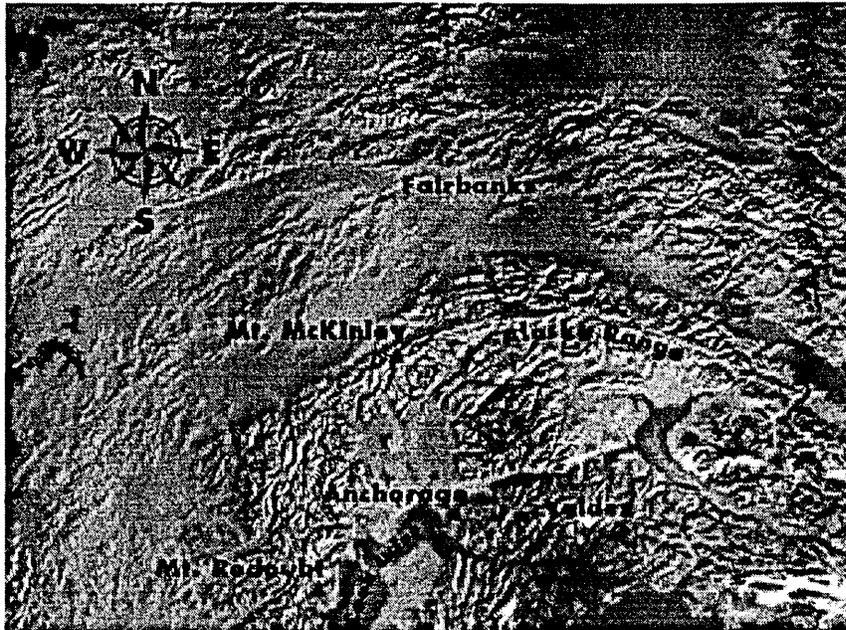


Figure 5. Study area with texture mapped labels.

6. Flight Path Visualization

The flight path of the jetliner in the animation was produced by applying the **tube** module to a polyline geometry obtained through **read geom** as shown in Figure 6. The animation of the tube was performed by a simple program which takes as its input a time dependent cubic spline. The program evaluates the spline at specified points to create a polyline geometry for **read geom**. Each new point added to the polyline causes a new segment of the flight path to be generated by **tube**. In Figure 7, the entire flight path spline function is displayed. Four separate **tube** modules were employed to allow the flight path segments to be colored green, red, yellow, and green during the engine failure and restart sequence.

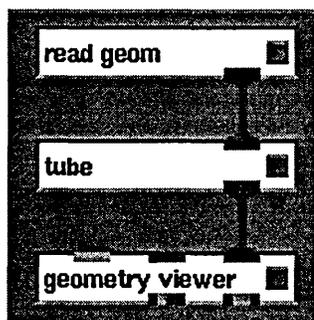


Figure 6. AVS flight path network.

The path of the jetliner is based on flight recorder data obtained from the Federal Aviation Administration. The flight path was modeled using three dimensional time dependent cubic splines. The technique for deriving and manipulating the spline functions is so powerful that we created a new module called the **Spline Animator** for this purpose. The details of this module are described in a paper by Astley [1]. A similar technique is used to control the camera motion required for the flyby in the video animation.

By combining the jetliner flight path with the animation of the ash plume described earlier, a simulated encounter of the jet with the ash cloud can be studied in an animated sequence. The resulting simulation provides valuable information about the accuracy of the plume model. Because ash plumes are invisible to radar and may be hidden from satellites by weather clouds, it is often very difficult to determine the exact position and extent of an ash cloud from direct observations. However, when a jetliner penetrates an ash cloud, the effects are immediate and unmistakable and the aircraft position is usually known rather accurately. This was the case during the December 15 encounter.

Thus, by comparing the intersection point of the jetliner flight path with the plume model to the point of intersection with the actual plume, one can determine if the leading edge of the plume model is in the correct position. Both the plume model and the flight path must be correctly co-registered to the terrain data in order to perform such a test. Using standard transformations between latitude-longitude and x-y coordinates for the terrain, we calculated the appropriate coordinate transformations for the plume model and jet flight path. The first time the animation

was run we were quite amazed to observe the flight path turn red, denoting engine failure, at precisely the point where the flight path encountered the leading edge of the modeled plume.

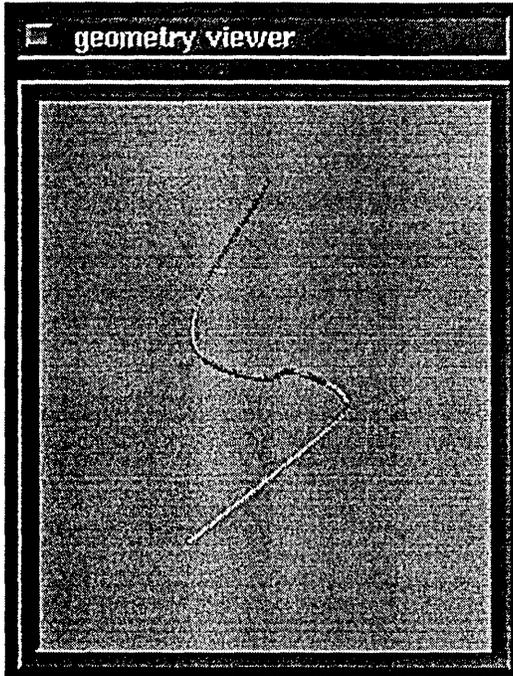


Figure 7. Flight path geometry created by spline animator.

Apparently this was one of those rare times when we got everything right. The fact that the aircraft position is well known at all times, and that it encounters the ash cloud at the correct time and place lends strong support for the correctness of the model. Figure 8 shows a frame from the video animation at the time when the jetliner landed in Anchorage. The ash cloud in this image is drifting from left to right and away from the viewer.

7. Satellite Image Comparison

Ash clouds can often be detected in AVHRR satellite images. For the December 15 events, only one image recorded at 1:27pm AST was available. At the time of this image most of the study area was blanketed by clouds. Nevertheless, certain atmospheric features become visible when the image is subjected to enhancement, as shown in Figure 9. A north-south frontal system is moving northeasterly from the left side of the image. To the left of the front, the sky is generally clear and surface features are visible. To the right of the front, the sky is completely overcast and no surface features are visible. One prominent cloud feature is a mountain wave created by Mount McKinley. This shows up as a long plume moving in a north-northeasterly direction from Mount McKinley and is consistent with upper altitude winds on this date.

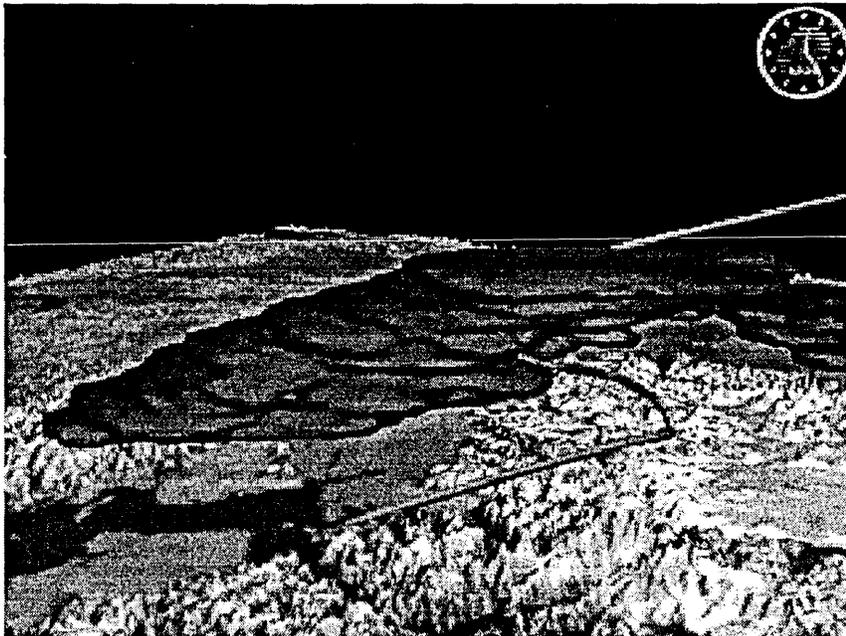


Figure 8. Flight path through ash plume.



Figure 9. Enhanced AVHRR satellite image taken at 1:27pm AST.

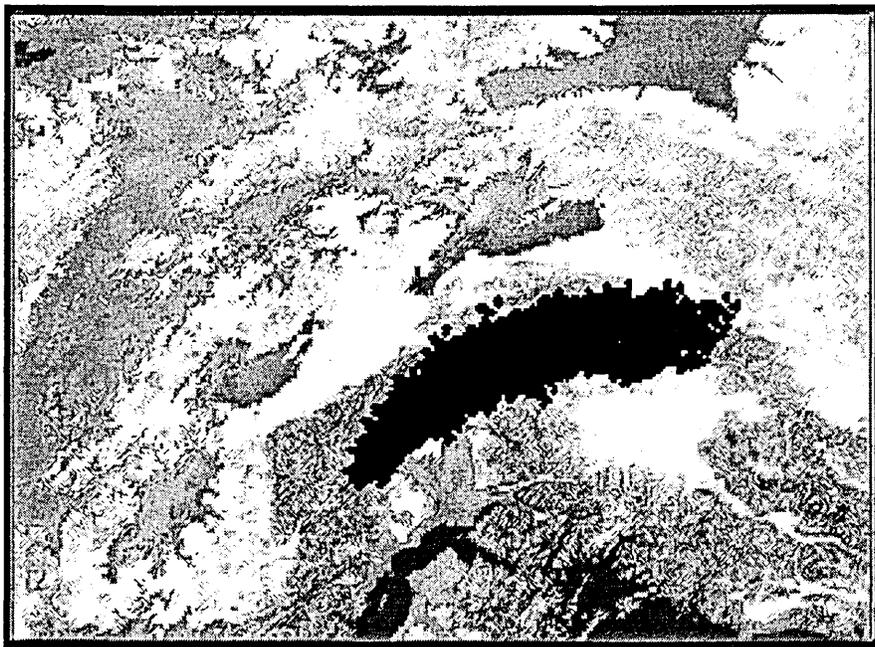


Figure 10. Position of simulated plume at 1:30pm AST.

The satellite image was enhanced in a manner which causes ash clouds to appear black. There is clearly a dark plume extending from Mount Redoubt near the lower edge of the image to the northeast and ending in the vicinity of Anchorage. The size of

this plume indicates that it is less than an hour old. Thus, it could not be the source of the plume which the jet encountered approximately 2 hours before this image was taken.

There are additional black areas in the upper right quadrant of the image which are believed to have originated with the 10:15am eruption. These are the clouds which the jet is believed to have penetrated approximately 2 hours before this image was taken. The image has been annotated with the jetliner flight path entering in the top center of the image and proceeding from top to bottom in the center of the image. The ash cloud encounter occurred at the point where the flight path reverses course to the north and east. However, the satellite image does not show any ash clouds remaining in the vicinity of the flight path by the time of this image.

When the satellite image is compared with the plume model for the same time period, shown at approximately the same scale in Figure 10, a difference in the size of the ash cloud is readily apparent. While the leading edge of the simulated plume stretching to the northeast is located in approximately the same position as the dark clouds in the satellite image, the cloud from the simulated plume is much longer. The length of the simulated plume is controlled by the duration of the eruption, which was 40 minutes.

Two explanations for the differences have been proposed. The first is that the length of the eruption was determined from seismic data. Seismicity does not necessarily imply the emission of ash and therefore the actual ash emission time may have been less than 40 minutes. The second possibility is that the trailing end of the ash cloud may be invisible in the satellite image due to cloud cover. It is worth noting that the ash cloud signatures in this satellite image are extremely weak compared to cloudless images. In studies of the few other eruptions where clear images were available, the ash clouds are unmistakable in the satellite image and the model showed excellent agreement with the satellite data.

8. Flyby Animation

One of the great advantages of three dimensional animation methods is the ability to move around in a simulated 3D environment interactively, or to create a programmed tour or flyby. For the ash cloud visualization, we wanted to follow the moving ash clouds in relation to the terrain and to look at them from different distances and different directions. AVS allows fully interactive manipulation of the views of the ash cloud, but the rendering process is too slow (minutes per frame) to allow for realtime animation. For this reason we decided to create a flyby of the events on December 15 by combining camera animation with the time dependent animations of the plume and jet flight path.

In our initial attempts, we used the AVS Animator module and found that it worked well for the linear time dependent portions of the animation. However, the camera animation was a different story altogether because camera motion in a flyby situation is seldom linear. When we attempted to use the Animator in its "smooth" mode, we found it was only possible to control the camera accurately when we introduced dozens of key frames in order to tightly constrain the frame acceleration introduced by the

sinusoidal interpolation technique used in the Animator. Having to use a large number of key frames makes it very time consuming to construct a flight path, because changing the flight path requires all the key frames near the change to be modified in a consistent manner. We eventually realized that a minor extension to the flight path spline algorithm already developed could easily provide the desired camera coordinates. In essence, the camera coordinates could be determined from the flight path of the viewer in the same manner that we computed the flight path of the jetliner.

The first version of the Spline Animator used a text file for input which contained the key frame information. The output was a sequence of camera coordinates which were edited into a CLI script which could be played back interactively or in batch mode. In this first effort we were able to define a flight path using about a half dozen key frames in place of the dozens required by the AVS Animator and the smoothness and predictability of results were far superior. After the video animation of the eruption visualization was completed, a second version of the Spline Animator was created with a Motif interface and a module is now available for use in AVS networks. For more information about this module, the reader is referred to Astley [1].

9. Conclusions

An ash plume modeling and prediction system has been developed using AVS for visualization and a Cray supercomputer for model computations. A simulation of the December 15 encounter with ash clouds from Mount Redoubt by a jetliner provides strong support for the accuracy of the model. Although the satellite data for this event are relatively limited, agreement of the model with satellite data for other events is very good. The animated visualization of the eruption which was produced using AVS demonstrates that AVS is an extremely effective tool for developing visualizations and animations. The Spline Animator module was developed to perform flybys and may be used to construct animated curves or flight paths in 3D.

10. Acknowledgments

The eruption visualization of Mount Redoubt Volcano was produced in a collaborative effort by the University of Alaska Geophysical Institute and the Arctic Region Supercomputing Center. Special thanks are due Ken Dean of the Alaska Volcano Observatory and to Mark Astley and Greg Johnson of ARSC.

This project was supported by the Strategic Environmental Research and Development Program (SERDP) under the sponsorship of the Army Corps of Engineers Waterways Experiment Station.

11. References

- [1] Astley, M. and M. Roth, Spline Animator: Smooth camera motion for AVS animations, AVS '94 Conference Proceedings, May 1994.
- [2] Tanaka, H., K.G. Dean, and S. Akasofu, Prediction of the movement of volcanic ash clouds, submitted to EOS Transactions, Am. Geophys. Union, Dec. 1992.
- [3] Tanaka, H., Development of a prediction scheme for the volcanic ash fall from Redoubt Volcano, First International Symposium on Volcanic Ash and Aviation Safety, Seattle, Washington, July 8-12 1991, U. S. Geological Survey Circular 165, 58 pp.

A Graphical User Interface for Networked Volume Rendering on the CRAY C90

Allan Snavely

T. Todd Elvins

San Diego Supercomputer Center

Abstract

SDSC_NetV is a networked volume rendering package developed at the San Diego Supercomputer Center. Its purpose is to offload computationally intensive aspects of three-dimensional data image rendering to appropriate rendering engines. This means that SDSC_NetV users can transparently obtain network-based imaging capabilities that may not be available to them locally. An image that might take minutes to render on a desktop computer can be rendered in seconds on an SDSC rendering engine. The SDSC_NetV graphical user interface (GUI), a Motif-based application developed using a commercially available tool TeleUSE, was recently ported to SDSC's CRAY C90. Because TeleUSE is not available on the C90, the interface was developed on a SUN SPARC workstation and ported to the C90. Now, if users have an account on the C90, they can use SDSC_NetV directly on the CRAY platform. All that is required is a terminal running XWindows, such as a Macintosh running MacX, the SDSC_NetV graphical user interface runs on the C90 and displays on the terminal.

1 Introduction

Volume rendering is the process of generating a two-dimensional image of a three-dimensional data-set. The inputs are a data-set representing either a real world object or a theoretical model, and a set of parameters such as viewing angle, substance opacities, substance color ranges and lighting

values. The output is an image which represents the data as viewed under these constraints. The user of a volume rendering program will want to be able to input these parameters in an easy fashion and to get images back quickly.

The task of generating the image is usually compute intensive. Three-dimensional objects are represented as three-dimensional arrays where each cell of the array corresponds to a sample value for the object at a point in space. The size of the array depends on the size of the object and the sampling rate. Data collected from tomographic devices such as CT scanners are often 256*256*256 real numbers. Grids with 1024 sample points per dimension are becoming common. As sampling rates increase due to improved technology, the data sizes will grow proportionally. Data generated from a theoretical model can also be very large.

There are several algorithms that traverse such sample point grids to generate images. Two of the most popular are splatting and ray-casting. Both of these involve visiting each cell in the array and building up an image as the array is traversed. Without going into the details of the algorithms, it will be apparent that their theoretical time complexity is order

$$O(N_1 * N_2 * N_3)$$

where N_i is the size of the i th dimension. When large arrays are considered, the actual run time on a workstation class CPU may be quite long. The CPU speed and memory limitations of the typical scientific workstation make it unsuitable for interactive speed rendering.

If we were to characterize an ideal rendering machine, it would be; inexpensive, so everyone could have one; very fast to allow interactive exploration of large three-dimensional datasets; and it would sit on the desktop to allow researchers to do their visualizations without traveling.

SDSC_NetV, a networked volume rendering tool developed at the San Diego Supercomputer Center, addresses the needs of researchers who have limited desktop power. SDSC_NetV distributes CPU intensive jobs to the appropriate rendering resources. At SDSC these resources include fast rendering engines on the machine floor. Researchers access these resources via a graphical user interface (GUI) running on their desktop machines. The GUI allows the researcher to enter viewing parameters and color classifications in an easy, intuitive way. The GUI also presents the image when the result is sent back over the network.

The GUI itself is quite powerful. It allows the user to interactively examine slices of the data-set and to do substance property classification without requesting services via the network. Up to eight data ranges can each be associated with a color and opacity values.

Once the user has set all the parameters to his/her satisfaction, a render request causes the render job to be spawned on the appropriate rendering engine at SDSC. The optimized renderer subsequently sends images to the GUI.

The design of such a network GUI is a significant software engineering task, actually as complicated as the coding of the rendering algorithms. Programmers can write GUIs for X-window based applications in raw X/Motif or with a GUI building utility. The second approach is the most reasonable one when the envisioned GUI is large and complex.

2 A C90 GUI

Recently, the SDSC_NetV GUI was ported to the SDSC CRAY C90. The goal was to extend the availability of SDSC_NetV. Previously, the GUI only ran on workstation class platforms. Specifically, Sun SPARC, SGI, DEC and DEC Alpha workstations. The porting challenge proved to be significant due to the fact that we needed to preserve I/O compatibility between the different platforms. Also, there is no GUI builder program on the SDSC C90. An examination of SDSC_NetV's architecture highlights the need for I/O compatibility. SDSC_NetV is a distributed program. The GUI usually runs on the user's workstation. This first component sends requests across the network to a second component, running on an SDSC server which accepts incoming requests for render tasks, and delegates these tasks to a third class of machines, the rendering engines. This means that any time a new architecture is added to the mix, communication protocols between the various machines must be established. Typically, this sort of problem is solved by writing socket level code which reads and writes ASCII data between the machines. Communication via a more efficient and compact binary protocol requires each machine to determine what type of architecture is writing to its port and decoding the input based on what it knows about the sender's data representations. Among a network of heterogeneous machines, establishing all the correct protocols can become a big programming job.

3 A Better Solution

The GUI for SDSC_NetV was developed on workstation platforms using TeleUse. TeleUse is a commercially available GUI builder that allows one to quickly define widget hierarchies and call-back functions. A GUI that might take several days to develop in raw X/Motif can be implemented in a few hours using TeleUse. TeleUse generates source code in C with calls to the X-Motif library. To get SDSC_NetV's GUI running on the C90, we ported the TeleUse generated source code and compiled it. Although this source code compiled almost without modification, the object produced was not executable because of pointer arithmetic unsuitable to the C90's 64 bit word. Programmers in C on 32 bit word machines often treat addresses and integers interchangeably. Of course this will not work on a machine with an address word length different from the integer representation length. These sorts of problems were easily corrected.

As described in the previous section, the GUI has to communicate across the network to request a render. The GUI has to be I/O compatible with the SDSC machines in order for this communication to take place. The problem of I/O compatibility is one that has been encountered before at SDSC. In response to this problem, we have built a library called the SDSC Binary I/O library for compatible I/O between heterogeneous architectures. A network version allows the data to be written across the network with a call to SNetWrite. This results in conversion of the data representation the appropriate format for the target architecture. When the data are read in at the other end, using SNetRead, they are reassembled into the correct representation for the receiver. SDSC_NetV and SDSC Binary I/O are available via ftp from ftp.sdsc.edu.

Once the SDSC_NetV GUI was linked with the SDSC Binary I/O library, a more subtle problem arose. The main window widget was responsive to user input, but none of the sub-windows brought up in response would take input. Eventually it was discovered that an X application structured as groups of cooperating top level widgets would not function. The X-Motif library installed on the C90 expected one widget to be the designated top level manager. The exact reason for this remains unclear as the version of X on the CRAY is the same as the version on the workstations. Once this fact was discovered, the GUI was restructured on our development workstations and re-reported to the C90. With the new widget hierarchy the GUI worked

fine.

Considering the description of rendering given in the introduction, we can form an idea of the attributes for an ideal rendering machine. It would be very fast to allow interactive exploration of large three-dimensional data sets. It would sit on the desktop to allow researchers to do their visualizations without traveling. It would be inexpensive so everyone could have one. SDSC_NetV gives many CRAY users a virtual ideal machine.

4 Results

The goal of SDSC_NetV is to provide cutting edge rendering technology to the researcher on the desktop. Porting SDSC_NetV to the C90 has helped to realize this goal. The GUI runs quickly on the C90. Graphically intensive operations such as slice examination and classification run at interactive speed. The availability of SDSC_NetV has been extended. Now a user with an account on the C90 can display the GUI on a Mac or even an X terminal. The performance of the GUI over the network depends on the speed of the the link. However, the basic functions of setting parameters and displaying images are now available on a platform where such functionality was not available before. SDSC_NetV is used by scientists in a number of disciplines.

5 Images

Art Winfree, a researcher at The University of Arizona, is using SDSC_NetV to gain intuition into stability in chemically reactive environments. Figure 1. shows the main window of the SDSC_NetV GUI with an image of a theoretical model known as an equal diffusion meander vortex. It represents a compact *organizing center* in a chemically excitable medium (or really, a math model thereof.) The main feature is that all substances involved diffuse at the same rate. The organizing center, in this case, is a pair of linked rings, which you can see only as the edges of iso-value fronts you colored with the Classifier. The key thing about this, besides equal diffusion, is that the rings are not stationary, but are forever wiggling in a way discovered only a couple years ago, called *meander*. Despite their endless writhing, which precludes adjective stable, the rings and their linkage persist quite stably. A picture

allows reasoning about such structures which may not be apparent from an equational model.

6 Conclusion

The accessibility of the C90 and the versatility of SDSC_NetV work together to provide a state-of-the-art tool for scientists involved in a wide range of disciplines. Visualization of data representing natural phenomena and theoretical models is now available on more scientist's desk tops.

Acknowledgements

This work was supported by the National Science Foundation under grant ASC8902825 to the San Diego Supercomputer Center. Additional support was provided by the State of California and industrial partners.

Thanks to the network volume rendering enthusiasts at SDSC, in particular, Mike Bailey, Max Pazirandeh, Tricia Koszycki, and the members of the visualization technology group.

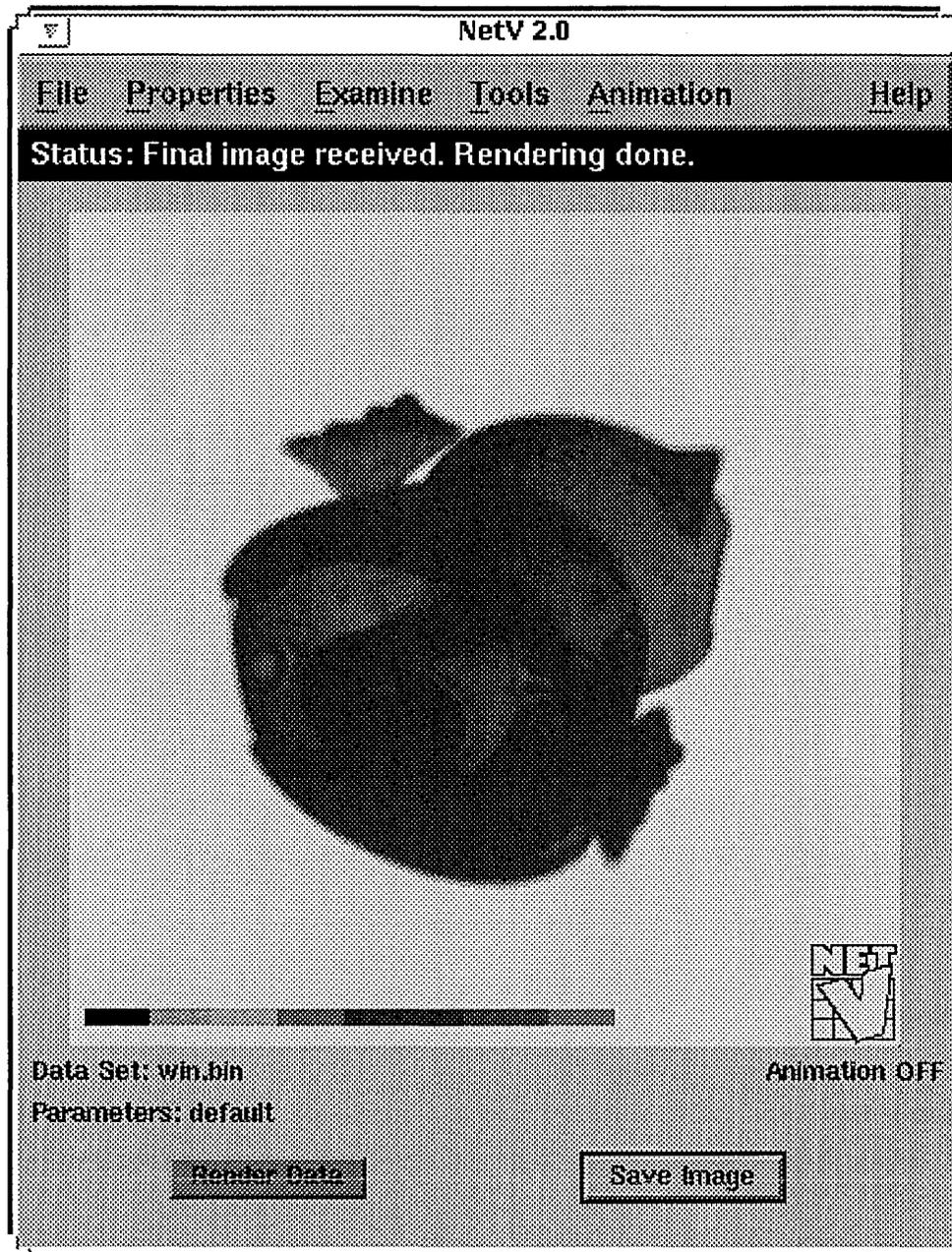


Figure 1: An Equal Diffusion Meander Vortex.

HETEROGENOUS COMPUTING USING THE CRAY Y-MP AND T3D

Bob Carruthers

Cray Research (UK) Ltd.
Bracknell, England

Introduction

Two applications which use a CRAY Y-MP and a T3D in a heterogenous environment will be described. The first application is a chemistry code that was ported from the Y-MP to the T3D, and then distributed between the two machines. The combined solution using both machines ran the test cases faster than either the Y-MP or T3D separately.

The second application is slightly different, since the complete problem could not be run on the T3D because of memory limitations imposed by the design of the code and the strategy used to generate a parallel version. Distributing the problem across the Y-MP and T3D allowed the application to run, and produced a time faster than that on the Y-MP.

Both these applications were encountered as part of two benchmark efforts, and thus show how real user problems can be adapted to heterogenous computing.

Application 1

The first application is a large chemistry code, that is regularly used in a production environment, and runs on both CRAY Y-MP's and on several MPP platforms. The version that was included in the benchmark used PVM for message passing, and was initially run on a Y-MP in this form to validate the code. Following this, the code was ported to the T3D, and various optimisations to the I/O and message passing implemented to improve the performance. In this form, the T3D ran the code considerably faster than the Y-MP.

As part of the benchmark activity, we were looking for ways to improve the performance of the code, and to demonstrate the benefits of the Cray Research Heterogenous Computing Environment. This particular code had sections that were not well suited to an MPP, but were known to run well on a vector machine. With this in mind, a strategy was evolved to do the well vectorised part on the Y-MP, and leave the rest on the T3D where it could take advantage of the MPP architecture. The strategy for this is outlined below.

First, a master control program had to be written that executed on the Y-MP. This was responsible for establishing contact with PVM, reading in the number of PE's to be used on the T3D, and then spawning the necessary tasks on the T3D via PVM. The Fortran code for this part of the operation is shown below:

```
      include './sizes'
c
      common/wordlen/nrbyte,nibyte
      common/pvm/npvm,ipvm
c
      print *, 'Input the Number of Processes Required ',
2 'on the CRAY T3D'
      read(*,*)npvm
c
      call pvm_control()
      stop 'End of Solver'
      end
c
      subroutine pvm_control()
      .
      include './sizes'
      include './fpvm3.h'
      include './jrc_buf_sizes'
c
      common/wordlen/nrbyte,nibyte
      common/pvm/npvm,ipvm
      character*8 mess_buff
      dimension itids(npvm)
c
      call pvmfmytid(itid)
      if(jrc_debug.eq.1) then
         write(0,*) '
         write(0,2000) itid
2000   format('TID of SOLVER Process on the ',
2 ' CRAY Y-MP is ',z16)
      endif
c
      call pvmfspawn("a.out", PvmTaskArch,
2 'CRAY', npvm, itids,numt)
c
      if(numt.ne.npvm) then
         write(0,*)'Response from PVMFSPAWN',
2 ' was ',numt,' rather than ',npvm
```

```

stop 'Error in PVMFSPAWN'
else
do i=1,npvm
  if(itids(i).ne.1) then
    itid_pe0=itids(i)
    ntid_pe0=i
    goto 1000
  endif
end do
write(0,*)'All T3D TID's are 1 - PE 0 is absent'
stop 'TID Error'
1000 continue
if(jrc_debug.eq.1) write(0,2100) itid_pe0
2100 format('T3D Initialised by SOLVER - PE 0 ',
2 ' has TID ',z17)
endif

```

The main program begins by asking the user for the number of PE's to be used on the T3D, and then calls the main control routine, `pvm_control`. This enables the latter routine to accurately dimension the array 'itids' which holds the PVM Task Identifiers (TID's) of the PE's on the T3D.

This routine finds the TID of the Y-MP process and prints it out, and then spawns the T3D processes. The variable 'numt' contains the number of processes actually spawned by PVM, which is checked against the number requested. The final part of the set up procedure involves the Y-MP searching for the TID of PE 0 on the T3D. By default only PE 0 can communicate with the Y-MP, and PE's that cannot communicate with the Y-MP have their TID's set to unity. This can be modified if required by an environment variable.

The control routine then enters a state where it waits for the T3D to send it work to do.

```

c
1100 continue
call pvmfrecv(itid_pe0, 9971, ibufid)
call pvmfunpack(BYTE1, mess_buff, 8, 1, info)
c
if(jrc_debug.eq.1) write(0,*)'SOLVER ',
2 'Received Message "',mess_buff,'" from PE 0'
c
if(mess_buff(1:5).eq.'Solve') then
  call bob_do_work(itid_pe0, jrc_debug)
else if(mess_buff(1:8).eq.'Finished') then
  return
else
  write(0,*)'Illegal Message Found in ',
2 'SOLVER - ',mess_buff
  stop 'Protocol Error'
endif

```

```

goto 1100
end

```

The control routine can interpret two messages from the T3D - 'Solve' indicating that it should do some work, and 'Finished' indicating that the T3D has finished its work, and that the Y-MP process should clean up and terminate.

Further data is exchanged between the T3D and Y-MP via both PVM and the UNICOS file system. Control parameters are sent via PVM, while the main data array is sent over as a file. The Y-MP is responsible for performing the necessary data format conversion from IEEE to Cray Research Floating Point format, and performing the reverse operation when it has finished computing and wishes to return information back to the T3D. This is done using IEG2CRAY and CRAY2IEG respectively with conversion type 8.

While the Y-MP is working, the T3D enters a wait loop similar to that on the Y-MP, and waits for the signal 'Done' from the Y-MP. At this point it picks up the new data file. Note that both the T3D and the Y-MP must have set up their data files before signalling that there is work to do.

The final point to remember is that any task spawned by PVM will use the directory pointed to by the shell variable \$HOME in which to create and search for files. If the Y-MP executes in any other directory than \$HOME, the exchange of files with the T3D will not take place. This can be controlled by changing \$HOME prior to starting the Y-MP process so that it points to the correct directory while the tasks are running.

This heterogenous approach enabled the time for the complete job to be reduced, so that it was less than either the time on the Y-MP or the T3D.

Application 2

Like the first code, this application is a large user program that regularly runs on a Y-MP. The owners of the code funded one of the universities in the UK to produce a parallel, distributed version which could be run on a group of workstations.

We started to look at this version of the code when we received a benchmark containing it. The initial part of the work consisted of converting the code from its 32-bit version using a local message passing language to a 64-bit version that used PVM. This PVM version was eventually ported to both the Y-MP and the T3D, and ran the small test cases provided.

However, the design of the program and the parallel

implementation constrained the maximum size problem that could be run on the T3D, and meant that the large problems of greatest interest could not be run. The underlying method of locating data in memory used the "standard" Fortran technique of allocating most of the large data arrays in one large common block via a memory manager. The strategy to generate a parallel version of the code relied on one master processor doing all the input and data set up, and then distributing the data to the other processors immediately before running the parallel, compute-intensive part of the code. Similarly at the end, the results were collected back into the master processor for output and termination processing. This meant that the master processor needed sufficient space to store all the data at the start and the end of the compute phase. For the problems of interest, this space is typically over 30 Mwords on the Y-MP, well beyond the capacity of single PE on the T3D.

The strategy to solve this dilemma was to split the computation into three distinct phases. The first, which is the initialisation, runs on the Y-MP, and instead of distributing the data to other processors prior to the parallel section, outputs the required arrays to a series of files and then terminates. The second phase which runs on the T3D picks up the required data from the UNICOS file system, completes the parallel compute-intensive part of the calculation, and then outputs the results to a second set of files. The third phase runs on the Y-MP and performs the merging of the resultant arrays and outputs the results.

In this approach, those phases that require a large amount of memory are run on the Y-MP, while the T3D executes the parallel part which contains the heavy computation. Although the scheme sounds simple in outline, the implementation was actually quite tricky for a number of reasons:

- The arrays to be distributed for parallel processing are defined by the user in the input data, as are those required to be saved after the parallel processing. This means that all three phases must be able to read the input data, but only select those commands that are necessary to perform their operations.
- Data other than the arrays mentioned above need to be passed between the various phases - for example, control variables and physical constants that define the problem. These are typically stored in separate common blocks, and do not have to be selected via the user input for distribution and merging.

For the transition between the input processing and the parallel computation phases, this proved easy to sort out, since the original parallel code had had to distribute most of this information from the

master processor as well. At the end of the parallel processing, however, the master processor was assumed to have all the data it needed, and the other PE's only sent back their share of the distributed arrays. The merge process thus needed careful analysis to ensure that all the relevant data was regenerated in the master processor.

- Although the problem is defined above in terms of the T3D performing phase 2, there is no reason why any other machine running PVM could not perform the parallel part, in particular several processors of a Y-MP or C90. To allow this to happen, the T3D specific code had to be compiled in or out conditionally, and data conversion only applied when strictly necessary.
- For small problems, there is no need for three separate images to be run. The code therefore needed an option to allow all three phases to be executed during one pass through the code, either on the T3D or any other platform. This imposes some extra logic in the code, but means that it can be run as originally intended.
- For a given hardware platform, the same binary executes any of the phases, or all three phases together. The logic for this is embedded in the code, and controlled by input variables.
- To simplify code maintenance, it was decided that there would be only one version of the source. Different hardware platforms are selected via conditional compilation.

The progress through the various stages is made transparent to the user via the use of shell scripts. The user can submit a problem for execution to the Y-MP, and the various phases are executed on the appropriate hardware. Restart facilities can be included if necessary.

This approach also offers considerable flexibility when either running or testing the program. Unlike the first application, it is not necessary to run each part immediately after the preceding one, nor do the Y-MP and T3D have to wait for messages from each other. It is simply necessary to preserve the files between the various phases, so that each phase can be started when convenient. Finally, code can be tested or developed on either the Y-MP, T3D or both.

This approach allows what seems at first glance to be intractable problem to be solved using the heterogeneous environment available with the Cray Research T3D. The Y-MP component is used for the pieces that it is best suited to, while the T3D performs the heavy computation for which it was designed.

FUTURE OPERATING SYSTEM DIRECTION

Don Mason
Cray Research, Inc.
Eagan, MN

INTRODUCTION

The UNICOS operating system has been under development since UNIX System V was ported to Cray Research hardware in 1983. With ten years of development effort, UNICOS has become the leading UNIX implementation in a supercomputing environment, in all aspects important to our customers: functionality, performance, stability and ease of use.

During these ten years, we have matured from a system architecture supporting only four processor CRAY-2s or X-MPs, to the complexity of 16 processors of the C90, and hundreds of processors on a T3D. We invented and implemented new I/O structures, multiprocessing support, including automatic parallelization, and a host of industry leading tools and utilities to help the users to solve their problems, and the system administrators to maximize their investment.

The system, which initially was simple and small, grew in both size and complexity.

The evolution of hardware architectures towards increasing number of processors in both shared and distributed memory systems and the requirements of open, heterogeneous environments, present challenges that will be difficult to meet with current operating system architecture. We decided that it was time for us to revise our operating system strategy, and to define a new path for the evolution of UNICOS. This evolution should enable us to face the challenges of the future, and at the same time preserve our customers' and our own software investments.

After two years of careful evaluation and studies, in 1993 we decided to base the future architecture of UNICOS on microkernel technology, and we selected Chorus Systems as the technology provider.

This evolution will preserve all the functionality and applications interfaces of the current system, and the enhancements that will come in the meantime.

MICROKERNEL/SERVER ARCHITECTURE

Figure 1 compares the current UNICOS architecture to the future one, that we refer to as "serverized."

The current system architecture is depicted on the left side. It is characterized by a relatively large, monolithic system kernel, that executes in the system address

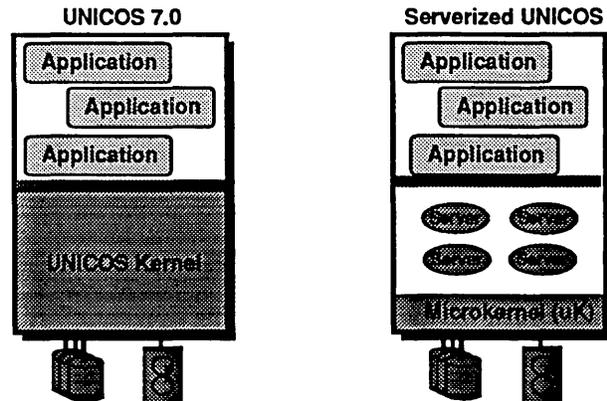


Figure 1

space, and that performs system work on behalf of user's applications, on the top part of the figure. The applications interact with the system via a "system call interface," represented on the figure by the "interface" area. The kernel, to perform its tasks, uses "privileged" instructions, that can only be executed in the system space.

The right side of Figure 1 represents the new architecture. Nothing changes in the upper part of the figure: the users applications are exactly the same, and they use the same system interface as before. The monolithic kernel is now replaced by a "microkernel" and a set of new entities, called "servers." The microkernel provides the "insulation" of the system software from the particular hardware architecture, and provides a message passing capability for transferring messages between applications and servers, as well as between servers themselves.

System tasks that previously were performed by the system kernel will be executed by the specialized servers. Each of them is assigned a particular and well defined task: "memory manager", "file manager", "process manager" etc... The servers, in their majority, operate in the system space. However, some of them, which do not require access to privileged instructions, can perform in user space. Each of the servers is "firewalled" from the others, and can communicate with the others only through a well defined message passing interface, under the microkernel's control.

Microkernels can also communicate from one physical system to another, across a network. This is the situation represented by the Figure 2. In a configuration like the one depicted, not all systems need to have all the servers. Certain systems can be specialized for particular tasks, and therefore might require only a

subset of the available servers. In the case of a CRAY MPP system, each of the processing elements contains a microkernel, and few servers, only those that are necessary for executing applications. Most of the system services can be provided either by some other PEs of the system, which have the appropriate server, or even by a different system on the network.

For an MPP in particular, this frees the PEs memory space for user applications, rather than using it for the system.

A long-term objective with the new architecture is to provide a single UNICOS Operating System which will manage the resources of diverse hardware architectures. This is called "Single System Image" (SSI) in the industry. From an administrator's point of view SSI will facilitate management of computing resources as if they were a single platform; for example, an MPP and a C90. From an applications point of view SSI means OS support for scheduling computing resources such that components of the application can efficiently utilize diverse platform characteristics.

There are several advantages of adopting a serverized architecture:

- o Most of the system software is "insulated" from the hardware. Therefore, porting to new architectures is made much easier, and safer.
- o System functions can be distributed across platforms.
- o Servers are easier to maintain, since each of them is relatively small and self-contained. Interactions between the server and the "external world" are done via clear interfaces. A change to a server should not have any impact on other parts of the systems.
- o Servers can be made more reliable, precisely because of their smaller size, and well defined functions and interfaces. They can be "cleanly" designed, and well tested.
- o A serverized system can evolve in a "safer" way than a monolithic kernel.
- o Systems can be customized by introduction of custom servers, designed to perform a particular task, not required by other customers.
- o If an industry agreement on microkernel interfaces is achieved, this would open the way for leveraging servers across different platforms and architectures.

THE CHORUS TECHNOLOGY CHOICE

Before selecting Chorus technology, we conducted a comprehensive study of the technologies available in

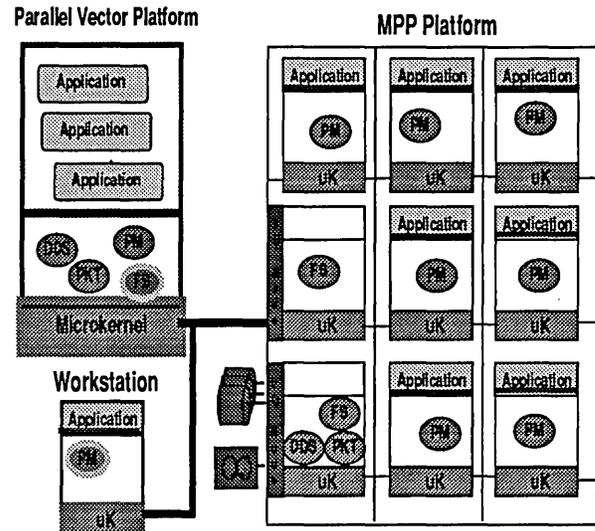


Figure 2

the market, and their adaptability to Cray Research hardware architectures. Several options were examined, including Mach. There are more similarities than differences among the available microkernel technologies. All microkernels attempt to mask hardware uniqueness and manage message passing. The differences become evident only when looking at the application of the technology to specific hardware platforms such as the CRAY Y-MP or CRAY T3D series. The selection criteria included the following:

- o The adaptability to a real memory architecture
- o Performance
- o Ease of implementation — time-to-market
- o Existence of programming tools
- o Serverization model (existence of a multi-server implementation)

The technology from Chorus Systems was selected, since it perfectly satisfied all of our selection criteria. In particular, this choice will allow us to deliver a stand-alone MPP capability approximately 18 months sooner than with any other technology. Also, the same technology can be used on both real and virtual memory architectures.

TIMETABLE

The transition from current UNICOS and UNICOS MAX to the new architecture will take from two to four years, depending on hardware platforms. Initial implementation will become available on the MPP systems. In parallel with the development of current structure of UNICOS, we will work on a serverized version for parallel vector platforms.

Future Operating System Directions - Serverized Unicos

Jim Harrell

Cray Research, Inc.
655-F Lone Oak Drive
Eagan, Minnesota 55121

ABSTRACT

This paper is the technical component of a discussion of plans for changes in operating system design at Cray Research. The paper focuses on the organization and architecture of Unicos in the future. The Unicos operating system software is being modified and ported to this new architecture now.

1 Introduction

Over the past several years the Unicos Operating System Group has been studying the needs and requirements for changes in Unicos to provide support for new Cray hardware architectures, and new software features as required by Cray customers. At previous Cray User Group Meetings and in Unicos Advisory Meetings there have been open discussions of the requirements and choices available. In 1993 a decision was made to move forward with the Chorus microkernel and Unicos as the feature base and application interface. This talk describes some of the technical components of the new operating system architecture, explains how the new system is organized, and what has been learned so far.

The talk is composed of five parts. The first part describes the Chorus microkernel and the server model. This is the base architecture we have chosen to use in the future. We reviewed the choices at the Cray User Group Meeting in Montreaux in March of 1993. The second part of this talk describes our experiences in porting Chorus to a Cray YMP architecture. This phase of the program was an important step in proving the technology is capable of supporting high performance computing. The third part of the talk explains how we expect Unicos will "map" onto the Chorus model. We have chosen to move Unicos forward as the feature base and application interface. Our goal remains to provide full application compatibility with Unicos. The fourth part describes the technical milestone for 1993 and the status of that milestone. The final part of the talk discusses our interest in interfaces for servers and microkernels that can allow vendors and users with different operating system bases to provide heterogeneous distributed systems in the future.

2 The Chorus Model

The Chorus model is based on decomposing a monolithic system into a microkernel and a set of servers. The microkernel manages the hardware and provides a basic set of operations for the servers. The servers implement a user or application interface and, in Chorus, are usually bound into the same address space as the microkernel. This binding of servers is done for performance. Servers can run in user mode. This allows flexibility in the operating system organization and can add resiliency. An important component of the Chorus model is the Remote Procedure Call (RPC) support. Traditionally RPCs require a context switch, and message or data copies. In the Chorus model this is referred to as Full Weight RPC (FWRPC). Chorus provides a Light Weight RPC (LWRPC) that can be used to communicate between servers in kernel space, that is, servers bound in the same address space as the microkernel. LWRPC does not context switch, or change stacks. Instead of copying message data, a pointer to the message is passed. The result is a significant reduction in the cost of communication between servers. The offset to the performance improvement is that servers in the same address space are not protected from random memory stores. There is no "firewall" between the servers. There is an obvious requirement for FWRPC across a network, and for servers running in user mode, user space. But when servers are in the same address space the requirement is not as obvious.

3 Porting Chorus to a YMP

In 1992 we worked with Chorus Systems to port a minimal Chorus operating system to a YMP machine. The purpose of this test was to determine if the software architecture was viable on a YMP hardware architecture. There were concerns about Chorus memory management on a YMP. Most microkernel based systems use virtual

memory extensively. Chorus claimed to be architecture neutral in memory management. There were other concerns about machine dependent differences. Chorus is normally ported to smaller machines. The port would provide answers to these concerns and allow us real hands-on experience with Chorus. Use of the code would make the evaluation and comparison with other systems real. We set a goal of demonstrating at the end of six months.

The Chorus components of the port were the microkernel, a Process Manager (PM), and a very simplified Object Manager (OM), or filesystem server. The machine-dependent parts of the microkernel were modified to support the YMP, and a simple memory management scheme was put in place to support the YMP. The Chorus PM was modified to use Unicos system call numbering so we could run Unicos binaries. This greatly simplified what had to be done to run tests. The Chorus OM was greatly simplified. The support for Unix filesystems was removed and in its place was put a table of files that would be "known" to this OM. All of the disk device support was removed from Chorus, and replaced with code to access files from YMP memory. This dispensed with the need for a filesystem, drivers, and IOS packet management.

The ported system was booted on a YMP and simple tests run from a standard Unicos shell, /bin/sh. This confirmed our view that Unicos could be used as the operating system personality. The Unicos shell had been built under standard Unicos and yet under the test system it functioned exactly as it did under Unicos. The tests that were run did a variety of very simple system operations. The test results were compared to Unicos results using the same tests. The Unicos system was run on the same YMP and configured to use a memory filesystem. The results showed two important facts. Using FWRPC the performance of Chorus was 2 times slower than Unicos for the same system call. Using LWRPC the performance was comparable to Unicos.

Cray Research is not planning on using all of the Chorus product. We had previously decided, in conjunction with our customers, that we should use Unicos as the base of any future systems. We want to use certain features from Chorus to help split Unicos into components. The primary Chorus technology that is being used in the restructuring of Unicos is the microkernel. It is much the same as the Chorus version, with the exception of the machine dependent portions. We are augmenting it to provide support for some other services like swapping and enhancements to scheduling and monitoring. We are also using the Chorus Process Manager (PM) as the basis for our PM. We are modifying the way that the system calls, signalling, etc., work to match Unicos.

The last major piece of Chorus technology we are using is the server "wrappers". This is a series of routines that provide two capabilities. The first is a model for the interfaces needed to get a server to communicate with another server. The second is as a model for how to mimic Unix or Unicos internal kernel requests or convert the kernel requests to microkernel requests.

4 Mapping Unicos onto the Chorus Model

The restructuring of Unicos will maintain complete user compatibility. At a very early stage of the project we determined that the best way to provide this compatibility is to use as much Unicos code directly as possible. This has a side effect, in that we can move more quickly to serverize Unicos without having to rewrite code. We have chosen to have a large number of servers, aggressively trying to modularize Unicos wherever possible. We believe that there are at least a dozen different potential servers. For example the device drivers for disk, tape, and networking will form three separate servers. The IOS packet management code has already been made into a server. The terminal or console support is also already a separate server.

5 1993 Milestone - Some Progress

At the end of 1993 we completed one of the formal project milestones. We ran a system composed of a Chorus based microkernel, our PM, an OM or filesystem server that implements the NC1 filesystem, a disk server for disk device support and a packet server. We also added a terminal server for console communications. The system was run on a YMP using an 8.0 filesystem on Model E IOS connected disks. This milestone verified that several major servers were functioning together and that device access worked. This milestone also continues to monitor progress with the goal of Unicos application compatibility.

6 Future Interfaces

In the computer industry there are several companies and research facilities that are studying microkernels and serverized systems. We expect that a number of distributed systems will take a similar form to the direction we have chosen. Cray Research believes that in the future heterogeneous systems will depend on different operating systems from different vendors being able to interact and interoperate at a deeper level than currently exists. This interoperation is required to support Single System Image and system resource management in distributed systems. In order to facilitate this communication Cray is taking a leadership role in trying to find

ways to standardize server and microkernel interfaces. This work has met with some success, but will require the interest and participation of our customers to convince computer vendors that Single System Image is a serious requirement in the future.

7 Summary

We have shown progress towards the reorganization of Unicos into a more modular form. We expect that this new system will be capable of supporting all Cray hardware architectures, and capable of supporting all Cray customer requirements by providing a better base for new functionality and compatibility for current Unicos applications.

Mass Storage Systems

Storage Management Update

Brad Strand

Section Leader, UNICOS Storage Management Products

Cray Research, Inc.
655-F Lone Oak Drive
Eagan, Minnesota 55121
bstrand@cray.com

ABSTRACT

This paper presents an update to the status of UNICOS Storage Management products and projects at Cray Research. Status is reported in the areas of Hierarchical Storage Management products (HSMs), Volume Management products, and Transparent File Access products. The paper concludes with a timeline indicating the approximate introduction of several Storage Management products

1 Topics

Work on Storage Management products and projects at Cray Research is currently focused in three major areas. The first area is Hierarchical Storage Management products, or HSMs. These products are designed to allow sites to increase the effective size of their disks by transparently moving, or *migrating*, data between disks and cheaper media, such as tape. These products allow sites to make their disk storage pool appear larger than they actually are. The second area where Cray Research is currently doing Storage Management work is in Volume Management. Volume Management products allow users and administrators to use and manage a large set of logical and physical tape volumes in an easy manner. Finally, Cray Research is very active in the area of Remote Transparent File Access products. These are products which allow users and applications to access files which physically reside on another system as if they were on the local system. These products are often called "Remote File System" products, because of the way they effectively extend the local physical file system across the network. Each of these three areas will be discussed in terms of current product availability, and in terms of development projects currently active and underway. The paper concludes with a timeline designed to indicate the relative times in which Storage Management products are expected to be introduced into the UNICOS system.

2 Hierarchical Storage Management Products (HSMs)

2.1 *Data Migration Facility (DMF)*

2.1.1 *DMF 2.1*

A new version of the CRAY Data Migration Facility, or DMF, version 2.1, is now available. DMF 2.1 is designed to run on top of UNICOS 7.0, UNICOS 7.C, and UNICOS 8.0. DMF 2.1 provides several important new features in DMF, a few of which will be described below.

DMF 2.1 adds support for Multi-Level Security, or MLS. This means that sites running with UNICOS MLS can use DMF to provide their HSM solution. DMF is even part of the "Evaluated System," which means that sites running Trusted UNICOS can run DMF without violating the security rating of their system.

DMF 2.1 also provides support for a multi-tiered data management hierarchy, in that data may be moved between Media Specific Processes (or MSPs). This means that sites can configure their systems to migrate data from, for example, one tape format to another.

DMF 2.1 also adds support for gathering a variety of dmdaemon statistics. These statistics may then be analyzed using the new dmastat(1) utility.

2.1.2 Client/Server DMF Project

One of the DMF development projects currently underway at Cray Research is called Client/Server DMF. This project adds the functionality which will allow sites to use DMF to manage their data which are stored on Shared File Systems, or SFSs. The Shared File System is an evolving product, not yet available, which allows multiple UNICOS systems to share direct access to disk devices. The Client/Server DMF project allows a single DMF server to manage all the data in a UNICOS Shared File System environment.

Each UNICOS machine in the SFS complex needs to run a copy of the DMF Client. One or more UNICOS hosts with access to the secondary storage media (tapes) needs to run the DMF Server. The system can also be configured to provide redundancy, should one particular DMF Server process fail.

The Client/Server DMF project internal goal is to demonstrate the functionality by year-end. More information on Client/Server DMF will be available at the "Clusters BOF," hosted by Dan Ferber.

2.1.3 New Tape Media Specific Process (MSP) Project

The other major DMF development project currently underway is that which is developing a new tape MSP. This is the tape MSP that was originally planned to be available in DMF 2.1, but has since slipped into DMF 2.2. This new tape MSP is designed to provide a variety of important improvements. Several of these are listed below.

- *Support for Improved Data Recording Capability (IDRC).* Some tape devices have controllers which provide on-the-fly data compression. This feature is called Improved Data Recording Capability, or IDRC. The new tape MSP will provide support for controllers using IDRC.
- *Improved Media Utilization.* The current tape MSP design does not support the function of "append" to partially written tapes. The new tape MSP will support appending to tapes, and will thereby obtain greater utilization of tape media.
- *Much Improved Media Recovery.* The new tape MSP writes to tapes in *blocks*. The new tape MSP is designed to be able to read and recover all blocks which do not contain unrecoverable media errors. Thus, only data in blocks which contain unrecoverable media errors would be lost. This is a major improvement to the current design, which is unable to retrieve data written beyond bad spots on the tape.

- *Absolute Block Positioning.* Some new tape devices support high speed tape positioning to absolute block addresses. The new tape MSP will utilize this feature whenever it is available.

- *Asynchronous, Double-Buffered I/O.* To fully utilize the greater bandwidth available on some tape devices, the new tape MSP will use asynchronous, double buffered I/O. We expect this will yield very near full channel I/O rates for these devices.

- *New Tape and Database Formats.* To provide some of the new functionality, changes were made to the tape format, and to the MSP database format. The new tape MSP will support reading tapes in the old format. Conversion utilities will be supplied to convert databases from the old format to the new format.

- *Availability.* The new tape MSP will be available in DMF 2.2. We expect this release to be available in the fourth quarter of 1994.

2.2 UniTree

The UniTree HSM product has now been ported to the Y-MP EL platform by Titan Client/Server Technologies. We are currently awaiting a Titan installation of UniTree at our first customer site. The UniTree version ported to UNICOS 7.0.6 is version 1.7. Titan has not shared their plans to port version 1.8 to UNICOS.

2.3 FileServ

A port of the FileServ HSM product to UNICOS is currently underway. This port is being done by EMASS. Cray Research has cooperated with EMASS by providing "hooks" into the UNICOS kernel which allows FileServ to obtain the information it needs more easily. These hooks are integrated into UNICOS 8.0. Since the porting work is being done by EMASS, persons interested in obtaining more detailed information, or project schedules, should contact them directly.

2.4 "Open HSM" Project

Although DMF provides an excellent, high-performance HSM solution on UNICOS platforms, some of our customers have indicated that the proprietary nature of DMF is a disadvantage to them. They would prefer a solution that is more "open," in the sense of being available on more than one hardware platform. In this way, the customer's choice of HSM solution would not necessarily dictate their choice of hardware platform. In response to this requirement, Cray Research has begun a project with the goal of providing an HSM product on UNICOS which is close to DMF in performance and

functionality, yet which is also available on other hardware platforms.

We have been evaluating potential candidates to become our open HSM product for about six months. Much of our work has been analyzing product designs, to see which ones have the potential for being integrated into our unique supercomputing environment. As one might imagine, there are many challenges to address when attempting to integrate an HSM product with our Tape Daemon, Multi-Level Security, and very large, high performance peripherals. Moreover, most of the products we evaluated were not designed for multi-processor architectures, so there is a significant amount of design work required to determine just where we can add parallelism into these products.

Despite these hurdles, we feel we have made significant progress on the project. Indeed, we feel we are close to a decision point for selecting the product we will use as our base. Our target is to have an open HSM product running on UNICOS in 1995. Depending on which product we choose, and the platforms on which it already runs, it may be possible that a version of the open HSM product will be available on the Cray Research SuperServer platform before a UNICOS-based product is available.

3 Volume Management Products

3.1 CRAY/REELlibrarian (CRL)

3.1.1 CRL 2.0.5 Complete and Available

Release 2.0.5 of CRAY/REELlibrarian is now available. There are several important changes and improvements to the product, including MLS support, ER90 tape device support, and support for 44-character file ids. The database format for CRL 2.0.5 is incompatible with the format used in CRL 1.0.x, but a conversion utility is supplied with CRL 2.0.5. Because CRL 2.0.5 takes advantage of Tape Daemon interface enhancements in UNICOS 8.0, CRL 2.0.5 will only run on UNICOS 8.0 or higher. CRL 2.0.5 is not supported on UNICOS 7.0 or UNICOS 7.C.

3.1.2 CRL Database Study Project

The primary CRL project currently underway is a study which is examining the feasibility of incorporating an improved database technology into CRL. The motive for this study is to improve the reliability and the scalability of the CRL product. No decision has yet been made as to whether or not we will proceed with this database upgrade.

4 Remote Transparent File Access Products

4.1 Open Network Computing/ Network File System (ONC/NFS)

4.1.1 NFS Improvements in UNICOS 8.0

A great deal of effort went into improving the NFS product Cray Research released in UNICOS 8.0. Improvements include the implementation of server side readaheads, improved management techniques of the mbufs used by the Y-MP EL networking code, new options for the mount(8) and exportfs(8) commands which can provide dramatically improved performance in the appropriate circumstances, and the support for B1 level MLS.

Much more information about the NFS changes introduced in UNICOS 8.0 is given in the CUG presentation given in Kyoto last September. Please refer to that presentation for further details.

4.1.2 ONC+ Project

The primary active development project in the NFS area is ONC+. ONC+ is a set of enhancements to the current set of ONC protocols. Features of our ONC+ product are listed below.

- NFS Version 3 is an enhancement to the current NFS protocol, NFS Version 2. NFS Version 3 provides native support for 64-bit file and file system sizes, provides improved support for files with Access Control Lists (ACLs), and provides a wide range of performance improvements.

- Support for Version 3 of the LockManager protocol, which provides advisory record locking for NFS Version 3 files.

- Support for the AUTH_KERB flavor of Remote Procedure Call (RPC) authentication. AUTH_KERB implements Kerberos Version 4 authentication to RPC on a per-request basis. This component of the project adds AUTH_KERB to both user-level RPC calls, and to the kernel-level RPC calls that are used by NFS. The result is a much greater level of RPC security than is offered by either AUTH_NONE, AUTH_UNIX, or AUTH_DES, the current supported RPC authentication types.

- Support for NIS+, the enhanced version of the Network Information Services (NIS) protocols. These are important enhancements which add security, functionality, and performance to NIS.

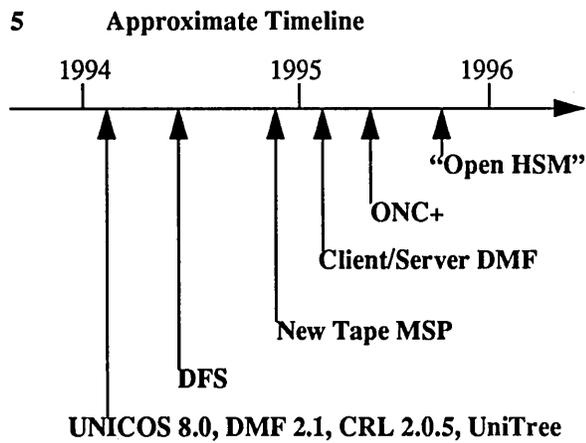
The CRAY ONC+ product will be a separately licensed product, available in UNICOS 9.0.

4.2 Open Software Foundation/ Distributed File System (OSF/DFS)

An important project in the area of Remote Transparent File Access software is OSF/DFS. The DFS product provided by Cray Research will provide most of the important features of DFS, including support for both client and server, as well as for file caching. However, the Episode file system is not provided with DFS, so certain Episode file system specific functions, such as support for filesets, is not yet supported by our DFS implementation.

The DFS server will be available as the CRAY DCE DFS Server. The DFS client will be available as part of the CRAY DCE Client Services product. Both of these products are separately licensed, and both will be available 3Q94.

More detailed information about the Cray Research DFS product will be given by Brian Gaffey in his talk "DFS on UNICOS," scheduled for Thursday, 3/17/94 at 9:30.



RAID Integration on Model-E IOS

Bob Ciotti

Numerical Aerodynamic Simulation

NASA Ames Research Center

Moffett Field, CA 94035, USA

ciotti@nas.nasa.gov

Abstract

The Redundant Array of Inexpensive Disks (RAID) technology has finally made its way into the supercomputing market. CRI has recently made available software for UNICOS to support this. This paper discusses the experiences over the past twelve months of integrating a Maximum Strategy RAID into the C90 environment. Initial performance and reliability were poor using the early release Cray driver. Over time, performance and reliability have risen to expected levels albeit with some caveats. Random i/o tests show that RAID is much faster than expected compared to CRI DD60s.

1.0 Introduction

The Numerical Aerodynamic Simulation facility at NASA Ames Research Center provides a large scale simulation capability that is recognized as a key element of NASA's aeronautics program, augmenting both theory and experimentation [cooper93]. As a pathfinder in advanced large scale computing, the NAS program tests and evaluates new hardware and software to provide a unique national resource. This role provided the basis for NAS entering into a development/integration project using HiPPI connected RAID. As such, NAS was the first site to use the CRI IPI-3 driver to access a HiPPI connected RAID from the C90.

Maximum Strategy Incorporated (MSI) began manufacturing a HiPPI attached RAID beginning with the Gen-3 system in early 1990. These systems cost \$15/megabyte. Comparable CRI disk was available at \$50/megabyte (DD40), with their top of the line disk offered at a hefty \$200/megabyte (DD49). With such a difference in cost and the potential of high performance, the MSI systems were extremely attractive.

The availability of the first Gen-3 systems, led to the prospect of providing inexpensive directly attached disks which transferred data at over eighty megabytes/second. IDA Princeton led the first integration project of this technology into a Cray environment [cave92], connecting a Gen-3 system to a

CRAY2. They developed a software driver which was the starting point for that available on CRI Model-E IOS systems today. NAS considered attaching the Gen-3 to a YMP8-8/256 IOS-D system, but development time, loss of production, and short expected lifetime negated any cost savings. At that time the CRI solution cost \$39/megabyte while MSI RAID was \$13/megabyte. These prices included all required hardware (e.g., \$250,000 for a CRI IOS-D HiPPI channel).

Understandably, CRI has been extremely slow to integrate this cost effective storage into their product offerings, choosing instead to build their own narrow stripe RAID product from Single Large Expensive Disks (SLEDs) [badger92]. This largely ignores the calls of customers to provide fast inexpensive media. Thus, the procurement for High Speed Processor 3 (HSP3) contained a requirement that potential vendors supply support for IPI-3 over HiPPI.

Better performance would be achieved with direct support of the MSI RAID system in CRI IOS hardware, yet even with the 20%-30% overhead of IPI-3 over HiPPI, performance is still very good.

In late 1992, a separate procurement for HiPPI attached RAID awarded MSI a contract to supply 75 gigabytes of storage. The cost was approximately \$9/megabyte. Since that time, competition has fostered falling prices with Gen-4 systems available in quantity today at around \$5/megabyte.

After the installation of HSP3 (C916/1024) in March 1993, twelve months of testing were required before RAID provided a reliable low cost and high performance alternative to CRI proprietary disks.

2.0 Overview

The original RAID papers came out of the University of California at Berkeley in late 1987 [patterson87, patterson88]. It was clear the gains in capacity and performance of SLEDs was modest compared to that achievable from RAID. At the time of

[patterson87], Maximum Strategy had already built and marketed its first RAID product and completed the design of its second. MSI introduced the Strategy-1 in mid 1987, a RAID level 0 system capable of a sustained 10 megabytes/second over VME. August 1988 marked the Strategy-2 introduction, a RAID level 3 product capable of 20 megabytes/second over VME. In June 1990, MSI introduced the Gen-3, also RAID level 3, that sustained a transfer rate of over 80 megabytes/second via HiPPI. Gen-4 became available in August 1992.

2.1 Gen-4

2.2 Overview

The MSI Gen-4 product is composed of a main processor, HiPPI channel interface, ethernet interface and 1 or 2 facilities. At NAS, each facility has 20 1.3 gigabyte drives. Two drives are combined into a module and are striped either 4, 8 or 9 wide. Optimal conditions can produce transfer rates of over 80 megabytes/second. A hot standby module is available in the 8 wide stripe configuration for automatic substitution should any drive fail within the facility. We chose the 8+1+1 (8 data, 1 parity, 1 spare) configuration for the best transfer rate and reliability.

The Gen-4 supports RAID levels 1, 3, and 5, and the capability to partition facilities into different RAID levels. We configured the entire system as RAID level 5.

The MSI RAID achieves fault tolerance in several ways. Data reads cause an access to 8 modules. A read that fails on any drive (because of ECC, time-out, etc.) is retried up to five times. Successful retries result in successful reads. A soft error is then logged and its sector address optionally saved in the suspect permanent flaw table. If the data cannot be successfully read after 5 retries, the data is reconstructed using the XOR of the remaining 7 drives and the parity drive, called "parity replacement". In this case, a firm error is logged and the sector address saved in the suspect permanent flaw table. A read failure occurs when more than one firm error occurs at the same sector offset (2 or more of 9), This results in a hard error being logged.

A higher failure level is the loss of an entire disk. If, in the process of any operation, a drive fails, an immediate automatic hot spare replacement and reconstruction is initiated. This operation is transparent but requires approximately half of the bandwidth of the RAID (*i.e.*, throughput drops by 1/2 during reconstruction). Reconstruction takes approximately 15 minutes. If there are firm errors on any of the remaining drives, reconstruction will fail for those sectors and data loss occurs. With the effective system MTBF of a drive at eight months, it

is not unrealistic to imagine such a scenario. For this reason, MSI has agreed to add automatic reallocation. Automatic reallocation will map out bad sectors which cause firm errors the first time they are encountered. This will lessen the likelihood of data loss. Failed drives are easily replaced by operations staff. For a further description of the MSI RAID see [homan92]. For a discussion of Cray directions in disk technology, see [badger92] or [anderson93].

2.3 System Maintenance Console (SMC)

The SMC monitors activity on the system, supports configuration modification, and maintenance. It is accessible by a direct vt100/rs232 connection or Telnet. The SMC, while providing robust control over the RAID, is non-intuitive and cumbersome to use at times. It is time for MSI to redesign this software.

2.4 Status and Preventative Maintenance

Operations staff must perform preventative maintenance regularly. While some operations are inherently manual, others lend themselves to automation. MSI needs to automate some of these functions, such as the ones described below. While not a big problem for a few systems, a center considering the installation of a large number of systems will find it necessary to do so.

2.4.1 Daily RAID Preventative Maintenance

UNICOS Kernel Log - Inspect the UNICOS kernel log for "hdd.c" errors. These indicate problems detected on the CRI side. Look to the SMC to diagnose the problem.

Response Error Log - Accessible via the SMC, messages in the response log indicate the nature of the problem with an error code and a text description.

Real Time Status - On the Main display of the SMC (Real Time Status display) counters indicate accumulated errors (*e.g.*, soft, firm, hard).

2.4.2 Weekly RAID Preventative Maintenance

Read Scrub - Reading all sectors on the RAID is necessary to check for disk flaws. A utility program provided for this purpose can be run during production. This operation takes approximately 20 minutes per facility and should be done during periods of low activity.

Reallocation - Manual reallocation of suspected permanent flaws is necessary to prevent data loss. This operation does not use significant bandwidth.

2.4.3 Monthly RAID Preventative Maintenance

Flaw Management - The sudden occurrence of a large number of permanent flaws on a drive may indicate a failing drive. To monitor this accumulation, one must download the information to a 3 1/2" floppy, and inspect the logs on a system capable of reading DOS floppies.

3.0 Product Impressions

During the past 18 months, there have been a number of goals met, problems encountered and obstacles overcome. The appendix contains a chronological listing of these events. Consistent throughout the process of testing the MSI RAID was:

1. MSI always responded immediately to problems.
2. MSI diagnosed hardware problems rapidly and replaced boards immediately.
3. MSI added software functionality as requested.
4. MSI fixed software bugs immediately.
5. CRI would respond to problems expeditiously, in that problems were acknowledged and duplicated.
6. CRI did not experience any hardware problems.
7. CRI software functionality not added when requested.
8. CRI software bugs fixed at the leisure of CRI. Fixes for critical bugs (e.g., corrupted data) took as long as 6 weeks.

3.1 Reliability

Overall reliability of the RAID system for the first 10 months has been poor. This is due exclusively to the support and quality of the CRI driver. RAID reliability has been 100% over the two months since the last bug fix was installed.

Other than the initial board failure, the MSI hardware has been stable and reliable. A visual inspection of the boards indicates that they are well constructed and cleanly engineered. The finish work on the cabinets and other mechanical aspects of the construction is also well done. Overall I would rate the quality of the MSI RAID product as excellent.

4.0 Performance Analysis

Several tests were done to test the performance of the MSI RAID under various configurations. When possible, comparative performance numbers are provided for CRI proprietary disks. All tests were run under UNICOS 8.0. All data was generated on a dedicated machine, except those shown in figure

10, and those of the random i/o test (figures 16, 17, 18 and 19). Tests were run to simulate unix functions, applications, and administration.

4.1 Key to figures

Below is a description of the test configurations used for the results shown in section 4.2. Tests were conducted to evaluate the performance of the MSI RAID system against CRI proprietary disks and the effectiveness of combing the two.

4.1.1 Filesystem types

RAID-H - this "hybrid" filesystem was composed of two slices, a primary and a secondary. The primary was a CRI DD60 and the secondary was one facility of an MSI RAID (approximately 24 gigabytes). This configuration is such that inodes and small data blocks are allocated on the primary, while files that grow over 65k are allocated on the secondary. This feature of the CRI software is extremely useful in enhancing the performance of the MSI RAID. As testing will show, small block transfers on RAID are slow compared to the proprietary CRI DD60 disks. Peak performance of the DD60 is approximately 20 megabytes/second while that of the MSI RAID is approximately 80 megabytes/second. The following mkfs command was used to create the filesystem:

```
mkfs -A 4 -B 65536 -P 4 -S 64 -p 1 -q /dev/dsk/raid
```

RAID-P - this filesystem was composed of one primary slice, a single facility of the MSI RAID. All inodes and data blocks are stored directly on the MSI RAID. Peak performance of the MSI RAID is approximately 80 megabytes/second. The following mkfs command was used to create the filesystem:

```
mkfs -A 64 -B 65536 -P 16 -q /dev/dsk/raid
```

DD60-SP - This filesystem consisted of 2 primary slices. Each slice was composed of 4 DD60 drives that were software stripped via UNICOS. This was used as the gold standard against which to measure others. Peak performance is approximately 80 megabytes/second.

DD42-P - This filesystem consisted of one primary slice, a portion of a DD42 disk subsystem (approximately 8 gigabytes). The DD42 is based on a previous generation technology, the DD40. Big file throughput should be no better than 9 megabytes/second.

4.1.2 Idcaching

Several tests were run to show the benefit of ldcache on filesystem operations. Three levels of cache were used. The first level, was *no*, which simply means that there was no cache

used in the test. The second level of cache was *sm*, which consisted of 20 ldcache units, where the size of the cache unit was 128 for *RAID-H* and *RAID-P*, 92 for *DD60-SP*, and 48 for *DD42-P*. This resulted in a total amount of ldcache of 10.0 megabytes for the *RAID-H* and *RAID-P* filesystems, 7.19 megabytes for the *DD60-SP* filesystem, and 3.75 megabytes for the *DD42* filesystem. The objective of the *sm* cache was to provide a minimal amount of buffer memory so that commands could be more effectively queued and/or requests coalesced, if the OS was so inclined. The third level of cache was *lg*, which consisted of 1438 cache units for the *RAID-H* and *RAID-P* filesystem, 2000 cache units for the *DD60-SP* filesystems, 3833 cache units for the *DD42-P* filesystem. The cache unit size was the same as that of the *sm* cache configuration. This resulted in approximately 719 megabytes of ldcache for each of the 4 filesystem types. The objective of *lg* cache was to check for anomalous behavior. Figures 3 through 8 and 16 through 19 are annotated along the x-axis at the base of the bar graph to indicate the ldcache level associated with the results.

4.1.3 Test Filesystems

In tests for the *fsck*, *find* and *ls -lR*, two different data sets were duplicated over the 4 different filesystems described in section 4.1.1. The *smfs* filesystem consisted of 11,647 files, and 639 directories, totaling 1 gigabyte of data. The *bigfs* filesystem consisted of 33,051 files and 2,077 directories, totaling 3 gigabytes of data. Figure 1 shows the distribution of files and their sizes. *Smfs count* and *bigfs count* graph lines represent the distribution by size as a function of the running total percentage of the total number of files. *Smfs size* and *bigfs size* represent the distribution by size as a function of the running total percentage of all outstanding bytes. The data show that a large

number of small files occupy a small portion of space used. This mimics the home directory structure. In fact, the *bigfs* was a copy of our NAS C90 /u/va filesystem. Figure 9 is annotated along the x-axis at the base of each bar graph to indicate the file data set associated with the results.

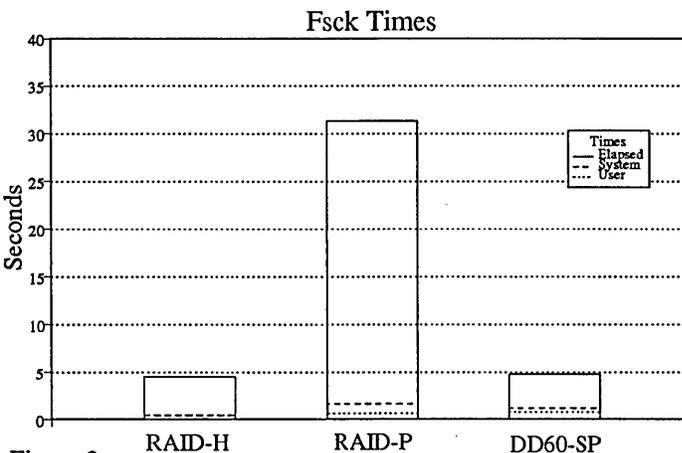


Figure 2

4.2 Tests

4.2.1 /etc/fsck

Several tests were run to compare the difference in time required to perform filesystem checks. */etc/fsck(1)* was run on the *RAID-H*, *RAID-P*, and *DD60-SP* filesystems using the *smfs* data set (figure 2). The ranking and magnitude of the results are as expected. The 6x performance differential between *RAID-H/DD60-SP* and *RAID-P* is attributed to the 3x average latency of the RAID (25 ms) and the 4x sector size (64k). Of interest is

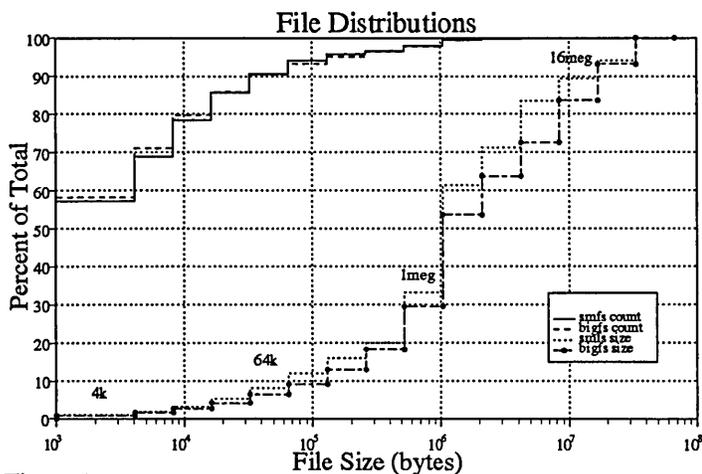


Figure 1

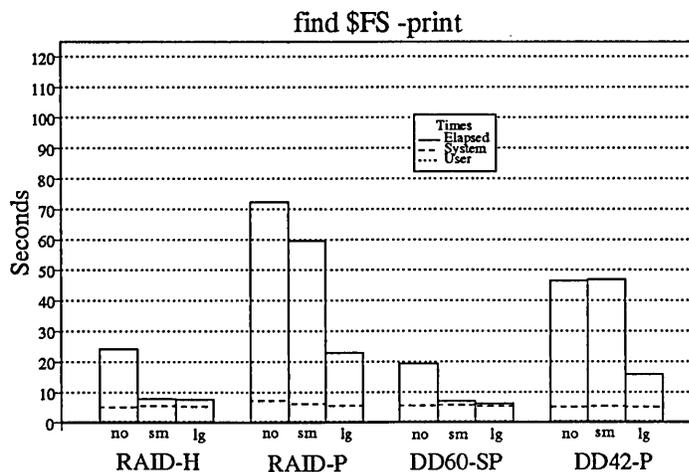


Figure 3

how effectively the *RAID-H* filesystem is able to take advantage of the DD60 primary partition. */etc/fsck* times are further analyzed in the section below that compares *bigfs* vs. *smfs* performance.

4.2.2 /bin/find

A */bin/find . -print* was executed on the *smfs* data set for each of the 4 filesystems at 3 ldcache levels. Shown in figure 3, we take the performance of the *DD60-SP* filesystem at face value. It is interesting to note the factor of 3 improvement in wall clock achieved of *sm* cache over *no* in both the *DD60-SP* and *RAID-H* filesystems. An additional option to ldcache to cache filesystem metadata only is justified from these results. The *RAID-P* filesystem is showing one of its weaknesses here in the greater latency that cannot be amortized with small block reads. A 3x margin at the *no* cache level stretches to 6x for the *sm* cache. A large amount of cache is effective only for the *RAID-P* and *DD42-P* filesystems. Again, note the effectiveness of *RAID-H*.

4.2.3 /bin/lis -IR

A */bin/lis -IR* was executed on the *smfs* data set for the *RAID-H* and *RAID-P* filesystems with 3 different cache levels (figure 4). The results are quite similar to the */bin/find* results above, except that the *sm* ldcache has much better performance for the *RAID-P* filesystem. Confusing is the observation that the */bin/lis -IR* test requires more processing than the */bin/find . -print* test, and its execution time on *RAID-H* bears this out. However, the *RAID-P* test completes in less time in all cases!

4.2.4 /bin/dd

This test was run twice, once to write a 1 gigabyte file to the filesystem under test, and once to read a 1 gigabyte file from the filesystem under test. The source for the write test and the destination for the read test was an SSD resident filesystem (RAM disk). In figures 5 through 8, the transfer rate in megabytes/second is shown along the top of each bar graph. The block transfer size for each of the tests was 16 megabytes.

Figure 5 shows the performance achieved while writing to the filesystem under test. The *RAID-P*, *DD60-SP* and *DD42-P* configurations performed at expected levels, however the *RAID-H* filesystem showed dramatic performance degradation when ldcache is used. This clearly indicates a problem which needs to be addressed by CRI.

Figure 6 shows the performance achieved while reading from the filesystem under test. Each configuration performed at satisfactorily close to the peak sustainable rate for the underlying

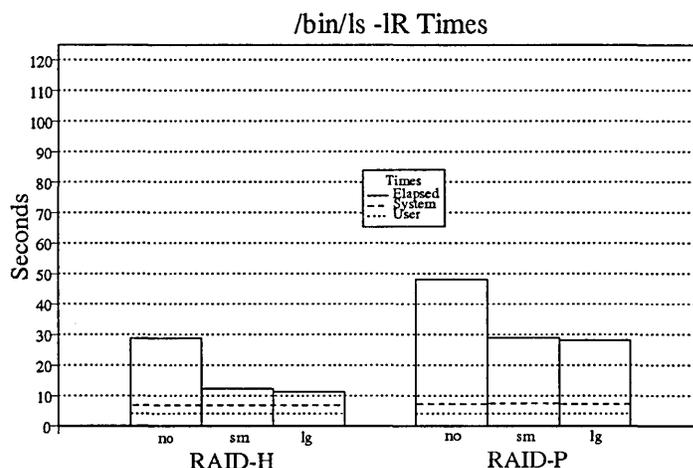


Figure 4

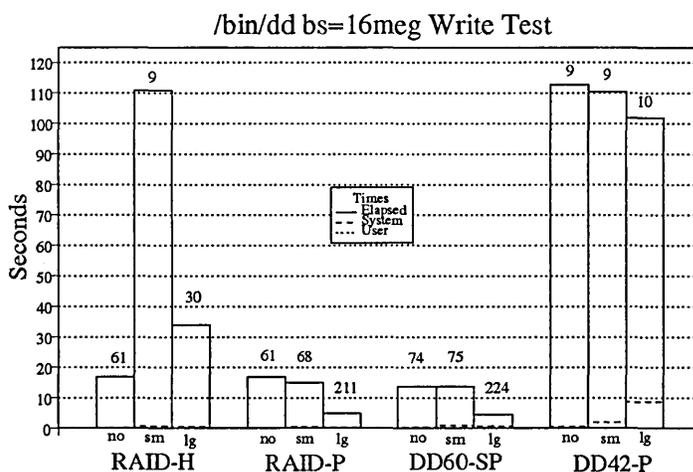


Figure 5

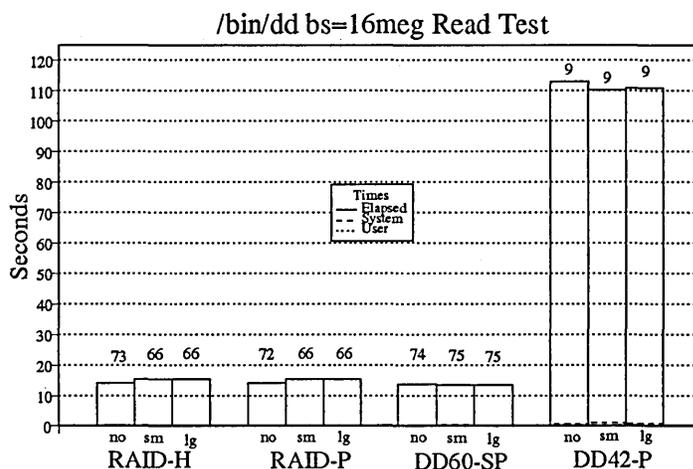


Figure 6

hardware, although *sm* cache degrades *RAID-H* and *RAID-P* performance by 10%.

4.2.5 /bin/cp

This test was run twice, once to write a 1 gigabyte file the filesystem under test, and once to read a 1 gigabyte file from the filesystem under test. The source for the write test and the destination for the read test was an SSD resident filesystem.

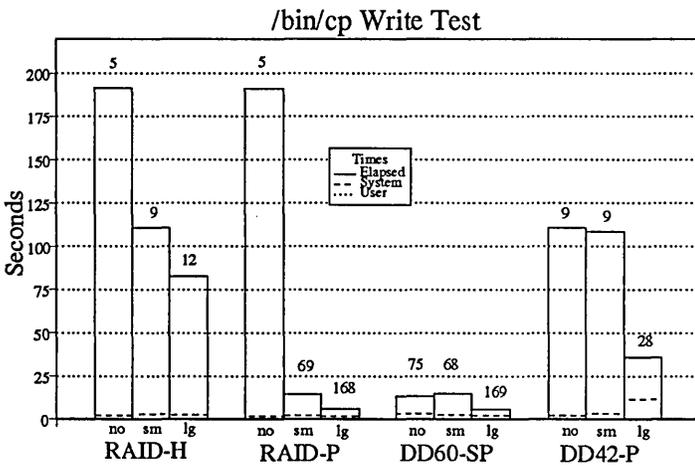


Figure 7

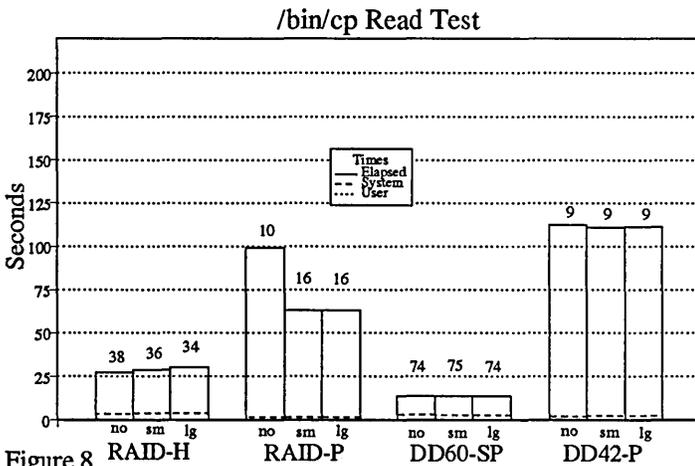


Figure 8

Figure 7 shows the performance achieved while writing to the filesystem under test. A combination of factors, a 32k i/o library buffer size, inability of the kernel to coalesce or otherwise optimize embarrassingly sequential requests, and the 25ms average latency of the MSI RAID system all lead to extremely poor performance for both the *RAID-H* and *RAID-P*

tests with *no* cache. Accounting for the latency, the performance of the *no* cache *DD60-SP* exceeds expected performance. This leads to the conclusion that UNICOS is performing additional optimization beyond that which is done for the RAID. These shortcomings can largely be overcome with *sm* cache on the *RAID-P* filesystem. In particular, this shows the benefit of write-behind optimization, boosting the *RAID-P* performance from 5 megabytes/second to over 69 megabytes/second. Consistent with the *dd* write test is the severe performance problem with *sm* and *lg* cache with *RAID-H* performance at 10% of its potential. Again the CRI proprietary filesystems *DD60-SP* and *DD42-P*, perform well.

Figure 8 shows the performance achieved while reading from the filesystem under test. The data clearly shows the optimization effort which CRI has put into their proprietary disks systems. At the other end of the spectrum, the worst performing filesystem was *RAID-P*. Although performance is somewhat enhanced with a small amount of ldcache, it is still far below what should be obtainable from the device. Doing far better is *RAID-H* which performed at about 1/2 of its potential, but could be substantially improved with read-ahead optimization. The fact that the *RAID-H* filesystem outperforms the *RAID-P* filesystem is very interesting, given that the file primarily resides on the RAID disk in both tests. Is the OS possibly doing read-ahead here or is it simply inode/extent caching?

4.2.6 Bigfs vs. smfs

This test attempts to gauge the relative increase in time required to perform certain operations, based upon the size and number of files and directories in a filesystem.

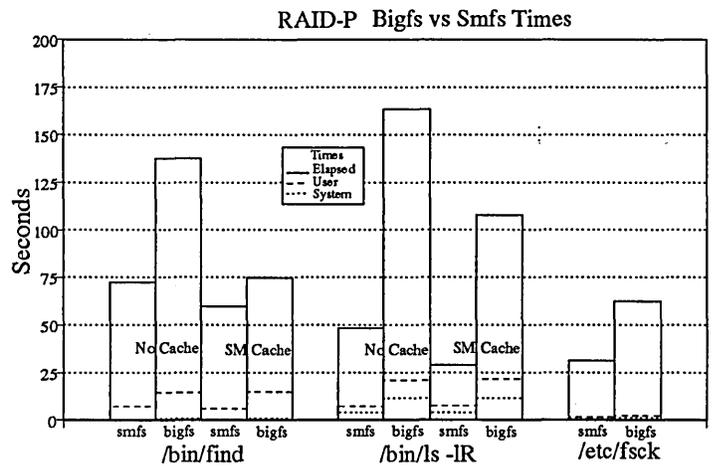


Figure 9

All tests were run on *RAID-P* with one set under the *smfs* file/directory collection and the other under the *bigfs* file/directory collection.

Figure 9 shows run times for a */bin/find . -print, /bin/ls -lR ., and /etc/fsck -u*. The results are consistent with those shown in figures 2, 3 and 4. The application of a small amount of ldcache gives some benefit, which is also consistent with other results. The caveat being that the performance of *RAID-P* is still between 1/6 and 1/3 of the performance of *DD60-SP*.

The increase in elapsed time for the */bin/find* test when going from the *smfs* to the *bigfs* tests with *no* cache is somewhat less than expected since the increase in complexity between the two data sets is about 3. The */etc/fsck* and */bin/ls -lR* test complete in about the expected time.

Because of the comparable performance on these and other tasks between *DD60-SP* and *RAID-H*, it would be most advantageous to utilize *RAID-H* in production.

4.2.7 Well formed vs. Ill formed I/O test

The next series of figures (10-15) contrasts the significant performance differential for applications that make i/o requests on boundaries that map well to the allocation unit of the device and those that do not. Each figure consists of 4 graph lines, a read and write request that is well formed, and a read and write request that is ill formed. Well formed requests in this case are successive synchronous sequential i/o accesses with a block size of 2^n where n ranges from 15 (32k) to 24 (16 megabytes). Ill formed requests are also successive synchronous sequential i/o accesses with a block size of $2^n + 1024$ where n ranges from 15 (33k) to 24 (16 megabytes + 1024 bytes). The blocks are

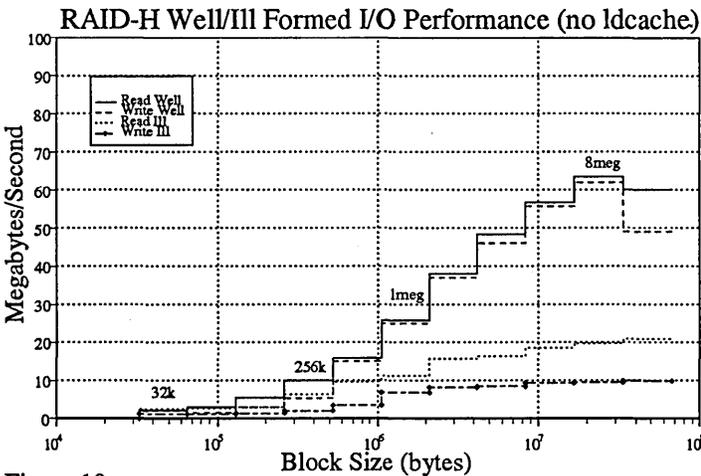


Figure 10

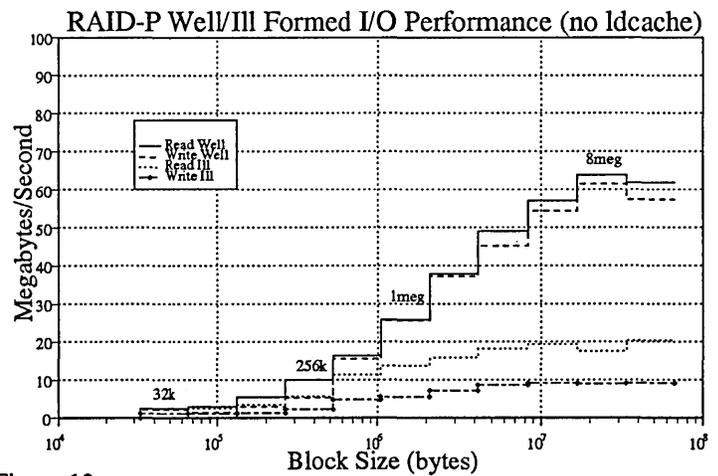


Figure 12

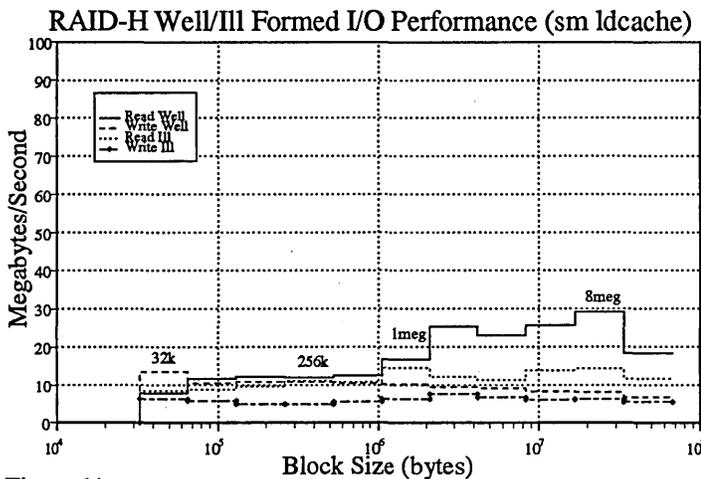


Figure 11

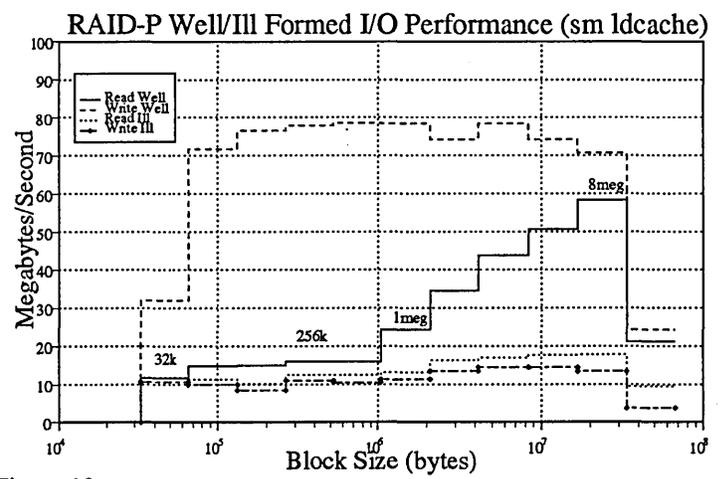


Figure 13

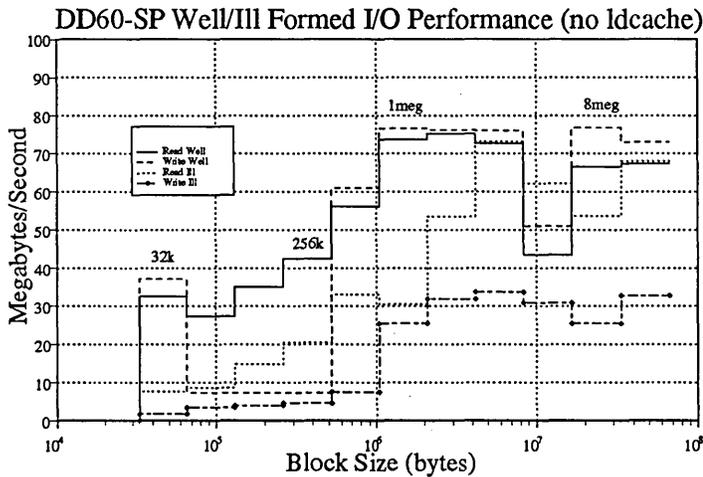


Figure 14

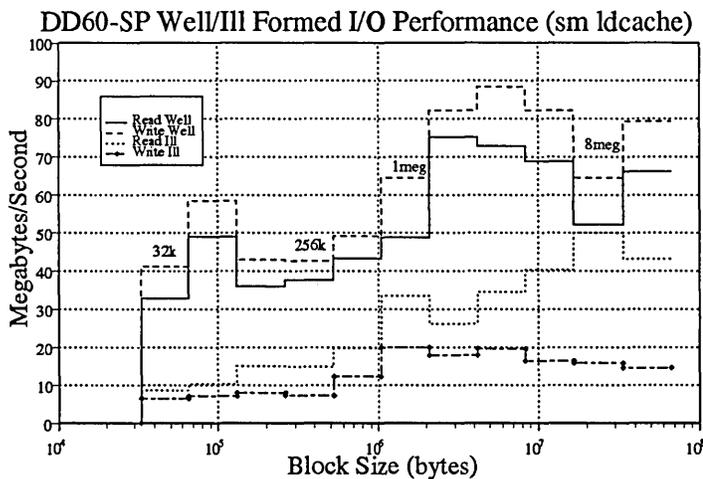


Figure 15

passed directly to the system using the read(2) and write(2) system call. The test case results are shown for block size 2^n by the horizontal line running from 2^n to 2^{n+1} . Block size reference points are shown along the *read well* graph line.

Figure 10 shows the *RAID-H* filesystem with no ldcache. Without the benefit of read ahead or write behind, these graphs depict worst case performance for the range tested. In fact, due to CRI's implementation, the performance results should be identical for an i/o test that was random instead of sequential (see section 4.2.9 on random i/o test results)! Also shown is the dramatic degradation (a factor of 2 to 3) that occurs with the ill formed requests.

Figure 11 shows the *RAID-H* filesystem with *sm* ldcache. The results are consistent with those in figures 5 and 7 in that there

is an apparent problem in using ldcache effectively with *RAID-H*. In all instances, performance is lower than expected and worse than those obtained from *RAID-P* (figure 13).

Figure 12 shows *RAID-P* with no ldcache. The results are consistent with those of *RAID-H* (figure 10) again representing throughputs indicative of a random i/o test. It is confounding that CRI would insist that read ahead and write behind would not be advantageous (see June 17, 1993 entry in the chronology appendix).

Figure 13 shows *RAID-P* with *sm* ldcache. The *sm* ldcache configuration provides an insight into what write behind optimization can actually do. For 64k byte transfers, *RAID-P* shows a remarkable 72 megabytes/second. Write behind allows for stacking commands in the MSI RAID controller and allows the application to continue on asynchronously to the i/o processing. Similar performance gains are possible from read-ahead optimization, which could be implemented utilizing ldcache. In fact, contrasting figures 12 and 13 shows this. For all transfers (read/write, well/ill) less than a megabyte, utilizing ldcache boosts performance by as much as a factor 10 for reads and a factor of 40 for writes. To better illustrate the problem for reads, the amount of time required to return 32k to an application is approximately 0.022 seconds. It takes 0.039 seconds to return 1 megabyte to an application. For a 56% increase in time, a 32 fold increase in data is achieved.

Consistent in figures 10 through 13, is the unexplained degradation when going from 8 to 16 megabyte transfers.

Figures 14 and 15 are included for completeness and show that CRI disks are also subject to degradation with ill formed requests but to much less an extent.

4.2.8 Workload Test

Table 2 shows the results of a simulated workload run on *RAID-P* and *RAID-H* configurations. Both were configured with 170 units of ldcache at a size of 128. The test consisted of the simultaneous execution of 12 data streams all on the tested

TABLE 1.

	Rate mby/sec	User Time	System Time	Megabytes Transferred	Duration (Hours)
<i>RAID-H</i>	15.51	129.65	100.89	66,998	1.20
<i>RAID-P</i>	51.78	410.66	284.98	210,634	1.13

filesystem. Block transfer sizes ranged from 32k to 8 megabytes. The ratio of well-formed to ill-formed i/o requests was 6 to 1. The tests were run over the period of approximately 1

hour. With the *RAID-P* filesystem, good performance is maintained even with the mixing of small and large requests. Again, the problem of ldcaching the *RAID-H* filesystem is apparent

4.2.9 Random I/O Test

This test was added in an effort to highlight how much better CRI's DD60 disks were at handling random small block transfers than the MSI RAID. Given that there is a 2x to 3x difference in average latency, one would expect quite a performance differential. As it turns out, this may be the *coup de grace* for SLED's.

Dedicated machine time was unavailable to run these tests. To report best case results for each configuration, each test was run 10 times and the test yielding the least elapsed time was selected as representative.

This test takes a randomly generated set of numbers which are offsets into a 64 megabyte test file. The range of the numbers potentially span the entire file. The program reads in a number, calls `lseek(2)` to position itself in the file to the specified offset, and then either does a read or write (depending on the test) of *block-size* bytes at that offset. This is done 1024 times for each execution of the test. The test was run for a *block-size* of 4096, 16384, and 65536. The test was executed against *RAID-H*, *RAID-P*, and a filesystem created on a single DD60.

Two different sets of input `lseek` numbers were used. One set was completely random in that the offset into the file could be at any byte address. These are the "Off Boundary" results shown in figures 16 and 17. The other set was also random, but the offset was restricted to be an integer multiple of the *block-size* used in the test. These are the "On Boundary" results

shown in Figures 18 and 19. *Block-size* is annotated at the base of each bar graph.

Figure 16 shows the read results for the off boundary test. With the sector size of the MSI RAID at 64k, the 4k and 16k tests are likely to require access to only one sector. The 64k test will likely require access to 2 sectors, which justifies the additional time required for the operation. The *sm* cache is quite useful in this test in negating this effect while only slightly degrading the 4k and 16k results. Most unexpected are the results of the *DD60* test. Averaging the 4k, 16k and 64k results and comparing the best RAID configuration against the best *DD60* configuration, the *DD60* only performs 28% faster than RAID! For some reason, the CRI disks are taking at best, 17ms to return a 4k block. This has not been a substantial problem in production, as no one has noticed either here or at CRI. Since most file accesses are sequential [ousterhout85], sequential performance tends to dominate, thus latency from random i/o may not be a problem, which would imply that optimizing sequential access is most important. The primary advantage that CRI disks have over the RAID is in filesystem metadata access (e.g., inodes) which is not a factor with *RAID-H*.

Figure 17 shows the read results for the on boundary test. Cache has a typically negative (minor at that) effect on the results. This is expected in that the on boundary tests only require access to one sector (except the *DD60* 64k test that requires 4). Here, the *DD60* is only 16% faster than *RAID-P*.

Figure 18 shows the write results for the off boundary test. Notice that the y-axis scale has been increased to 90 seconds for these tests. The results are expected with the 64k RAID tests requiring up to two read/modify/write operations for each

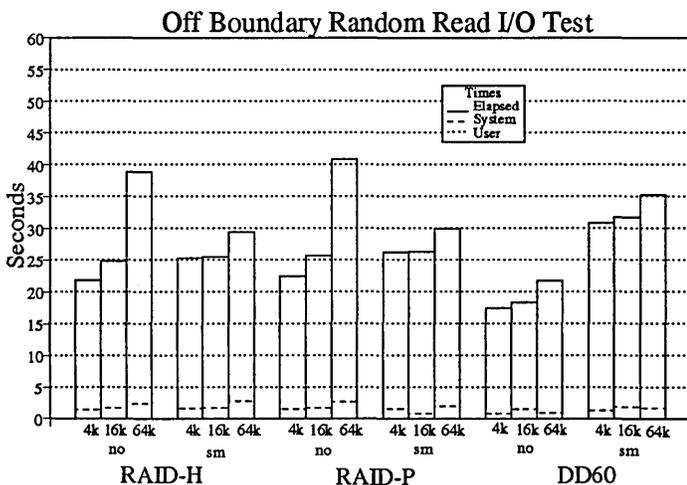


Figure 16

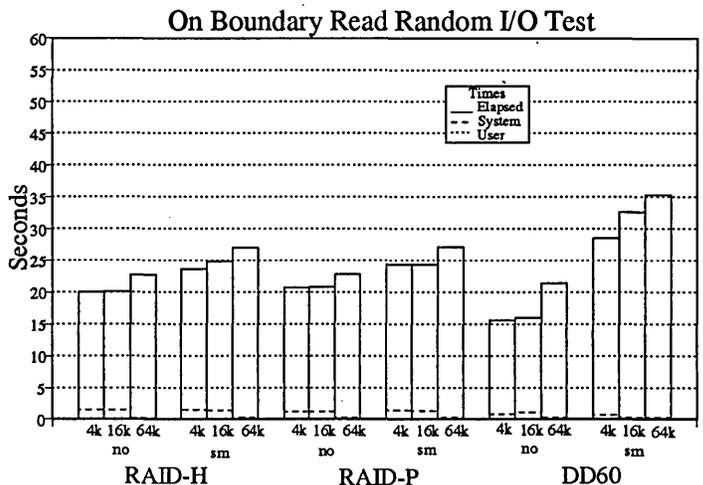


Figure 17

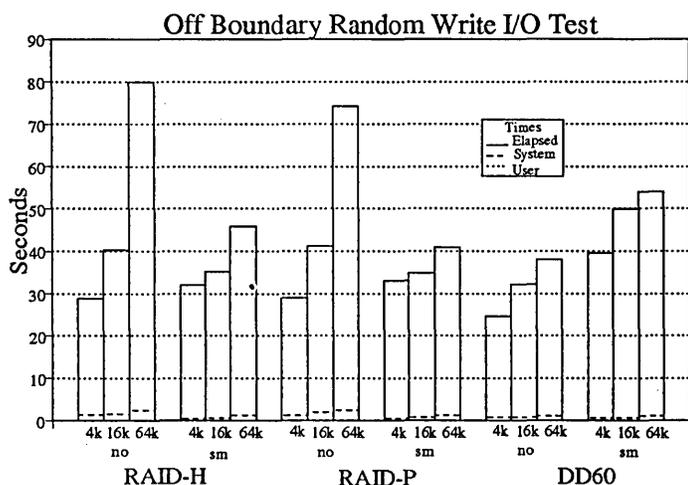


Figure 18

user level write. The best *DD60* configuration is only 13% faster than the *RAID-P* in this test.

Figure 19 shows the write results for the on boundary test. For both RAID based filesystems, *sm* cache significantly improves performance at the 64k level. The 4k and 16k tests are as expected because of the read/modify/write that occurs with transfers less than 64k on the RAID based systems. Note also that the RAID based filesystems are 3x faster than the *DD60* for the 64k test! Overall the *DD60* is only 32% faster in this test.

Given this information, one can say with some certainty, that the DA60 RAID product from CRI which is a RAID level 3 system with a 64k sector size, should perform worse than the *DD60* used for this test, especially with the 16k test from figure 19, which is the only test that the *DD60* was significantly better.

The overall random i/o performance advantage that the best *DD60* configuration (*no* cache) has over the best RAID configuration (*RAID-H sm* cache) is 24% faster for writes and 29% faster for reads.

4.2.10 Suggested Additions

On the MSI Side:

- Reallocation - As previously discussed, data loss can occur when a drive fails and there are bad sectors on other drives. Having the capability to automatically reallocate bad sectors on the fly would all but eliminate this potential for data loss.
- SMC - The addition of a mode that the SMC can be placed into that prevents any unintended modification of opera-

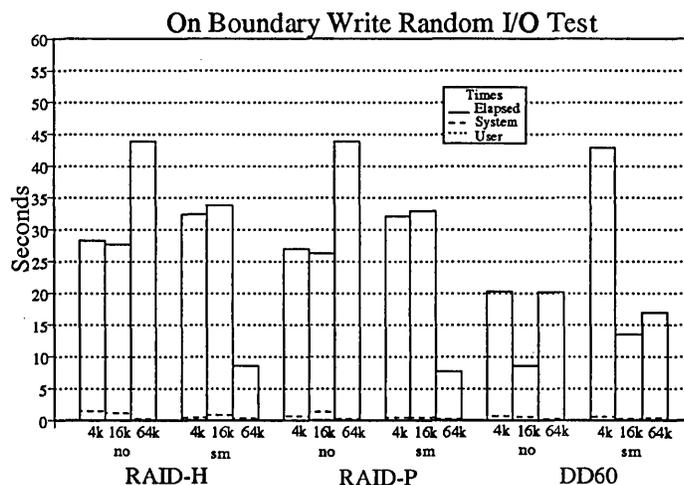


Figure 19

tional parameters. As is stands now, anyone who has physical access to the SMC effectively has unlimited power to change anything at will, or by mistake.

- Remote Status - A command that could be run from a remote workstation that would tell an operator whether or not someone actually needed to take further action. This could then be put into a cron script to status the system several times per day and fire off email if a problem is indicated. This could be extended to for such maintenance activities as read scrub and flaw management.
- Rewrite SMC software - This software needs to be rethought. Many common operations are not intuitive and or awkward. For example, it takes approximately 200 key-strokes to examine suspect permanent flaws.

On the CRI Side:

- Buffer Alignment/grouping - As shown in the performance section, some operations performed utilizing *ldcache* cause unexplained and significant degraded performance, most notably with the *RAID-H* configuration.
- Read Ahead into *ldcache* - When read-ahead is indicated, perform the read-ahead into *ldcache*.
- Default Buffer Sizes - Evaluate the potential for enhancing performance by modifying the default buffer size for library routines, and elsewhere that it is indicated. Matching this to better suit the MSI RAID should help to improve performance while requiring only a modest increase in the amount of main memory.
- Metadata Cache - Add an option to *ldcache* so that filesystem metadata and data can be cached separately. A metadata write-through cache should improve performance

without degrading filesystem integrity when recovering from crashes.

- Metadata Access - The results of the random i/o test indicate that the DD60 outperforms the RAID by 25% to 30%. Why then are the /bin/find times 3x and the /etc/fsck times 6x that of a DD60?

5.0 Summary

It has taken much longer than expected to get the MSI RAID up to production quality standards. There are still some performance problems that need to be addressed. Overall, the performance and reliability of the MSI RAID system is good and it is certainly the least expensive high performance alternative.

The RAID-H filesystem is advantageous for several reasons. Combining this configuration with some amount of ldcache is the desired configuration for NAS, given that the ldcache buffer problems can be resolved.

The results from the random i/o test indicate that CRI proprietary disks work better primarily because UNICOS optimizes their access. The same optimization techniques applied to the RAID should bring all performance to within 10% to 20% of the DD60-SP configuration. They also show that DD60s do not significantly outperform the RAID. It then follows that questions concerning RAID latency cannot serve as an excuse to prevent optimization efforts any longer. Direct hardware support (e.g., eliminating IPI-3) would all but eliminate the small DD60 performance advantage.

RAID technology is already an attractive option. With fast SCSI-2 drives approaching \$0.50 per megabyte, there is still a 10 fold markup to build a fast RAID controller that integrates commodity disks and supercomputers, leading to the expectation of even lower prices.

6.0 Acknowledgments

I would like to thank Bob Cave and the others involved in the IDA Princeton project that proved this type of technology was usable in a supercomputer environment. Ken Katai and Neal Murry of MSI also deserve recognition for providing excellent support of their Gen-4 system.

7.0 Appendix

Over the past 18 months, there have been a number of goals met, problems encountered and obstacles overcome. This is a

partial chronological listing of these events and their eventual outcome.

- Nov 5 1992 - Product Performance Demo
As a requirement for the procurement of the RAID, the potential vendors were required to demonstrate performance. The testing was performed at the Maximum Strategy facility in Milpitas, CA. The test environment consisted of a Gen-3 RAID serving as the testing client and the IFB bid hardware consisting of a Gen-4 controller and 20 Seagate IPI-2 drives with a storage capacity of 27 gigabytes. Results shown in Table 2
- Mar 25 1993 - First Installed
HSP-3 (C9016-1024/256) installed. Cabled up MSI RAID. Able to access RAID with the alpha release of the CRI IPI-3/HiPPI driver. Having problems talking to more than one facility.
- Mar 31 1993 - Software Problem
Due to a minor device number conflict, we are unable to access both facilities of the MSI RAID. Ldcache give an ENXIO (errno 6) error when trying to ldcache a RAID filesystem.

TABLE 2.

	Requirement	MSI Gen-4
read	720 mbit/sec	740 mbit/sec (92.4 mby/sec)
write	680 mbit/sec	701 mbit/sec (87.6 mby/sec)

- April 9, 1993 - Hardware Problem
Data corruption is detected on reads, further diagnosis shows a parity error over HiPPI. Replaced HiPPI cable and MSI HiPPI controller board.
- April 12, 1993 - Hardware Problem
More data corruption errors on read. MSI replaces HiPPI controller board again. Diagnosis at MSI shows a hardware failure on the first replacement board.
- April 23, 1993 - Software Problem
Ldcaching on the MSI RAID now crashes the C90.
- April 28, 1993 - Software Problem
Utilizing the primary and secondary allocation on RAID causes the root filesystem to hang (i.e., ls -l / never returns) and eventually requires a re-boot to fix.
- May 19, 1993 - Software Problem
Inappropriate handling of soft errors on the CRI side. When a conditional success is sent to the Cray, it is interpreted as a failed request, The Cray then does this 5 times and quits -

propagating the problem to the application. It is suggested that the Cray attempt some type of error recovery.

- Jun 9, 1993 - Software Problem
Ldcache still not working. Primary/secondary allocations also still not working. No error recovery is done. No read ahead/write behind. I/o requests must be VERY well formed to extract performance from the RAID disk.
- Jun 15, 1993 - Software Fix
Kernel mod installed to fix ldcache problem.
- Jun 17, 1993 - CRI Response to issues
Primary/secondary allocations are not supported in the current HiPPI/IPI-3 implementation under UNICOS 7.C.2. This capability is available in Unicos 7.C.3, which is currently scheduled for release August 6th.
CRI declines to do any error processing (other than retries) on the C90 side. They do make a reasonable argument. The issue of read ahead/write behind for an IPI-3 driver came up during the HSP-3 contract negotiations. Cray Research replied in a letter dated November 4, 1992:

“Cray Research has investigated implementing read-ahead and write-behind in either the mainframe itself or in the IOS and believes that such an implementation would be ineffective in enhancing performance of a HIPPI-based IPI-3 driver. This is because both the mainframe and the IOS are too far away from the disk device itself to provide meaningful improvement in transfer rates. The appropriate place to put read-ahead and write-behind, in our view, is in the controller of the RAID device itself. This has not yet been done in Maximum Strategy products.”

- July 12, 1993 - Upgraded UNICOS
Primary/Secondary mods from 7.C.3 are added into 7.C.2 in an attempt to create the hybrid filesystem.
- July 19, 1993 - Fell back to plain 7.C.2
System time the has increased greatly. Experiencing lost mount points with mixed device filesystems. Returning to unmodified 7.C.2 system. Since we will be beta testing UNICOS 8.0 in August, no upgrade to the official 7.C.3 release is planned.
- Sep 17, 1993 - Software Problem
Duplicated the auto-reconstruct failure that occurred 2 weeks ago after several tries. Occasionally when powering off a drive, multiple retry failures cause an EIO (errno 5 - i/o error) to be propagated to applications. Two problem are apparent here;

1.CRI driver is not appropriately handling conditional success errors.

2.When a drive fails AND there are active flaws on other drives, correct data cannot reconstructed and the read fails. A request is made to MSI to add the capability to automatically reallocate suspected permanent flaws that occur during operation.

- Sep 28, 1993 - Software Fix
New driver available to fix the inappropriate handling of conditional success status.
- Sep 30, 1993 - Software Problem
Installed new CRI software and New MSI software to fix all current known problems. On the positive side, performance increased by almost 30% with the new software (80 mby/sec reads and 73 mby/sec writes). Testing has however uncovered a serious problem that causes corrupted data to be propagated to applications when a drive is powered off.
- Oct 1, 1993 - Response
MSI and CRI are investigating the problem.
- Oct 1, 1993 - Software Problems
Duplicated data corruption problems without powering off drives.
- Oct 7, 1993 - UNICOS 8.0
Began beta UNICOS 8.0 testing, primary/secondary allocations are still not operating correctly.
- Nov 17, 1993 - Software Fix
Installed a new IOS and a new HiPPI driver on the CRI side and a new driver on the MSI side Nov 10. Testing over the last week has not turned up any problems. Performance has dropped somewhat (about 10%) for both reads and writes
- Dec 3, 1993 - Production UNICOS 8.0
Primary/Secondary allocations functioning correctly. Performance is mixed yet consistent with 7.C.2/3.
- Jan 2, 1994 - Hardware Problem
Machine powered off for several days during facility maintenance, when power returned, RAID will not boot. MSI able to give instructions over the phone to return system back on-line. Problem traced to battery failure that has been fixed in subsequent systems. MSI provides upgraded system processor board.
- Jan 30, 1994 - Limited Production
Extensive testing has turned up no further problems with the MSI RAID. The system will now be put into limited production
- Mar 10, 1994 - Every Thing OK
No errors reported. No outstanding problems.

8.0 References

- [anderson93] Anderson, "Mass Storage Industry Directions," Proceedings of the Cray User Group, Montreux Switzerland, Spring 1993. pp307-310.
- [badger92] Badger, "The Future of Disk Array Products at Cray Research," Proceedings of the Cray User Group, Washington D.C., Fall 1992. pp177-190.
- [cave92] Cave, Kelley, Prisner, "RAID Disk for a Cray-2 System," Proceedings of the Cray User Group, Berlin, Germany, Spring 1992. pp126-129.
- [cooper93] Cooper, et al, "Numerical Aerodynamic Simulation Program Plan," NASA Communication, September 93.
- [homan92] Homan, "Maximum Strategie's Gen-4 Storage Server," Proceedings of the Cray User Group, Washington D.C., Fall 1992. pp191-194.
- [ousterhout85] Ousterhout, Da Costa, Harrison, Kunze, Kupfer, Thompson, "A Trace Driven Analysis of the UNIX 4.2 BSD File System," ACM Operating Systems Review, Vol 19, No. 5 (1985), pp15-24.
- [patterson87] Patterson, Garth, Gibson, Katz, "A case for Redundant Arrays of Inexpensive Disks(RAID)," University of California Technical Report UCB/CSD 87/391, Berkeley CA, December 1987, preprint of [patterson88].
- [patterson88] Patterson, Garth, Gibson, Katz, "A case for Redundant Arrays of Inexpensive Disks(RAID)," Proceedings of the 1988 ACM SIGMOD Conference on Management of Data, Chicago IL, June 1988, pp109-116

Automatic DMF File Expiration

Andy Haxby (andy@trumpet.demon.co.uk)

SCIS - Shell Common Information Services

(Formerly Shell UK Information and Computing Services)

Wythenshawe, Manchester M22 5SB England

Abstract

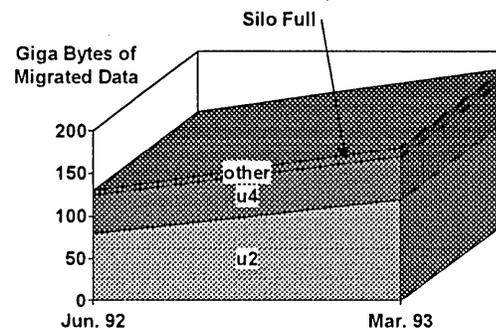
SCIS has a YMP/264 (until recently a XMP-EA/264) with a 4000 slot STK 4400 silo that is mainly used to hold DMF and backup tapes. Since running DMF users have tended to regard the file systems as an infinite resource and have been very lax about deleting unwanted files. Eighteen months ago it was apparent that our DMF pools would grow beyond the capacity of our silo, so it was necessary to implement a file expiration date for DMF that could be set by users on their files. The system has significantly reduced our DMF pools. This paper describes the external and internal workings of the file expiration system.

Background

SCIS is a Royal Dutch Shell Group company that provides a range of computing services to other members of the Shell group in the UK and the Netherlands. The SCIS Cray is used for petroleum engineering applications, mainly reservoir modelling. A typical model will consist of a 10MB executable and a number of data files each up to 100MB. There are approximately 400 users in 30 different groups. The majority of users are located at various sites around the UK, but a diminishing number are in locations as far apart as Canada, New Zealand and the Middle East. Many users access the Cray infrequently and only by batch, and the large geographic spread of users makes it very difficult to manually ask people using large amounts of storage to delete unwanted files.

For the purpose of disk organisation users are split into two 5GB file systems, /u2 and /u4, depending on whether they are UK or non-UK based. DMF is run on these file systems and a few others such as the support staff's file system and a file system used for archiving old log files and accounting data. In July 1992 the DMF tape pools had grown to 900 cartridges each. The rate of growth was linear, consistent with users not deleting old files, and indicating that we would run out of space in the silo early in 1993. It was clearly necessary to implement a system whereby files would be deleted after a pre-

determined length of time (a 'retention period') unless a user had explicitly requested that a file should be kept for longer.



Requirements For The Automatic Expiration System

It was considered important that the system must:

- Be easy to use.
- Require little or no maintenance or administration once installed.
- Require the minimum amount of local code.
- Integrate cleanly with Unicos, i.e. have a Unix flavour interface and be completely application independent.
- Not compromise any existing security mechanism, i.e. users can only set retention periods on files they have write access to.
- Work with subsequent versions of Unicos. At the time we were running Unicos 6.1.6 and 7.0 was still in Beta.

Design Considerations

Two routes to achieving the requirements were considered:

- A 'database' type system whereby users execute locally written commands to read and write lists of files into a database. The database could then be regularly interrogated by a cron job that would delete any files that had passed their 'best before' date. This system would require no modifications to Unicos source to write, and if it went wrong it would not interfere with Unicos itself. However, a large amount of local code would be needed to check file permissions and ownerships, cope with files restored from backup, delete the database entry if the file is deleted by the user rather than the system, etc.
- Modify Unicos commands to provide the tools to enable a simple file expiration facility to be written. The inode contains three time stamps, `cdi_atmsec`, `cdi_ctmsec` and `cdi_mtmsec` (`atime`, `ctime` and `mtime`) which correspond to date last accessed, date inode last modified and date file last modified respectively. Under Unicos 7.0 and above there is a site modifiable member of the inode structure, `cdi_sitebits`, which can be written to with `fctl(2)` and read with `stat(2)` system calls. The sitebits inode member can be used to store an additional time stamp corresponding to how long the file should be kept for. Commands such as `touch(1)`, `ls(1)` and `find(1)` can be modified to read and write time stamps into the inode sitebits member.

This method has the advantage of being an elegant and totally seamless enhancement to Unicos requiring little local code, and utilising all of the security checking mechanisms already coded into the commands. The disadvantages are that a source licence would always be required and the mods would have to be carried forwards to each Unicos release. Additionally, under Unicos 6.1 whilst the sitebits member was present in the inode, the `fctl(2)` and `stat(2)` system calls were missing the code to read and write to it, so kernel mods were necessary to provide this functionality until Unicos 7.0 was available.

Implementation Details

The second method of implementing a retention system was chosen. Kernel mods were made to Unicos 6.1.6 to provide the extra functionality in the `fctl(2)` and `stat(2)` system calls that is available in Unicos 7.0. Whilst doing this it was apparent that it would have been better if Cray had not implemented the facility to write into the inode sitebits member through `fctl(2)`, but had rather written another system call that would function similarly to `utime(2)`. This is because `fctl(2)` principally performs file locking and other operations on the data of a file and so takes a file descriptor as an argument, therefore the file

must first be opened with `open(2)`. `utime(2)` takes a path name as an argument and does not require that the file is open, which is more sensible since only the inode is being updated, not the data block.

The following commands were modified so that users could set and read retention times. The 'd' flag was chosen to be a mnemonic for 'detention' since the `ls` command already has 'e' and 'r' flags.

- To set a retention period the `touch(1)` command was modified:

```
touch [-a] [-c] [-m] [-d] [mmddhhmm[yy]] [-D dd]
```

where `-d` is the detention (retention) time as a time stamp and `-D` is the detention time in 'days from now'. The default for `touch` is still `-am`. Using the `-d` or `-D` options on their own does not cause the modification time to be altered. Unicos sets the sitebits member of the inode structure to be 0 on every file by default, and hence the retention time stamp for all files will be 0 by default. It is not possible to set a retention time on a directory as the `touch(1)` command must `open(2)` the file first and it is not possible to `open(fd, O_WRONLY)` a directory.

It is interesting to note an 'undocumented feature' of `touch(1)`: because `utime(2)` is used to change the `actime` and `modtime` time stamps on a file it is not possible to touch a file you don't own even if you have write permission. This is not the case for the retention time since the `open(2)` and `fctl(2)` system calls are used to update the retention time stamp.

- To enable users to list the retention time on their files both `ls(1)` and `ls(1BSD)` commands were modified to accept a `-D` option which, when used in conjunction with `-l` or `-lt` options, lists the retention time in the same format as the `-c` and `-u` options. The code generates an error message if the `-D` option is used in conjunction with either `-c` or `-u`. If the retention time is zero, i.e. has not been set on a file, the `mtime` is used in the default manner of `ls(1)` and `ls(1BSD)`. The `-D` option used on its own is valid but meaningless, as are `-c` and `-u` options.
- In order to search file systems for files that have exceeded their retention period, the `find(1)` command was modified to accept a `-dtime` argument in the manner of `-atime`, `-ctime` and `-mtime`. If the retention time is zero, i.e. has not been set on a file, the last date of modification is used in the manner of the `-mtime` option.

Sites implementing a system such as this could give consideration to modifying other commands such as `fck(1)` to

report the retention time, or perhaps `rm(1)` to warn users trying to delete files with an unexpired retention time.

The mods to commands described above provided the necessary tools with which to implement a 'file expiration' system. Users were forewarned, run stream generators for user jobs were modified to allow batch users the option of touching their files with a retention time, and a shell script was written to delete files past their retention time. The shell script could have been run from cron, but in order to enable us to mail users who only read VAX mail the script is run from a batch job submitted from a VAX front end once every month. It was decided to delete all files more than 93 days past both their retention time and date of last modification. Date of last access was not used as it is updated by commands such as `dmget(1)` `dmpu(1)` and `file(1)`. The shell script essentially just does:

```
find $FS -mtime +93 -mtime +93 -type f -exec rm {} \; / -print
```

Note that the above command leaves directory structures in place in case this is necessary for some applications to run. A list of the files deleted is mailed to the user, along with a list of files that will be deleted next month unless a new retention time is set upon them.

Effect Of Implementation On DMF

The system went live on February 1st 1993. The first deletion removed approximately 80GB of files. A month later when the files were hard deleted utilisation of our DMF tape pools was reduced by nearly 50%. Having made a large number of tapes free we wanted to remove a contiguous range of higher VSN tapes from the dmf pools rather than just removing random VSNs as they became free. This was time consuming and messy, and could only be done by setting the hold read only (`hro`) flag on the VSNs with the `dmvdbgen(8)` command, and then waiting until more files were hard deleted from the tapes before merging them. Multi-volume files that were not due to be deleted had to be manually recalled, touched to make DMF think they had been modified and so put them back somewhere else, and then re-migrated. Tidying the tape pools up after the start of the file expiration system took several months.

One initial problem was caused by the fact that the modified `touch(1)` command uses the `fcntl(2)` system call to write the retention date, and so must `open(2)` a file first. This means that if a user sets a retention time on a migrated file, the file is needlessly recalled. As a result of this there was some thrashing of tapes during the month before the first delete as users set retention periods on old migrated files. Fortunately the act of touching the retention time on the file does not update the modification time, else a subsequent `dmpu(1)` of the file would create yet another copy! On the 2nd of December

1992 Design SPR No 58900 was submitted suggesting that a new system call should be written to allow `cdi_sitebits` to be updated without recalling migrated files.

The system has been running for over a year now. It had been anticipated that some users would try to circumvent the retention system by setting very long retention periods on all their files by default, but so far this has not happened. We also anticipated being deluged by requests to restore deleted files but apart from a small number of genuine mistakes this has not happened either.

Some users do unfortunately consider the retention system to be an alternative to the `rm(1)` command and just leave files lying around until the retention system deletes them. This causes a problem because the files are held in DMF for three months before the retention system deletes them and then a further month before they are hard deleted from DMF. In the worst case this can result in garbage being held in DMF for nearly five months.

First Problems

In July '93 a disk problem caused a user file system to be restored from a backup tape. Shortly afterwards it was noticed that all of the retention periods set on files had been lost. This was due to a bug in the `restore(8)` command. The `dump(8)` command correctly dumps the value of `cdi_sitebits` but there is no code in `restore(8)` to put it back again. Dump and restore had been inadvertently omitted during testing of the retention system! Cray confirmed that the bug was present in 7.0.

Since `restore(8)` runs as 'user code' i.e. does everything through the system call interface rather than accessing the file system directly, it was not possible to fix this bug in Unicos 7.0 - restore would have to `open(2)` and `unmigrate` every restored file! Design SPR No 66926 was submitted on the 8th July 1993 against this problem. The Cray reply was that a new system call would have to be written which would not be available for some time. The Design SPR suggesting that a new system call should be used to set `cdi_sitebits` had been submitted over six months previously but no action had been taken by Cray. This reinforces a general concern of the author that Cray pay insufficient attention to Design SPRs.

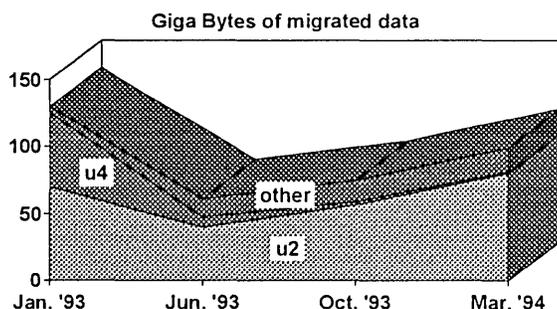
Fortunately Cray did suggest a work-around to the problem. A script was written to read a dump tape and produce a list of inode numbers and file names. A program then re-reads the dump tape and produces a list of inode numbers and `cdi_sitebits` values. The two lists are read by an `awk` program that matches path names to `cdi_sitebits` values and pipes the result into `fsed(8)` to reset the values. Finally the file system must be unmounted and mounted again before Unicos recognises the `cdi_sitebits`

values. The whole process is a kludge and requires multiple reads of the dump tapes, but it works.

Where Are We Now?

Our DMF pools are still growing and are now, at 900 tapes each, the same size as before we introduced the retention system. This is due to three main factors.

- The users in the /u2 file system have generated a large number of files which legitimately must be kept for several months. This is unavoidable and the main cause of the growth.
- We keep more log files and accounting data on the 'other' file systems than we used to.
- Systems support staff themselves are untidy and are keeping ever increasing amounts of data.



Outstanding Problems

The following problems are outstanding:

- It is necessary to unmigrate a file in order to set a retention time on it.
- Restoring a dump tape is a convoluted process.

Both of the above problems would be solved if Cray were to implement a new system call for writing into `cdi_sitebits`. According to **CRAY PRIVATE PRE-RELEASE** information (liable to change), there will be a new system call, `lsetattr(2)`, in 8.2 which will allow all user-accessible meta-data associated with a file to be set in a single system call. The system call takes a pointer to a file name and a structure containing the desired changes as arguments, and does not require the file to be open.

However, due to the internal workings of the Unicos 8.x virtual file system, the initial implementation will be

restricted to super-user. This should allow the bug in `restore(8)` to be fixed but may cause problems with our local mods to `touch(1)`: it may be necessary to make `touch(1)` a `suid` program to allow users to set retention periods. The restriction on super-user may be lifted in a later release and the functionality of `fcntl(2)` will remain indefinitely should we decide to stick with the current limitations. Unicos 8.2 is scheduled for release fourth quarter 1994.

- Unwanted files can still remain on DMF tapes for several months.

The only way to prevent this is to reduce the time after which expired files are deleted, and/or reduce the time after which files are hard deleted.

Future Plans

We plan to run the retention system on support staff file systems in the near future, and will look at reducing the amount of log files kept. These changes should remove up to 10 GB of migrated data.

Files more than one day (rather than 93 days) past their retention period will be deleted provided they are more than 93 days past the date of last modification. This change will delete 21 GB of migrated data.

Conclusions

It is very easy to implement a simple and reliable method of removing unwanted files and preventing DMF tape pools from growing indefinitely. From Unicos 8.2 existing problems with having to unmigrate files to set a retention time on them, and the failure of `restore(8)` to reset the retention times on restored files should be fixed.

The system is not, however, a magical remedy for all storage space limitations, and good storage management is still required to ensure that users (and systems staff!) do not abuse the retention system or carelessly keep unnecessarily large amounts of data.

Acknowledgements

I would like to thank: my colleague Guy Hall who wrote the code to run the monthly file deletion from the VAX and send mail to users; Neil Storer at ECMWF for assistance with the kernel mods to 6.1.6; the staff at Cray UK, in particular Barry Howling, who provided much help during early discussions about how to implement a file retention system, and who endured endless 'phone calls and complaints about design SPRs and the way Cray implemented the `sitebits` facility.

ER90[®] DATA STORAGE PERIPHERAL

Gary R. Early
Anthony L. Peterson

EMASS Storage System Solutions
From E-Systems
Dallas, Texas

Abstract

This paper provides an architectural overview of the EMASS[®] ER90[®] data storage peripheral, a DD-2 tape storage subsystem. The areas discussed cover the functionality of tape formatting, tape drive design, and robotics support. The design of the ER90 transport is an innovative approach in helical recording. The unit utilizes proven technology developed for the video broadcast industry as core technology. The remainder of the system is designed specifically for the computer processing industry.

Introduction

In 1988, E-Systems initiated a project which required a high performance, high capacity tape storage device for use in a mass storage system. E-Systems performed an extensive, worldwide search of the current technologies. That search resulted in the identification of the Ampex broadcast recorder that utilized D-2 media as the best transport. E-Systems then initiated a joint development effort with Ampex to use their proven video transport technology and design the additional electronics required to produce a computer peripheral device. The result of this development was the EMASS ER90 tape drive which connects to computers using the IPI-3 tape peripheral interface.

Core Technology

The ER90 transport design meets the stringent requirements for long media life in its approach to tape handling. A simplified threading mechanism, a simplified tape path, and automatic scan tracking along with a proven head-to-tape interface are all features that lead to selection of the Ampex transport for the ER90 drive.

The ER90 transport uses this direct-coupled capstan hub similar to high performance reel-to-reel tape drives instead of the usual pinch-roller design. Advantages include fast accelerations and direction reversal without tape damage, plus elimination of the scuffing and stretching problems of pinch roller systems. Since a direct drive capstan must couple to the backside of the tape, it must be introduced inside the loop extracted from the cassette. In order to avoid a "pop up" or moving capstan and the problems of precise registration, the capstan was placed under the cassette elevator, so that it is introduced into the threading cavity as the cassette is lowered onto the turntables.

In order to prevent tension buildup and potential tape damage, none of the tape guides within the transport are conventional fixed posts. Air film lubricated guides are used throughout; one exception is the precision rotating guide which is in contact with the backside of the tape.

All motors are equipped with tachometers to provide speed, direction, or position information to the servo system, including the gear motors which power the cassette elevator and the threading apparatus. There are no end position sensors; instead, the servo learns the limit positions of the mechanisms and subsequently applies acceleration profiles to drive them rapidly and without crash stops. This approach also permits the machine to recover from an interruption during any phase of operation without damage to the machine or tape.

The tape transport also features a functional intermediate tape path that allows high speed searches and reading or writing of the longitudinal tracks without the tape being in contact with the helical scan drum.

The ER90 tape drive architecture and media provides several unique functions that enhance the ability to achieve high media space utilizations and fast access. Access times are enhanced through the implementation of multiple areas (called system zones) on the media where the media may be unloaded. This feature reduces positioning time in loading and unloading the cassette. Access times are reduced through high speed positioning in excess of 800 megabytes per second. These core technology designs support a set of advantages unique to tape transports. Figure 1 depicts the ER90 transport and associated electronics.

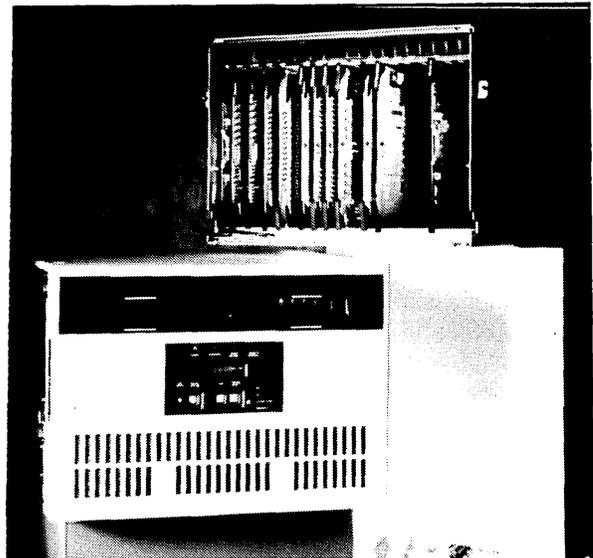


Figure 1. ER90 transport and associated electronics.

Head Life

Helical heads are warranted for 500 hours; however, experience with helical head contact time exceeds 2000 hours. Because of the system zones and the ability to move between system zones without tape loaded to the helical scanner drum, the actual head life with tape mounted on the drive may be substantially longer. In addition, when heads do need to be replaced, service personnel may quickly install new heads on-site, without shipping any transport subassemblies off-site, in about 20 minutes.

Safe Time on Stopped Tape

Whenever the flow of data to or from the tape drive is interrupted, the media is moved to a system zone and unloaded from the helical drum. When data is being written, this should be a rare occurrence because each drive has a 64 megabyte buffer. When in retrieval mode, returning to a system zone whenever the access queue is zero should be standard practice. In this way, if the drive is needed for a different cassette, it is available sooner and if another access is directed at the same cassette, the average access time is not affected by where the tape now rests. With this type of drive management, the cassette may remain mounted indefinitely without exposure to the tape or heads.

Data Processing Design

An ER90 cassette can be partitioned into fixed size units which can be reclaimed for rewriting without invalidated other recorded data on the tape cassette. Most tape management systems achieve space reclamation by deleting an entire tape volume, then allowing users to request a “scratch tape” or “non-specific” volume when they wish to record data to tape. Physical cassette sizes of 25, 75, or 165 gigabytes will make this existing process inefficient or unusable. The ER90 cassette partitioning capability provides an efficient mechanism for addressing the tape space utilization problem.

ER90 cassette formatting provides for three levels of Reed-Solomon error correction. In addition, data is shuffled across the 32 tracks that make up a physical block, and interleaved within the physical track so that each byte of a block has maximum separation from every other byte that make up an error correction code word. Data is then recorded using a patented process called Miller Squared. This process is a self checking, DC free, rate 1/2 coding process that has a 100% probability of flagging a burst error. This has the effect of doubling the efficiency of a Reed-Solomon code by knowing where the power of the code should be applied. A data rate of 15 MB/sec is achieved with all error correction applied, resulting in no loss of drive performance for maximum data reliability.

An error rate of 10^{-15} should be achieved without factoring in the effect of the interleave, write retry, and write bias. C3 error correction is disabled during read back check when writing in order to bias the write process. If C2 is unable to correct the error of any one byte, a retry is invoked. Table 1 summarizes the error management system.

Table 1: Summary of the Error Management System

Format Item	Format Description
Bytes Per Track	48,972
User Bytes Per Track	37,495
C1 Dimensions	RS(228,220,8) T=4
C2 Dimensions	RS(106,96,10) T=5
C3 Dimensions	RS(96,86,10) T=5
Channel Code	Miller-Squared (rate 1/2)
C1-C2 Product Code Array	In-track block interleaver with dimensions 456 x 106 (two C1 words by one C2 word)
C3 Code Cross-Track Interleave Description	C3 codewords interleaved across a 32-track physical block
Outer CRC Error Detection of C1-C2-C3 Failure	Four 64 parity bit CRC codewords interlaced over 32 tracks which provide undetected error probability of 10^{-20}
Write Retry	Yes
Coding Overhead	28%
Erasurage Flagging Capability of Channel Code	Excellent: probability of flagging a burst error is near 1.0
Maximum Cross-Track Burst Correction	3.3 tracks (139,520 bytes)
Maximum Length of Tape Defect Affecting 4 Adjacent Tracks that is Correctable	35,112 bytes
Maximum Raw Byte Error Rate Which Maintains Corrected Error Rate $< 10^{-13}$	0.021
Maximum Width of Longitudinal Scratch that is Correctable	4,560 bytes

Drive Configuration

The Drive configuration allows for physical separation of the electronics from the transport module at distances up to 100 feet if desired. This allows users to maximize the transport density in robotic environments, and to place the electronics modules in close proximity to host computers.

Media Usage Life

One of the major applications for ER90 technology is a backstore for a Disk/Tape hierarchy. As such, the number of tape load and unload cycles, thread/unthread cycles and searches may be significant. The expected usage capabilities for the ER90 media should exceed 50,000 load/unload cycles, 50,000 tape thread/unthread cycles per system zone, and 5,000 end-to-end shuttle forward and rewind cycles. The number of end-to-end reads using incremental motion (less than 15 MB/sec) should exceed 2,000 while the number of reads of 1 gigabyte files using incremental motion should exceed 5,000. The operating environment should be maintained between 12 to 20 degrees centigrade with relative humidity between 30 and 70% to achieve best results.

Archival Stability

Assuming cassettes are stored within temperature ranges of 16 to 32 degrees centigrade with relative humidity between 20 and 80% non-condensing, storage of over 10 years is expected. For even longer archival stability, an environment maintained between 18.3 and 26.1 degrees centigrade with relative humidity between 20 and 60% non-condensing should result in archival stability exceeding 14 years.

Recent testing by Battelle Institute on D-2 metal particle tapes from four vendors revealed no detectable change after 28 days of exposure to accelerated testing in a mixed gas environment, equivalent to 14 years of typical storage in a computer facility. The following results were determined:

- No evidence was found of localized surface imperfections.
- Improved surface formulations provided a protective coating for the metal particles.
- The D-2 cassette housing protected the tape against damage by absorbing (gettering) corrosive gases.
- Change of magnetic remanence does not differ significantly when compared to other tape formulations in use today.¹

High Speed Search

ER90 data formats include full function use of the longitudinal tracks that can be read in either the forward or reverse direction. One of these tracks contains the geometric address of each physical block of data. This track can be searched at speeds of greater than 300 inches per second, equivalent to searching user data at more than 800 megabytes per second. Another longitudinal track is automatically recorded on tape which provides either addressability to the user data block or to a byte offset within a user file. No user action is required to cause these tracks to be written and they provide high speed search to any point in the recorded data, not just to points explicitly recorded at the time of data creation.

1. Fraser Morrison and John Cororan, "Accelerated Life testing of Metal Particle Tape", *SMPTE Journal*, January 1994

Multiple Unload Positions

Access times are enhanced through the implementation of multiple system zone areas where the media may be loaded and unloaded. Full rewind is therefore unnecessary. This reduces positioning time when loading and unloading the cassette, while eliminating mechanical actions of threading and unloading over recorded data, as well as eliminating the wear that is always inherent in any design that requires a return to beginning of tape.

Robotics

Full robotics support is provided for the ER90 drives by the EMASS DataTower[®] and DataLibrary[™], archives with storage capacities up to 10 petabytes. Both robotics are supported by EMASS VolServ[™] volume management software which is interfaced to tape daemon in the UNICOS operating system.

The DataTower archive uses proven robotics to transfer D-2 small cassettes between 227 storage bins and up to four ER90 drives. This yields a capacity of almost six terabytes of user data in a footprint of only 27 square feet. A bar code reader on the robot scans a bar code on each cassette for positive identification and manageability. Under VolServ software control, the robot inside the DataTower archive rotates on a vertical pole, grabs the designated D-2 cassette from its bin, moves it to an available drive where it is automatically loaded into an ER90 drive. This load operation completes in less than a minute. When the application completes its use of the D-2 cassette, VolServ software will instruct the robot to return the cassette to a storage bin.

For larger storage needs, the DataLibrary archive offers a modular solution that can be expanded to contain petabytes of user data. The DataLibrary archive stores D-2 small and medium cassettes in shelves that make up the cassette cabinets. Each four foot wide cassette cabinet can hold up to 14.4 terabytes of user data. Up to 20 cabinets can be added together to form a row of storage. Rows are placed parallel to each other to form aisles in which the robot travels to access the cassettes. A added benefit of this architecture is that cassettes on interior rows are accessed by two robots. ER90 drives are placed on the ends of the rows to complete the DataLibrary archive. As demand for robotic-accessed storage grows, a DataLibrary archive can be expanded by adding more storage cabinets, more robots, or more ER90 drives. As with the DataTower archive, VolServ software provides complete mount and dismount services through the UNICOS tape daemon.

Conclusions

The ER90 tape drive, by borrowing innovative techniques used in high-resolution video recording, provides the computer processing industry with a helical scan tape format that delivers data from a high density 19 mm metal particle tape. With a sustained rate of 15 MB/sec, input/output intensive applications now have a device that complements the processing speeds of Cray supercomputers. The implementation of system zones allows safe load and unload at other than BOT, providing improved access times. The dense DD-2 format means that the cost of storage media is dramatically reduced to less than \$2 per megabyte.

EMASS/CRAY EXPERIENCES AND PERFORMANCE ISSUES

Anton L. Ognio

Exxon Upstream Technical Computing Company

Introduction

At the Exxon Upstream Technical Computing Company (EUTeC) we have recently purchased an Emass Data Tower from E-Systems. The Data Tower is capable of holding 228 D2 tapes, with each tape having a capacity of 25 Gigabytes for a grand total of approximately 6 Terabytes of storage. The Tower also provides a robotics system to mount the tapes, a cabinet with room for four ER90 recorders, and a SparcStation System (VolServ) for managing media, drive and other component statuses. IPI Channels connect our Tower directly to the Cray, and each recorder is capable of sustaining 15 MB/s. We intend to harness the storage capacity of the Data Tower to provide a centralized repository for managing our large tape library. In the future, we may expand its usage to our IBM MVS machines and our UNIX workstations.

We faced two challenges in making the ER90 drives available for production use. The first challenge was to make the ER90 drives available to our users. To do so, we decided to extend the functionality of Data Migration (DMF) to include multiple Media Specific Processes (MSPs) solely for D2 media. This approach gave us a simple mechanism for storage and retrieval of data, as well as archival and protection services for our data files. DMF allows us to support ER90 use with minimal programming effort, but at the expense of some flexibility and some performance. The hurdles involved in making ER90s available through DMF include configuration and operational issues with VolServ and the Data Tower network, configuration of the tape daemon, formatting of the tape media, configuration of DMF,

and some modifications to DMF to work with multiple MSPs.

Performance has been our second major challenge. Each drive in the tower is capable of sustaining 15 MB/s from Cray memory to tape on a dedicated system. At the moment, Data Migration is giving us at most 7 MB/s per drive or about 20 MB/s aggregate. By improving the I/O algorithm used by DMF in a standalone program, we have been able to achieve a rate of 13.5 MB/s from a single drive to disk and a rate of 26 MB/s with two drives reading or writing concurrently. With such a significant performance gain, we concluded that the performance loss seen in Data Migration was due to the I/O algorithm of the DMF code. An explanation of these findings follows later.

DMF/ER90 Experiences

Hardware Hookup and Configuration of a Standalone ER90

Initially, EUTeC opted to have one ER90 drive on site until a Data Tower became available for installation. Before we could make the ER90 available to the Cray, we had to rebuild and reconfigure the tape daemon with ER90 software support, which Cray distributed as an asynchronous product. These drivers installed without incident; however, there were changes to the Tape Configuration File that did not install so easily. **ER90 Tape Daemon Support Technical Note (SG-2137)** gave us a rough outline of configuring a tape

loader for the ER90. Unfortunately, the documentation was insufficient to steer us around issues such as appropriate time-out values, maximum block size configuration and operator confirmation of "blp" tape mounts¹. As the technology matures, we expect that this document will mature commensurately. (All sites)

Installing and Configuring DMF

After we had established communication between the tape daemon and the ER90, we focused on configuring DMF to use the D2 MSPs. Because we needed to make the ER90s available quickly, we realized that only two packages could satisfy our data integrity requirements - Data Migration or Cray Reel Librarian (CRL). Unfortunately, CRL had to be abandoned because it cannot coexist with IBM Front End Servicing (a site requirement). Thus, we decided to allow access to the ER90s through DMF only. We had been running DMF with 3490 tapes for a year, which significantly lowered the learning curve for EUTeC and our clients. Before the first ER90 arrived, we were running DMF 2.04. To gain ER90 support, we upgraded to DMF 2.05, and with the installation of Unicos 8.0, we upgraded to DMF 2.10, all of which caused us no problems.

The following paragraphs outline the customizations we have made or expect to make to DMF:

Multiple MSP Selection

To allow our users to migrate to the D2 MSP while continuing to use the 3490 MSP, we made local mods to the dmmfunc routine, which Cray provides for just such a purpose.² At EUTeC, if a user sets the sitebit in the inode to a number between 1000 and 1010, our modified dmmfunc returns an appropriate MSP index from 0 to 10. This approach has several disadvantages, including a lack of access control to the ER90s. We have submitted a design SPR with

¹When formatting tapes and labelling tapes, the tpformat and tplabel commands use bypass label processing (see below)

²Dmmfunc receives a stat structure, and returns an MSP index for the specified file.

CRI to add a UDB field for controlling access to each MSP. (DMF sites with more than one MSP)

No tape unload

To leave cartridges mounted after DMF performs an operation, we have also added a "no unload" option to the ER90 tape mounts for DMF. If a program accesses files from the same cartridge sequentially, then this measure greatly decreases the number of tape mounts required. (DMF sites)

Hard Deletes and Default Copies.

Other issues concerning Multiple MSPs include the lack of hard delete and default copy parameters on a per MSP basis or per file system basis. We have submitted design SPRs for both of these parameters. (DMF sites with more than one MSP)

Read access equals dmput access.

Another problem that we ran into as users shared these D2 files, was that, while a user could retrieve files he did not own with dmget, he could not release their space with dmput. At EUTeC, we modified dmput to allow users with read or write access to dmput those files. (DMF sites)

Configuring the Tape Daemon and VolServ for the Data Tower

When the tower arrived, with the two additional ER90s, we had to set up the VolServ Sun to communicate with both the Data Tower and the Cray. Also, with the addition of the Data Tower, we added another set of mods to the tape daemon, and configured another tape configuration file. In this file, we configured the ER90 loader to use VolServ as the Front End Service. We have made only minor changes to VolServ, increasing the time-out values for RPC messages, and sending VolServ log messages to an alternate display. (All sites)

Formatting and Labelling tapes

Before the tape daemon uses a D2 tape, the tape must be formatted with tpformat. This process encodes

cartridge identification onto the tape, and divides each tape into logical partitions, which the tape daemon treats as separate volumes. For performance reasons, these logical partitions should be sized appropriately to the user's file sizes³.

You must run `tpformat` on every cartridge in the tower. Initializing all 228 tapes in a Data Tower is a time consuming process and could take several days utilizing all three recorders simultaneously. If you require labels on each partition, then the time for initialization doubles. Tape labelling is optional, and, if used, must be done for every partition on every tape. Labelling will also slow each tape mount by about 30 seconds, and for that reason is not recommended by CRI. We have written a script to perform this onerous task, but have run into problems because `tpformat` and `tplabel` perform their own `rsync` command, thus forcing the script to be single threaded. Also, our users' average file sizes may change over time, which may require us to reformat the remaining free tapes to achieve peak performance. One site has written a C program to format and label their entire archive by formatting on several drives at one time. I recommend that approach heartily. (All sites)

Emass Training and Establishing Operational Procedures

When installing a Data Tower, consider the training required for its support and administration. Several members of our staff have attended the VolServ Class offered by E-Systems. This class gives an overview of VolServ only. It does not cover configuring a Data Tower attached to a Cray, or Data Migration with ER90s, performance, or tape daemon configuration. Most of what we have learned in those areas has been through first hand experience, talking to Cray personnel and hashing out problems with E-Systems onsite support. (All sites)

Along with the burden of learning the ins and outs of the system ourselves, we have spent considerable effort training our operators to handle emergency situations, cycling the drives and the VolServ software, and monitoring skills. (All sites)

³An appendix to the DMF 2.05 Administrator's Guide helps you choose the optimum partition sizes.

Hardware Errors

Since the installation of the Data Tower, we have had a handful of hardware outages. Due to two recent computer room blackouts, we have lost two power supplies. We have also had a card in one drive controller go bad and some failures with unmounting tapes. Onsite support has been available to fix these problems, and E-Systems is working to improve its support at Cray installations. (All sites)

Media Recognition Problem

We have seen cases where tapes, that we had formatted and labeled, become unmountable. In some of these cases, when DMF mounts a tape, the tape daemon returns a recoverable error and attempts another mount, in other cases, the drive goes down and our hardware technician must manually remove the tape from the drive. Worse yet, we have seen a case where a formatted tape that DMF has written data to is sporadically unmountable. In this case, DMF has written 8 out of 11 partitions on one cartridge, and tape mounts of that cartridge still fail with the message "Unable to position to partition X." This problem was fixed in an upgrade of VolServ. (All sites)

Transition to FileServ

There are many potential advantages to switching to FileServ. To start, FileServ allows users to write directly to D2 tape. Second, FileServ allows variable partition sizes, which will optimize tape usage (see Formatting and Labeling Tapes). Third, FileServ backs up the data inodes to tape when migrating files. Lastly, FileServ will use Asynchronous I/O (See DMF/ER90 Performance). Overall, these enhancements will be an improvement over DMF in performance, functionality and recoverability. FileServ is due to be released in 3Q94, and we will consider it as an alternative to DMF. (Sites considering FileServ)

DMF/ER90 Performance

With a high performance tape subsystem and high performance disks come high expectations for I/O performance. Unfortunately, accessing the ER90s

through DMF did not give us the 15 MB/s transfer rate that we expected. By simulating DMF's I/O methodology in a standalone program, we were able to benchmark transfer rates for DMF and then improve upon the algorithm. Ultimately, we found that the ER90 caching features, combined with DMF's synchronous I/O, caused a transfer rate of between 4 to 8 MB/s. We also found that a program using buffered, asynchronous I/O could produce nearly 14 MB/s on a loaded system. Because of this study, we believe that our I/O methods could generically improve DMF/ER90 performance by 100%. To test our theory about buffering, we wrote several C programs using Unbuffered Raw I/O, Flexible File I/O (FFIO), and Asynchronous, Buffered I/O.

Unbuffered, Raw I/O

First, we wrote a program to simulate DMF's dmtpput and dmtpgt operations. This program opened a disk file with the O_RAW attribute to bypass system buffers⁴. As with DMF, this program looped - reading 4 MB at a time from a striped DD60 disk file and writing that block of data to D2 tape (or vice versa for tape reads). As with DMF, reads and writes occurred synchronously, and the program achieved a peak performance in the 7 MB/s range, significantly lower than the maximum speed of the ER90s and the DD60 drives.

Running this program we noticed a distinct pattern in our read and write times to and from tape.⁵ When writing to tape, the first write generally takes about 2,800,000,000 clocks, which is two orders of magnitude larger than most other writes. Since the first write includes drive ramp up, tape positioning, and label processing time, we rewrote the program to throw away this the time (see Appendix). After the first write, the program wrote the next 4 MB chunks in about 40,000,000 clocks each, until about the 13th write, which was generally about 400,000,000

clocks. After that, writes to tape followed a pattern of 10-14 writes at 40,000,000 clocks, and one write at 400,000,000 clocks.

Here is a sample of the timing:

```
readtime 25000000 clocks, writetime 2800000000 clocks
readtime 25000000 clocks, writetime 40000000 clocks
(Last result repeated 12-13 times)
readtime 25000000 clocks, writetime 4000000000 clocks
readtime 25000000 clocks, writetime 40000000 clocks
(Last result repeated 12-13 times)
readtime 25000000 clocks, writetime 4000000000 clocks
```

...

After some digging, we found that there were two causes for this pattern, both associated with the ER90 caching mechanism. Each ER90 drive buffer inputs into a 64 MB cache before sending it to tape. When the buffer fills to 45 MB, the drive ramps up and starts writing to tape. Because the buffer is initially empty, the Cray must write about 11 x 4 MB blocks before the transport starts. Ramp up for the drive and the transfer from buffer to tape then accounts for the slow write after the first series of writes. After that, we found that the ER90 actually reads its cache faster than the Cray disk could write. This would cause the drive buffer to empty, which caused the transport to stop until the buffer filled up and the transport started again. The combined effects of synchronous I/O, drive buffering and drive ramp up caused the entire transfer to be sluggish.

⁴Note that we did not use the O_SYNC option that DMF uses.

⁵Between each read and write we called rtime() to give us an approximate time for reads verses time for writes. We used time() to time the entire transfer from start to finish.

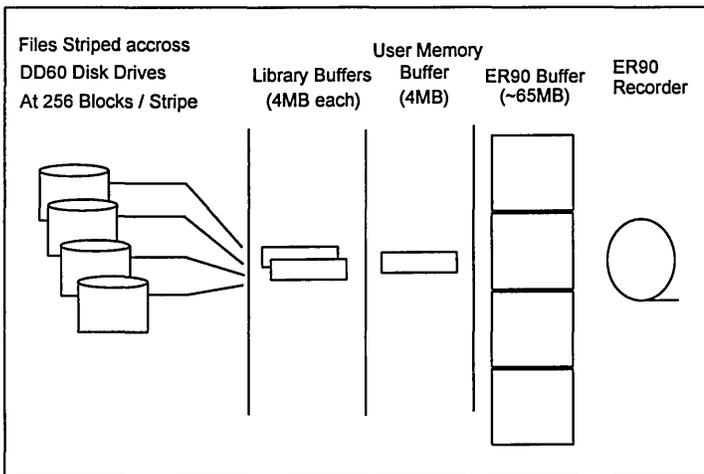
FFIO Program

Based on the results of the first program, we decided to rewrite the program to allow us the flexibility to experiment with various I/O methods without recoding. The tool of choice was Flexible File I/O (FFIO), which allowed us to change the I/O method of the program by assigning attributes to a file with the "asgcmd" command. A sample "asgcmd" command, assigning 2 x 4 MB library buffers would be:

```
asgcmd -F bufa:1024:2 diskfile
```

For our tests, we used 1024 block⁶ buffers to correspond with the 4 MB tape block size.⁷

The following diagram illustrates the data path from disk to tape with FFIO library buffering. Notice that 12 MB of mainframe memory is required to hold the library and user buffers and that the ER90 cache is several times larger than the tape block size:



$${}^6 1024 \text{ blocks} * \frac{512 \text{ words}}{\text{block}} * \frac{8 \text{ bytes}}{\text{word}} = 4,194,304$$

Bytes = 4 MB.

Results from the FFIO code showed a dramatic overall transfer rate increase when we used FFIO buffering with the disk files. The pattern of reads and writes between disk and tape also changed dramatically. When writing to tape, the first write still takes about two orders of magnitude longer than most other writes, and the first 13 writes behave the same as in the unbuffered I/O case. But after the drive ramps up once, reads and writes occur consistently in under 40,000,000 clocks, until the end of the transfer.

Because the disk I/O is sufficiently fast to fill or empty the drive cache, the drive transport never stops, which nearly doubles the transfer rate to 13.6 MB/s.

Here is a sample of the timing (note that the read times from disk are significantly faster than the write times to tape.):

```
readtime 800000 clocks, writetime 2800000000 clocks
readtime 800000 clocks, writetime 400000000 clocks
(Last result repeated 12-13 times)
readtime 800000 clocks, writetime 400000000 clocks
readtime 800000 clocks, writetime 400000000 clocks
...
(Last result repeated until end of transfer)
```

See Appendix A for a listing of the FFIO Code

⁷Varying buffer sizes did not produce significant performance changes and varying the number of buffers produced marginal changes

Asynchronous I/O Program

Finally, once we found the optimal buffering scheme via our FFIO program, we wrote an asynchronous I/O program with the same algorithm to benchmark against the FFIO program. We wanted to eliminate any overhead in CPU time, memory usage, and memory to memory copies that the FFIO buffering layer may have added.

In the Asynchronous I/O code, we hard coded the double buffering algorithm. Using the `reada` and `recall` system calls, we used a double buffering scheme that most closely approximated the FFIO example with 2 x 4 MB buffers (`asgcmd -F bufa:1024:2 diskfile`).

The results from transferring 2 GB of data were similar to the FFIO program at 13.6 MB/s, but we managed to eliminate one 4 MB buffer and most of our user CPU time. The elimination of an additional memory to memory copy between the FFIO library space and user space easily accounts for this decrease in CPU overhead.

See Appendix B for a listing of the Asynchronous I/O code.

Recommendations

Because ER90 tape performance is so dependent upon disk performance, I believe that Cray should consider writing separate I/O routines for ER90 tapes using the fastest and cheapest I/O method available. Because of the enhanced speed, low memory overhead and low CPU overhead of the Asynchronous I/O cases, the results of our testing at EUTeC clearly support Asynchronous I/O as the preferred I/O method. I feel that, although recoding may not produce 13.6 MB/s consistently from DMF, it would speed file transfers to consistently over 10 MB/s from DD60s.

Further tests that I would recommend are running transfers from unstriped files residing on DD60s, and running transfers from DD40 series disks. Other experiments could include concurrent ER90 testing and a bottleneck analysis of the data flow.

Acknowledgements

I would like to thank the many people who contributed to the success of this investigation. I am especially grateful to Bryan White from Cray Research, Kent Johnson and Barney Norton from E-Systems, and Doug Spragg, Troy Brown and John Cavanaugh from EUTeC for their insight and guidance in conducting the transfer tests and in critiquing this paper. Thank you.

Appendix A - FFIIO Program Partial Listing

```
#define AMEG      (long) 1024*1024
#define BSZ      (long) BLOCKS * 16384      /* 2097152 bytes */

/* do first write without timer on */
RET1=read(tapefd, buf, BSZ);
RET2=ffwrite(diskfd, buf, BSZ, &diskstb, FULL);
tfwrite=time((long *) 0);

do {
    bef_read=rtclock();
    RET1=read(tapefd, buf, BSZ);
    aft_read=rtclock();
    RET2=ffwrite(diskfd, buf, BSZ, &diskstb, FULL);
    aft_write=rtclock();
    bytes_w+=(long) RET2;
    printf("readtime %d clocks, writetime %d clocks\n",
           aft_read-bef_read,  aft_write-aft_read);
} while ( RET1 == BSZ && RET2 == BSZ );

tfinish=time((long *) 0);

speed=( (float)bytes_w / ((float)AMEG*((float)tfinish-(float)tfwrite) ) ) ;
printf("Wrote %d bytes in %d seconds at %7.3f MB/s\n",
       bytes_w, tfinish-tfwrite, speed);
```

Appendix B - Asynchronous I/O Program Partial Listing

```
#define BUFF_SIZE (4 * 1024 * 1024 )

/*
 * Priming Read
 */
statlist[0] = &rsw[0];
rc = read(in_fd, buffer[curr_buffer],BUFF_SIZE);
if (rc < BUFF_SIZE) {
    finished = 1;
    write_count = rc;
}

while (!finished)
{
    rc = reada(in_fd, buffer[(curr_buffer+1) % 2 ],BUFF_SIZE, &rsw[0], 0);

    rc = write(out_fd, buffer[curr_buffer],BUFF_SIZE);

    rc = recall(in_fd, 1, statlist);

    if (rsw[0].sw_count < BUFF_SIZE)
    {
        write_count = rsw[0].sw_count;
        finished = 1;
    }

    bread += rsw[0].sw_count;
    curr_buffer = (curr_buffer+1) % 2;

    /* Start timer after first write */
    if (loops == 1 ) tfwrite=time((long *) 0);

    loops++;
}

rc = write(out_fd, buffer[curr_buffer],write_count);

tfinish=time((long *) 0);

printf("Wrote %d bytes in %d seconds at %10.3f MB/s\n",
        bread, tfinish-tfwrite,
        (float)bread/(1024*1024*(tfinish-tfwrite)) );
```

Appendix C - Write Results

For easier comparison, we always used two Gigabyte files, user striped across four disks for transfers.

Since these tests were run on a loaded system, the results were slightly lower than the 15MB/s rate we had achieved from memory to tape on an idle system.

Read from Disk or Memory / Write to Tape						
I/O methodology	Tape Block size	Buffer Size	Number of Buffers	FFIO?	CPU seconds to transfer ~2GB	MAX Transfer Rate
Like DMF, open(disk,O_RAW), synchronous read/write	4MB	n/a	1 User	No	2.48 sys/ 0.05 usr	7.117 MB/s
Like DMF, but using FFIO. ffdopen(disk,O_RAW), asgcmd -F system disk, synchronous read/write	4MB	n/a	1 User	Yes	1.12 sys/ 0.06 usr	7.299 MB/s
Double buffered using FFIO ffdopen(disk,O_RAW), asgcmd -F bufa:1024:2 disk	4MB	1024 Blocks (4MB)	2 Library + 1 User	Yes	1.04sys/ 1.79user	13.699 MB/s
Double buffered using FFIO ffdopen(disk,O_RAW), asgcmd -F bufa:1024:4 disk	4MB	1024 Blocks (4MB)	4 Library + 1 User	Yes	1.07sys/ 1.79user	13.605 MB/s
Double buffered using Async. I/O opena(disk, O_RAW)	4MB	1024 Blocks (4MB)	2 User	No	0.85sys/ 0.01usr	13.644 MB/s
Memory to Tape, open(tape,O_RAW)	4MB	n/a	1 User	Yes	0.18sys/ 0.02usr	13.699 MB/s

Appendix D - Read Results

For easier comparison, we always used two Gigabyte files, user striped across four disks for transfers. Since these tests were run on a loaded system, the results were slightly lower than the 15MB/s rate we had achieved from tape to memory on an idle system.

Read from tape / Write to Disk or Memory							
I/O methodology	Tape Block size	Buffer Size	Number of Buffers	FFIO?	CPU seconds to transfer ~2GB	MAX Rate	Transfer
Like DMF, open(disk,O_RAW), synchronous read/write	4MB	n/a	1 User	No	1.61sys/ 0.05usr	6.770 MB/s	
Like DMF, but using FFIO. ffoopen(disk, O_RAW), asgcmd -F system disk, synchronous read/write	4MB	n/a	1 User	Yes	2.58sys/ 0.06usr	6.549 MB/s	
Double buffered using FFIO ffoopen(disk, O_RAW), asgcmd -F bufa:1024:2 disk	4MB	1024 Blocks (4MB)	2 Library + 1 User	Yes	1.03sys/ 1.86user	13.184 MB/s	
Double buffered using FFIO ffoopen(disk, O_RAW), asgcmd -F bufa:1024:4 disk	4MB	1024 Blocks (4MB)	4 Library + 1 User	Yes	1.17sys/ 1.85user	13.098 MB/s	
Double buffered using Async I/O opena(disk, O_RAW)	4MB	1024 Blocks (4MB)	2 User	No	1.05sys/ 0.01usr	13.132 MB/s	
Tape to memory, open(tape,O_RAW)	4MB	n/a	1 User	Yes	0.15sys/ 0.02usr	13.158 MB/s	

AFS EXPERIENCE AT THE PITTSBURGH SUPERCOMPUTING CENTER

Bill Zumach

Pittsburgh Supercomputing Center
Pittsburgh, Pennsylvania

Introduction

The Pittsburgh Supercomputing Center is one of four supercomputing centers funded by the National Science Foundation. We serve users across the country on projects ranging from gene sequencing to modeling to graphics simulation. Their data processing is done on either our C90, T3D, CM 2 Connection Machine, MASPAR, or our DEC-Alpha workstation farm. We have an EL/YMP for a file server using Cray DMF.

To support the data needs of our external users as well as our support staff, we use AFS, a distributed file system, to provide uniform, Kerberos secure access to data and for ease of management. To store the high volume of data, PSC designed a hierarchical mass storage system to provide access to DMF as well as other mass storage systems through AFS. We refer to this as multi-resident AFS.

This paper discusses this mass storage solution and how we use multi-resident AFS to provide access to data for our users. This presentation is done in the form of a chronology of our need for an ever larger and more flexible mass storage system. Also described are the other sites using our ports of the AFS client and multi-resident AFS. Lastly, a brief description of future work with both AFS and DFS.

AFS at PSC

PSC uses the Andrew File System to serve the binaries for upwards of 120 workstations and the home directories of about 100 staff. AFS provides a location transparent global name space. This gives users the same view of the directory structure no matter where they log in. It also makes workstation administration much easier as the programs need only be located in a single location. AFS also scales well, allowing for the huge amount of data that needs to be managed at the PSC.

We chose AFS over the defacto standard NFS for several reasons. First, NFS does not scale well. Most of the system binaries, all home directories and many project areas, totaling about 40 gigabytes, are currently stored in AFS. NFS has two main difficulties dealing with this amount of data spread across this many workstations. First, an NFS server becomes overloaded with requests with a sufficiently large number of clients. Second, administering the file name space quickly becomes cumbersome if many partitions are being exported.

An NFS client needs to contact the server for each read or write of a file. This quickly bogs down an NFS server. AFS on the other hand caches the pieces of a file which are in use on the client. The server grants a callback promise to the client, guaranteeing the data is good. This guarantee holds until some other client writes to the file. Thus, the AFS client only needs to talk to the server for a significant change of state in a file. At the same time, repeated reads and writes to a file by a single client occur at near local disk speeds.

With NFS, each client can mount an NFS partition anywhere in the directory structure. At PSC, most of the system binaries for all workstation architectures are in the distributed file system. This makes updating operating system software and data processing packages extremely easy. But, for NFS this makes it incumbent upon system administrators to be extremely careful in setting up each NFS client. When the number of workstations gets sufficiently high, this task becomes cumbersome and subject to error. AFS has a location independent, uniform global name space. So wherever a user logs in from, they see the same directory structure. An AFS client finds a file by looking in an AFS maintained database for the server offering the file and then goes to that server for the file. This all happens as part of the file name lookup and nothing is explicitly mounted on the client.

One other significant feature of AFS is security. This comes in two forms. First a user is authenticated using Kerberos security and file transfers from server to client depend on that authentication. This is a major improvement over NFS. Secondly, AFS supports the notion of access control lists. These lists apply to directories and give explicit permissions based on the Kerberos authentication.

AFS also has the concept of a volume which is a logically related set of files. Volumes are mounted in a manner similar to disk partitions, that is, at directories in the AFS name space. Volumes are typically used to house users' home directories, sets of binaries such as /usr/local/bin and for space for projects. They can have quotas attached to them for managing disk usage and since they are mounted on directories, access control lists apply to volumes as well. There can be several volumes per disk partition, so they provide a finer control of disk quota allocation. As quotas can be dynamically changed, disk usage can be modified as well. Volumes can also be moved from partition to partition and across servers, making data distribution manageable. Dumps are done on a per volume basis, giving more control over what gets backed up and when.

Volumes can also have backup volumes, which are read only snapshots of the volume at a given time. One use of this at the PSC is to maintain an OldFiles directory in each users home directory, which contains a copy of the home directory as it appeared the previous day. This makes it extremely easy for the user to get back that file they decided they should not have deleted yesterday.

Volumes can also be cloned. These are also read only snapshots of a volume and can reside on a different file server than the read write original. This is useful for distributing the read load for data across several machines. An AFS client randomly chooses which read only clone to access if one is available when reading a file. This is typically used for operating system binaries.

One last major concept in AFS is that of a cell. An AFS cell is an administrative domain. A user authenticates to an AFS cell to

access the files in that cell. A cell is delimited by a given set of file servers. This allows individual sites to maintain their own set of secure file servers and to restrict access to a selected set of users. At the PSC, we currently have two cells. One is a stock production cell and the other is the cell which implements our solution to the mass storage problem. We are currently looking into merging these two cells.

AFS is produced by Transarc, and is supported on a wide variety of platforms. Among the manufacturers supported are IBM, Sun, DEC, HP and SGI. In addition, Convex and CDC provide AFS for their machines. This list is by no means exhaustive. Through the work at PSC, the AFS client has been available for some time for Cray C90s and YMPs. We have recently ported the multi-resident AFS file server as well.

From its inception, the PSC used AFS to store binaries and users' home directories. AFS also served as the home directory and project space for the Sun front ends to the CM-2 Connection Machine. As stated earlier, this provided us with a uniform view of the file system. Since almost all of our users are off site, they could create their data on any AFS based client, and immediately work with it at the center without having to explicitly move any data to a particular machines local file system. Thus, they could examine their data on any of the front ends or their own workstation, process it on the CM-2 and view the results on their own system.

This provided the impetus to port the AFS 3.1 client to the Cray, a YMP at the time. This not only tied our main processing computer into the distributed file system, but allowed for user's to easily split their processing tasks based on which machine, CM-2 or YMP which was best suited to the task without having to move their data.

As is usual, data storage demands began to outstrip our capacity. The PSC decided to acquire a RAID disk system for fast, reliable access to data. We settled on a RAID-3 system from Maximum Strategy running on a Sun. The problem was that AFS does not support anything other than native local file systems and the RAID disks only had a user mode file system. To support this file system, the file I/O sub-system of the AFS file server was generalized to be a generic I/O system. We then had an AFS file server running on a Sun which was able to use the RAID disk bank as it's local file system.

We soon found that as the RAID disks stored data in 8 kilobytes blocks, the RAID disks were too inefficient at storing many of the files generated by a typical user. To this end we wanted to split the data up between SCSI disks, which are faster for writing and reading small files, and the RAID disks, which are more efficient at storing large files. Standard AFS only determines where a file should be stored based on where the volume is. That is, a volume's files are stored on the same partition as where the volume was created. The idea was to separate the idea of a file from the storage medium. This gives rise to the concept of a file's residency. That is, a file is in a volume, but we also store information as to where the file's data resides. In this case, files in a single volume could reside on either the RAID disk or the SCSI disk. The location information is stored in the meta-data in the volume. To determine where a file should reside, one needs a stored set of characteristics for the storage device. We call this a residency database. There is one entry in the database for each storage device. Each entry

contains, among other things, the desired minimum and maximum file size for that storage device. So, when the modified AFS file server wants to store a file the residency database is consulted to determine which storage device wants files of that particular size.

As demands for storage continued to grow, we realized that we needed some type of mass storage system. For some time, we had been using Los Alamos' Common File System (CFS) for our needs. This provided a simple get and put interface for user's files, but is also rather cumbersome for users as one needs to explicitly obtain one's files prior to using them in a program.

In order to tie CFS into AFS, several new concepts were required. We needed to get files into and out of CFS from an AFS file server. We needed to be able to move files automatically into CFS so that they got archived. And we needed to be able to free up disk space.

CFS did not run on any of our file server machines. We did not want to port the AFS file server to Unicos and we did not want to impact the performance of the YMP by making it a file server. The solution was to have a small server running on the YMP which handled file requests from an AFS file server. We refer to this small server as a remote I/O server. So, AFS clients would request a file from the file server. The volume meta data indicates the file is remote and the file server sends an RPC to the remote I/O server to deliver the file to the file server, which in turn delivers the file to the AFS client. In this case, a file in CFS would be spooled onto the Cray by the remote I/O server which would hand it back to the AFS file server who sends the data to the client. This made CFS access transparent to the user.

In order to migrate files from the disks local to the AFS server we needed a mechanism which would do this automatically and on a timely basis. This data migration process is accomplished by a daemon running on the AFS file server machine which periodically scans the disk, looking for older files which are candidates for migration to slower storage devices. In our case this meant scanning the SCSI and RAID disks, looking for files to migrate to CFS. This also meant that the residency data base entries needed an entry which indicated how old a file should be before being migrated to that storage system. We decided that if a file had not been accessed in 6 hours, it was a candidate for migration to mass storage. When the scanning daemon finds files which are 6 hours old it informs the AFS file server on the same machine and the file server is in charge of moving that file to CFS using the remote I/O server.

Now just because a file is old, that alone does not mean it should be deleted from faster storage media. So we leave the original copy of the file on the faster media. This means there are now two copies of the file. One on either SCSI or RAID disks and the other copy in CFS. This is called multiple residency and gave rise to the name multi-resident AFS. A residency is defined to be a storage device along with the list of file servers which can access that storage device on behalf of an AFS client. We store a list of all residencies for a given file in the volume's meta-data so the file server knows where it can go to find the file an AFS client requests. Note that one does not want to go to CFS if the file is on local disk. This gives rise to the concept of a priority for a residency, and each residency entry in the database contains a priority. While priorities can be arbitrary, we set priorities based on speed of file access. This means that if a file is both on a disk local to the AFS file server

as well as in CFS, the local disk copy would be obtained for a client, since it's at a higher priority.

Since we don't automatically delete a file from a given residency once it has been moved to tape, it is quite likely that the local disks would soon fill up. To avoid this problem, each fileserver machine has a scanning daemon running on it which ensures that older, migrated files are removed from the disk, once a free space threshold is reached. The removal algorithm is based on file age and file size.

We shortly encountered a major problem using CFS for mass storage. While CFS works well for storing files, the transaction overhead on each file update is quite high, on the order of 3 seconds. This causes problems with backup volumes in AFS. When a backup is made, the volume is first cloned and the volume is off line until the clone is complete in order to ensure a consistent image of the volume. Also, in multi-resident AFS, if a file is not on the local disk, it is not explicitly copied. Its reference count is incremented instead. This means that if the file is in CFS, CFS takes 3 seconds to increment the file's reference count. So cloning a volume with 1200 files in CFS would mean that the volume would be offline for an hour.

It was not possible to fix this transaction time overhead problem in CFS. As a result we investigated other mass storage systems and settled on Cray's Data Migration Facility (DMF). DMF provided us with simple access to the tape media simply by placing the files on a DMF backed partition. As multi-resident AFS is already taking care of migration policy, data landing on this DMF backed partition is already considered to be on slow media, so we explicitly call `dmput` to flush the data to tape and `dmget` to retrieve required files upon demand.

Current Usage and Performance

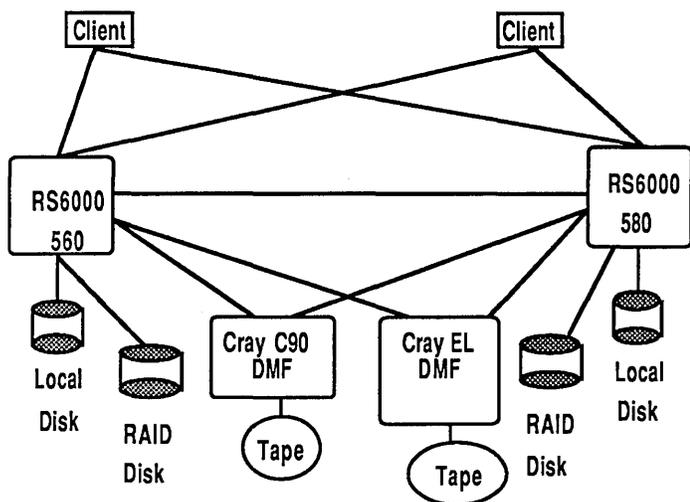


Figure 1. Current multi-resident system

A part of our current multi-resident AFS configuration is presented in figure 1. This figure shows two RS6000s for fast storage. Both have SCSI disks for small files and Maximum Strategy RAID disks for larger files. For archival storage we are currently sending the data to either the C90 or the EL/YMP, both using DMF for tape storage. We are in the process of migrating our all DMF usage from the C90 to the EL. DMF originally only wrote files larger than 4 kilobytes to tape, so we

only archive files larger than this. The small files are backed up using standard AFS backup practices. We have modified the standard AFS dump routines so that only files actually present on the local disk are backed up. Our AFS mass storage system currently contains approximately 383,000 files and about 98.7 gigabytes of data used by 100 users.

AFS Server	Media	Network	Read Speed
DS 3100	SCSI drive	Ethernet	360 KB/sec
DS 5000/200	SCSI drive	FDDI	726 KB/sec
Sun 4/470	IPI drive	Ultranet	20 KB/sec
Sun 4/470	IPI drive	Ethernet	554 KB/sec
Sun 4/470	RAID drives	FDDI	906 KB/sec
Sun 4/470	IPI drive	FDDI	986 KB/sec
IBM RS6000	IPI drive	FDDI	1667 KB/sec
EL/YMP	IPI drive	HIPPI	497 KB/sec

Table 1. Cray AFS client read performance.

Table 1 presents the read performance we see on our C90 AFS client from a variety of servers. Note that, while the EL file server performance is not spectacular, it is reasonable. The performance is somewhat slow due to the fact that we are using the Unix file system as the interface to the disk. This involves a lot of system overhead in opening and reading meta-data files which contain file location and access information. It would be possible to develop an interface for the Cray similar to the one standard AFS uses to obtain an appreciable improvement in file server performance. This would require modifying the `fsck` program in a fairly straightforward manner and adding 5 system call entry points to the kernel.

PSC Ports to Unicos of AFS

What follows is a brief technical discussion of the details of porting the AFS 3.1 client and multi-resident AFS to Unicos. As mentioned earlier, we ported the AFS 3.1 client early on in order to give uniform access to data to users using the YMP. We ported multi-resident AFS for use by the Max Plank Institute in Garching, Germany.

The initial AFS client port was to Unicos 6.0 on the YMP. This port has since been upgraded to Unicos 7.C on our C90. There were several substantive porting issues. First, Unicos until version 8.0, uses a file system switch, whereas AFS is vnode based. This meant a fake vnode system needed to be designed to map AFS vnodes for the cache files and the server files to Unicos NC1 inodes. There are also a large number of problems associated with the 64 bit word size and structure alignment. These problems appear in the packets which get sent across the network, AFS directory layout, and data encryption algorithms. In addition, since a Cray is a physical memory machine, a buffer pool needed to be devised to handle malloc'ing data areas. Lastly, we had to find every place where the kernel could sleep and ensure that either none of the variable in use across the sleep we stack variables, or fix the stack addresses once the kernel came back from the sleep. This is another problem which has been fixed in Unicos 8.0.

We are now in the process of porting the AFS 3.3 client to Unicos 7.C as part of the final plan to port the AFS 3.3 client to Unicos 8.0. The AFS 3.3 client is expected to run much faster than the AFS 3.1 client owing to improvements in the network layer, developed initially here by Jonathan Goldick. Further performance enhancements have also been added by

Transarc for AFS 3.3 and we also are beginning to investigate making further performance enhancements to the client.

Multi-resident AFS was written with Unicos in mind, so combined with the effort that had gone into the port of the AFS client, this port was much easier. Multi-resident AFS is based on AFS 3.2, whereas the Unicos client is based on AFS 3.1. The client and server share several common libraries, including the RPC layer, AFS directories, and data encryption for authentication. Porting these involved bringing the Cray port of AFS 3.1 up to AFS 3.2. There remained a number of 64 bit issues for the file server, including handling internet addresses in the hostent structure as well as a few word alignment issues. In addition, we need to lock a Unix file system when salvaging (AFS version of fsck for volumes). Multi-resident AFS depended on locking a directory using flock which is not possible with Unicos. The AFS vnode structure needed integer data types converted to bitfields and is now twice the size of the vnode for standard AFS. But this helped us optimize the file server as well as allowing volumes to move between Cray AFS file servers and AFS inode based file servers. One additional modification was to port the dump and restore routines to correctly dump access control lists. These were previously dumped as blobs of data. But with the change from 32 to 64 bit word size, we needed to ensure we converted during reads and writes of the dump.

We are currently in the process of porting multi-resident AFS 3.2 to the AFS 3.3 code base. Most of the port is complete and we are now in the process of testing multi-resident AFS 3.3 on several platforms.

Other Sites Using Multi-Resident AFS

The Max Planck Institute, IPP, in Garching, Germany recently purchased a Cray EL to serve as a multi-resident AFS file server and to support DMF for mass storage. This is currently beginning operation and should be a full production environment this summer.

NERSC is currently testing multi-resident AFS at their facility and intends to use the Unix file system interface to connect to Uintree. In addition Transarc is evaluating multi-resident AFS and if they decide to make a product of it, it will be available by the end of 1994. This Transarc product will not provide direct support for Unicos, but will retain the modifications we have made.

Several sites use our port of the AFS client in a production environment. Among them are NCSA in Illinois, SDSC in San Deigo, MP/IPP, LRZ in Munich, ETH in Zurich, RUS in Stuttgart, and EPFL in France. These sites appear to be satisfied with the AFS client.

DFS projects at PSC

As part of our close working relationship with Cray, we did the initial port of the DFS client for Unicos 8.0. During the course of this work we have also assisted in debugging the DFS file server.

We are beginning to think about the design of a multi-resident version of DFS. This will be backwards compatible with multi-resident AFS and will be able to use the same AFS to DFS translator that Transarc is supplying. DFS is still immature and

there are several basic design questions which need to be answered, particularly with regard to DFS filesets before we can devote a lot of time to this project.

Next Generation of Multi-resident AFS

As noted above, network performance is improved dramatically in AFS 3.3 as a result of initial work done here at the PSC with respect to packet size over FDDI. Our initial tests indicate a doubling in file transfer rate with a doubling of the packet size for FDDI. This provided the initial spark to consider adding alternate methods of delivering file data from a storage device to an AFS client. If one had the full bandwidth of HIPPI available and both the residency and the AFS client were on the same HIPPI switch, spectacular improvements in data transmission speeds could be achieved. So the means of asking for the data needs to be separated from the actual delivery of the data. In this scenario, the AFS file server serves as an arbitrator, deciding what is the best network transport (and storage device) to use to get data to the client. This notion is referred to as third party transport, the third party in this case being the storage system offering the file's data, with the first two parties being the client and the file server. Most of the software is written for this generation and we are at the point of beginning to debug it.

References

Collins Bill, Debaney Marjorie, and Kitts David, Profiles in Mass Storage: A Tale of Two Systems , IEEE

M. Satyanarayanan, Scalable, Secure, and Highly Available Distributed File Access, IEEE Trans. Computers, May, 1990. pp. 9-21.

M. Satyanarayanan, A Survey of Distributed File Systems, CMU-CS-89-116.

Nydick, D. et al., An AFS-Based Mass Storage System at the Pittsburgh Supercomputing Center, Proc. Eleventh IEEE Symposium on Mass Storage Systems, October, 1991.

Jonathan Goldick. et al., An AFS-Based Supercomputing Environment Proc. Twelfth IEEE Symposium on Mass Storage Systems, April, 1993.

AFS Experience at the University of Stuttgart

Uwe Fischer, Dieter Mack

Regionales Rechenzentrum der Universität Stuttgart Stuttgart, Germany

Since late 1991 the Andrew File System (AFS) is in use at the University of Stuttgart. For the Centers Service Cluster comprising more than 15 RISC workstations, it is a key component in providing a single-system-image. In addition, new services like distribution of public domain or licensed software are using AFS wherever it is appropriate. On the long run, the introduction of AFS was one of the first steps into the emerging OSF/DCE technologies.

1. About the Center

The University of Stuttgart Regional Computer Center (RUS) provides computing resources and related services to the academic users of the university. Especially the supercomputing service is available to all other state universities in Baden-Württemberg and to corporations under public law as well as industrial customers.

2. Chronology

Back in 1989/90 the Center started the process to replace its midrange-type mainframes front-ending the CRAY-2 supercomputer. Focusing on UNIX-derivates as the major operating system was a joint intension. As a result, in 1991 this effort lead to the challenging task to investigate, whether and how a RISC-based workstation cluster could cover the typical mainframe-based services within an university environment.

OSF/DCE - and DME - were a name and a concept at that time, but product availability was not expected within the next two years. And this was far too long ahead in the future. Thus the Center formed a DCE project to investigate and evaluate the forthcoming technologies of distributed computing. RUS asked vendors for support, and only IBM was capable of providing this.

As DCE is not built from scratch in evolving and melting existing technologies, wherever the functional components existed as independent products, they could be used to start with. AFS being

the predecessor of the Distributed File Service (DFS), one of the extended DCE services, which might also be regarded as one of the major DCE applications, is one of them. Thus it is obvious, that our AFS- and DCE-related milestones are tightly coupled.

The first tasks after the DCE project had been formed in August 1991 were to work with the xntp time service and the Kerberos security system. In November 1991 the RUS cell *rus.uni-stuttgart.de* was the first AFS cell installed in Germany. During summer 1992 RUS took part in IBMs AIX - DCE Early Participation Program. At that time the SERVus workstation cluster was installed and AFS run in a preproduction mode. In November 1992 the DCE cell *dce.rus.uni-stuttgart.de* was configured and the port of an AFS-Client to the CRAY Y-MP 2E file server was completed. In January 1993, the service cluster joint with AFS went into full production, replacing the midrange-type mainframes which have finally been shutdown by end of March 1993. Since summer 1993 a DFS prototype is running, and in September 1993 the attempts to port the AFS-Client to the CRAY-2 have been dropped.

3. Configurations

Since late 1986 the Center runs a CRAY-2 as its main supercomputer resource. As shown in figure 1, this will be replaced in April 1994 by a CRAY C94D. The CRAY Y-MP 2E is used for high-end visualization applications and as a base for the mass storage service. In the middle layer, the SERVus workstation cluster consists of IBM RS/6000 systems with models ranging from a 580 down to 220s. Today, the cluster is going heterogeneous by incorporating a multi-processor SUN/Sparcserver. Often neglected in such a picture drawn from a centers perspective is that the workstations on campus are now several thousands in number.

The shadowed area shows the AFS file space, with the AFS server machines tightly coupled to the SERVus cluster. The CRAY file server in the AFS context is acting only as a client, and it distributes its mass storage service to all requesting systems on the campus using NFS.

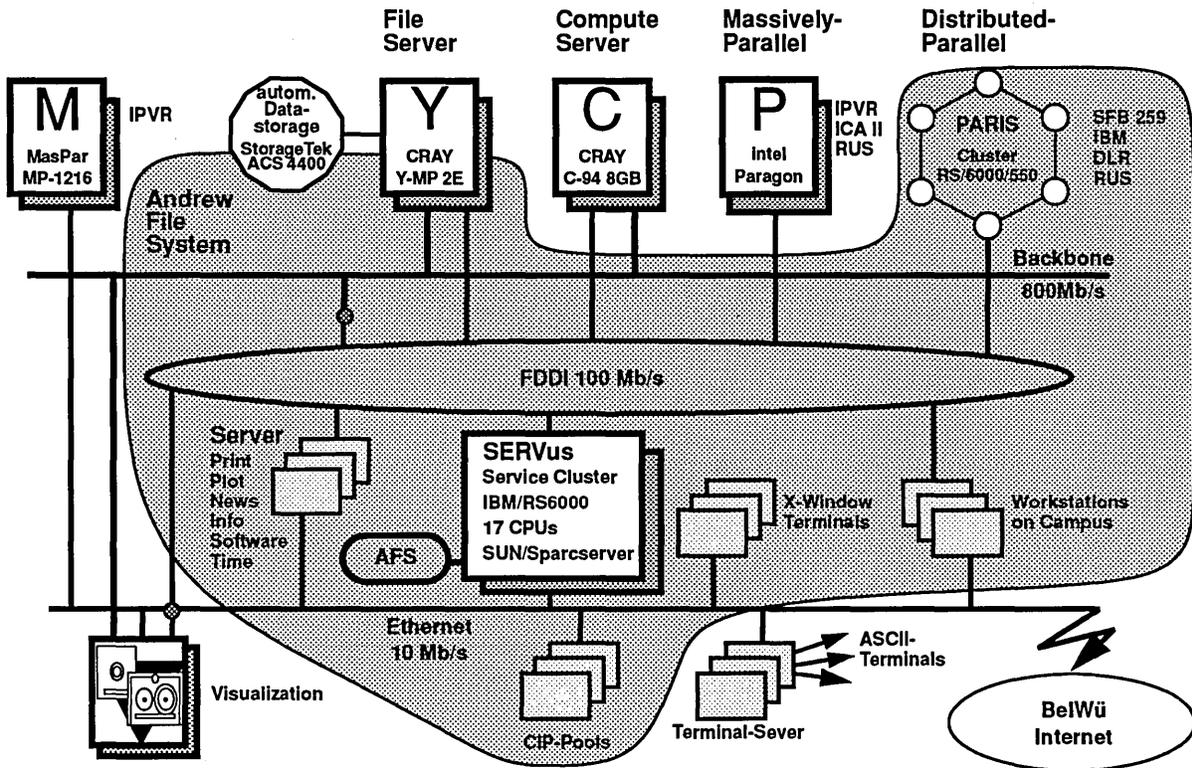


Figure 1: RUS Configuration

AFS Cells

As of today, the University of Stuttgart has 4 registered AFS cells. There is the main cell *rus.uni-stuttgart.de*, which houses all the HOME directories for the workstation cluster machines. In addition, public domain and licensed software is distributed using AFS for those platforms where AFS clients are available. And one particular university department placed its AFS file server in this cell.

There is a second cell *rus-cip.uni-stuttgart.de* dedicated to a workstation pool, to which students have public access.

Due to the deficiencies in delegating the administration of dedicated file servers, two university departments are running their own AFS cells *ihf.uni-stuttgart.de* and *mathematik.uni-stuttgart.de*. And there are more departments which show interest in using or running AFS file servers.

rus AFS Cell Configuration

The main cell has been upgraded to AFS 3.3 last month. It is based on a total of 4 file server machines. Three of them are provided by the Center, and the AFS database services are replicated on them. One is owned by a university department and run as their dedicated file server. All servers are IBM RS/6000 workstations, and the total disk capacity controlled by AFS is 19 + 8

GB. By a campus license agreement, AFS client software for a variety of platform - OS combinations is available:

DEC DECStation	Ultrix 4.0, 4.3
DEC VAXStation	Ultrix 4.0, 4.3
HP 9000 Series 700	HP-UX 9.0
IBM RS/6000	AIX 3.2
NEXT NeXTStation	NeXT OS 3.0
SGI 3000, 4000 Series	IRIX 5.0
SGI Indigo	IRIX 5.0, IRIX 5.1
SUN 3, 4, 4c	SunOS 4.1.1 - 4.1.3
SUN 4m	SunOS 4.1.2, Solaris 2.2, 2.3

4. Purpose

Key Component for the Service Cluster

As stated above, AFS is of strategic use for the SERVus workstation cluster in providing a single-system-image. First, the login authentication is done using only the Kerberos authentication service, which is part of AFS. Second, all the HOME directories of about 1600 users are within AFS. Thus every user has a consistent view and access to all his data, regardless of the single machine of the cluster he is using. In addition, the NQS batch subsystem running on the batch worker machines of the cluster has been modified to support that an users NQS job gets authenticated

and gains the appropriate authorization for file access.

Software Distribution

The basic software products available on the service cluster RS/6000 machines are installed only once in AFS, e. g. C and Fortran compiler, X11 and application software. That is essentially all except the software needed to bring up a single system.

During 2nd half of 1993, RUS developed a concept to distribute software to workstation users. The basic idea is that not every user or system administrator has to care about software installation and maintenance, instead this is done only once at the Center. All platform-OS specific executables are available for direct access by linking a specific directory into the users search PATH, and in addition the software is ready to be pulled and unfolded at the client side. As underlying technology AFS is used where appropriate, else NFS.

Thus without any big effort a huge variety of Public Domain Software will be available at every workstation. Licensed Software could be made available as well, using the AFS access control mechanism to restrict access to the authorized users. Currently this scheme is introduced to distribute PC software.

Statistics

The three rus AFS file server machines have allocated 17 disks (partitions) with 19 GB capacity. There are about 1500 user HOME directory volumes spread over 7 partitions with an assigned disk quota of 15 GB and an effective usage of 3,7 GB. For the purpose of software distribution there are about 280 volumes in use with an assigned disk quota of 9,4 GB and a data volume of 6,2 GB.

5. UNICOS Client

The AFS client installation on the CRAY Y-MP 2E happened at the time of the UNICOS upgrade from 6.1 to 7.0. Thus the first tests used a Pittsburgh Supercomputing Center code for a UNICOS 6 system, but finally a derivate of PSCs UNICOS 7.C AFS client went into production. The port itself was a minor effort.

Because the CRAY Y-MP 2E as a file server runs neither a general interactive nor batch service, the AFS client is used mainly for AFS backups. Every night, all AFS volumes are checked and for modified ones the backup clone is "vos dumped" into a special file system on the CRAY Y-MP 2E which purpose is to store backup data. This file system is subject to Data Migration Facility, thus the tape copies to 3480 cartridges stored in two ACS 4400 silos are handled by DMFs tape media specific processes.

This solution replaced the previous procedures, which took the volume dumps on a RS/6000 based AFS client and wrote the dump files via NFS to the file server. Today's procedures are more reliable and robust, using AFSs RX protocol rather than UDP/IP. Better transfer rates are assumed as well, but an honest

comparison is not possible, because at the time of change serious NFS bottlenecks at the file server side had been detected and eliminated.

The Center intended to integrate the CRAY-2 supercomputer into its AFS environment. But due to major differences in UNICOS internal structures the port of the AFS client code turned out to be a complicated and time-consuming task. With respect to the imminent replacement of the CRAY-2 this effort has been dropped.

6. Key Concepts

Security

AFS security comprises authentication, which is based on a Kerberos variant, and authorization via access control lists (ACLs), placed on directories.

The proof of an users identity is not guaranteed by local authentication but by a Kerberos token, which has to be acquired from a server on the network. Authorization to access data can be granted to individual users as well as user-defined groups. This allows for much finer granularity of access rights than the UNIX mode bits.

Volume (Fileset) Concept

One of the fundamental technologies of AFS is the volume concept. The name of these conceptual containers for sets of files changes to *fileset* in DFS. A volume corresponds to a directory in the file tree, and all data underneath this subtree, except for other volume mount directories, is kept in that volume.

Volumes are the base for *location transparency*. AFS is able to determine, on which file server and physical partition a volume resides. Thus once an AFS client has been set up, the user is fine. Something like registering new NFS file servers and file-systems at every client side is not necessary.

Volumes *disconnect the files from the physical media*. There are mechanisms to move, transparently to the user, volumes from one partition to another, even between different file servers. Thus disk usage balancing will be a manageable task. In the NFS context, assigning user directories to new file systems could not be realized without impacting the users.

Volumes represent *a new intermediate granularity level*. Running data migration on file systems, the disk space is not the limiting factor any more. Approaching 1 million files on the CRAY Y-MP 2E mass storage file system, as expected it turns out that the number of files (i-nodes) is the critical resource.

Volumes can be cloned for *replication or backup* purposes. Through replication high availability of basic or important data is achievable. In addition, network and access load could be balanced.

Global File System

Server and client machines are grouped into organisational units called cells. But it is possible to mount volumes which belong to foreign cells. This allows for the construction of a truly global file system. The AFS file space on the Internet currently comprises about 100 cells.

PSCs Multiple-Residency Extensions

There is a great potential in this feature. First, it provides the hooks to integrate hierarchical storage management schemes into the distributed file system. In addition, the separation of file data from file attributes is the first step into third party transfers. Thus functionality similar to NSL UniTree might be achieved.

7. User Barriers

Today's users are used to work with NFS. And in small configurations using NFS is quite straightforward, especially when the user is relieved of system administration tasks, and the system administrator has done a good job. Starting with AFS requires some additional setup work and extra knowledge.

File Access Control

First, the user has to acquire a token. Although this can be achieved transparently by the login process, the token has a finite lifetime. This has to be considered and taken care of. Access control is governed not only by the UNIX mode bits, but also by the ACLs. The user has to be aware of this and to familiarize himself with some new procedures and commands.

AFS Cache Configuration

An AFS client has to provide some cache space, either memory-resident or disk-based. It's highly recommended that the disk cache is provided in a separate file system. That's due to the fact that the cache manager relies on the specified cache capacity. And if the file system that contains the cache runs out of space, the results are unpredictable.

But in most cases the available disks are completely divided into used partitions, thus providing a separate file system for the AFS cache often requires disk repartitioning. And this is not an easy task.

8. Direction

Consulting Transarc's worldwide CellServDB gives an indication about the presence of AFS. Although an AFS cell has not to be registered there, in some sense representative numbers can be derived using that information. As of March 1993, a total of 12 AFS cells have been in place in Germany, all running at universities except for one at a research laboratory. Seven of those cells are based in the local state Baden-Württemberg - Germany consists of 16 states -, five of them in the capital Stuttgart. Taking those numbers it becomes quite obvious that there are focal points.

The general trend, that workstation clusters are replacing general purpose mainframes, might be more common in university-like environments than at commercial sites. And distributed computing technologies are of key importance for the success of integrating not only the clusters itself but linking all general and specialized computers together. Having this functionality in place, distributed computing will not stay restricted to single site locations. As an example, the state universities started collaborated efforts on software distribution and a "state-wide" file system, the latter can be provided by OSF/DCE's Distributed File Service.

Provided DCE/DFS is mature enough, and supported by a sufficient number of vendors, RUS might decide to switch from AFS to DFS servers by end of this year. One of the major arguments is to use the CRAY Y-MP 2E as a file server in the DCE/DFS context.

References

- Goldick, J. S., Benninger, K., Brown, W., Kirby, Ch., Nydick, D. S., Zumach, B. "An AFS-Based Supercomputing Environment." Digest of Papers, 12th IEEE Symposium on Mass Storage Systems, April 1993.
- Lanzatella, T. W. "An Evaluation of the Andrew File System." Proceedings, 28th CRAY User Group meeting, Santa Fe, September 1991.
- Mack, D. "Distributed Computing at the Computer Center of the University of Stuttgart." Proceedings, 29th CRAY User Group meeting, Berlin, April 1992.
- Mack, D. "Experiences with OSF-DCE/DFS in a 'Semi-Production' Environment." Proceedings, 33rd CRAY User Group meeting, San Diego, March 1994.
- Nydick, D., Benninger, K., Bosley, B., Ellis, J., Goldick, J., Kirby, Ch., Levine, M., Maher, Ch., Mathis, M. "An AFS-Based Mass Storage System at the Pittsburgh Supercomputing Center." Digest of Papers, 11th IEEE Symposium on Mass Storage Systems, October 1991.
- OSF. "Introduction to DCE" Preliminary Revision (Snapshot 4) - for OSF Members only, June, 1991.
- Transarc. "AFS System Administrator's Guide" FS-D200-00.10.4, 1993.
- Wehinger, W. "Client - Server Computing at RUS." Proceedings, 31th CRAY User Group meeting, Montreux, March/April 1993.
- Zeller, Ch., Wehinger, W. "SERVus - Ein RISC-Cluster für allgemeine Dienstleistungen am Rechenzentrum der Universität Stuttgart."
to be published

CRAY Research Status of the DCE/DFS Project

Brian Gaffey
CRAY Research, Inc.
Eagan, Minnesota

DCE is an integrated solution to distributed computing. It provides the services for customers to create distributed programs and to share data across a network. These services include: timing, security, naming, remote procedure call and a user space threads package. A key component of DCE is the Distributed File System (DFS). This talk will review CRI's plans for DCE, relate our early experiences porting DCE to UNICOS and describe the issues related to integrating DCE into UNICOS.

1. Distributed Computing Program

DCE is part of the Distributed Computing Program. The Distributed Computing Program defines the overall requirements and direction for many sub-programs. These sub-programs cover the major areas of the system needed to support the distributed computing model. More detail for each can be found in the program roadmaps. The intent is to show how each of these sub-programs supports the goals of distributed computing. The highest level description is called the Distributed Computing RoadMap. It is the highest-level description of the entire Program. A RoadMap exists for each of the sub-programs which in turn is a summary of product presentations. The other roadmaps are as follows:

- Distributed Job
- Distributed Data
- Connectivity
- Distributed Programming
- Network Security
- Visualization
- Distributed Administration

OSF DCE is covered in three of the RoadMaps : Distributed Programming which includes threads, RPC/IDL and naming; Distributed Data which includes the Distributed File System (DFS); and in Network Security which includes the DCE Security Services.

2. Distributed Computing Framework

This Framework represents a future CRI architecture that meets the needs of Distributed Computing. All Programs are represented but not necessarily in complete detail. Components of Federated Services (X/Open Federated Naming and Generic Security Switch [GSS]) such as NIS and Kerberos also exist today but are not federated. Distributed System Administration will track industry standards such as OSF's DME or COSE's working group. Meanwhile CRI will provide products to address the needs of customers in a heterogeneous environment. Nearly everything in our Framework is a standard or a de-facto standard. Nearly all of the software in our Framework was obtained from outside CRI or will be obtained from outside. The Framework represents the elements which are essential to high performance supercomputing and to our strategy of making connections to Cray systems easy.

OSF DCE is a key element in the Framework. DCE services will co-exist with ONC and ONC+ services at the RPC, Distributed File System, Security and Naming levels of the model. New services, such as CORBA will be built on top of DCE services.

3. Product Positioning

Architecturally DCE lies between the operating system and network services on one hand, and the distributed applications it supports on the other. DCE is based on the client/server model of distributed computing. DCE servers provide a variety of services to clients. These services are of two types: Fundamental Services: Tools for software developers to create the end-user services needed for distributed computing, i.e. the distributed applications and Data Sharing Services: Distributed file services for end-users and software developers. Clients and servers require common networking protocol suites for communication; they may run DCE on the same or different operating systems.

CRI will support all of the client services of DCE. CRI will also support the DFS server facility. CRI has no plans to support security, directory or time servers on

UNICOS.

4. DCE Component Review

The DCE source is a integrated set of technologies. The technologies rely upon one another to provide certain services. For example, all of the services rely on threads and most of the services make use of rpc to accomplish their task. The following is a review of the major components of DCE.

4.1. Threads in user space

In many computing environments, there is one thread of control. This thread of control starts and terminates when a program is started and terminated. With DCE threads, a program can make calls that start and terminate other threads of control within the same program. Other DCE components/services make calls to the threads package and therefore depend on DCE threads.

Cray already provides the following mechanisms which allow for additional threads of control:

1. Autotasking
2. Multitasking (macrotasking)
3. Multitasking (microtasking)
4. UNICOS libc.a multitasking

Autotasking allows a programmer to automatically insert directives that use items 2 and 3. Items 2 and 3 are a set of UNICOS library routines which provide multiple threads of execution. They may or may not use 4 to manage the multiple threads of execution. Item 4 is a low level set of UNICOS system calls and library routines which provide a multithreaded environment. The interfaces provided by these 4 mechanisms are Cray proprietary and therefore not a "standard."

The DCE threads interface is based on the Portable Operating System Interface (POSIX) 1003.4a standard (Draft 4). This interface is also known as the Pthreads interface. DCE threads has also implemented some additional capabilities above and beyond the *Pthreads* interface.

In CRI's product, the DCE thread interface routines are mapped directly to existing Multitasking (macrotasking) routines. This could be configured to restrict all threads to be within one real UNICOS thread or to allow for multiple UNICOS threads. With this approach, existing Multitasking (macrotasking) implementations function correctly. The downside to this approach is that all of the DCE Threads functionality

cannot be provided (in the short term). For example, multiple scheduling algorithms cannot be requested.

4.2. Threads in the kernel

In addition to threads in user space, the DFS kernel components require threads in the kernel. Actually, DFS relies on rpc runtime libraries which use the pthreads interface. The pthreads interface in the kernel maps into newproc() which creates a new process in the kernel. This process is scheduled as a normal process not as a thread.

4.3. RPC and IDL

RPC, "Remote Procedure Calls" allows programmers to call functions which execute on remote machines by extending the procedure interface across the network. RPC is broken into kernel RPC, used only by DFS, and user-space RPC which is used by most other DCE components. DCE provides a rich set of easy to use interfaces for creating remote processes, bind to them and communicating between the components.

Interfaces to RPC functions are written in a C-like language called the "Interface Definition Language". These interfaces are then compiled with the IDL compiler to produce object or C source code stubs. The stubs in turn are linked with the programmers code and the RPC libraries to produce client and server executables.

A few technical items to note:

- communication, naming and security are handled transparently by the RPC runtime library
- the network encoding is negotiable, but currently only Network Data Representation (NDR) is supported
- "receiver makes right" which means that machines with similar network data types will not need to do data conversions
- DCE RPC supports three types of execution semantics : "at most once", idempotent (possibly many times) and broadcast
- RPC will run over TCP/IP or UDP (with DCE RPC providing transport mechanisms)

CRI plans to rely on ONC's NTP protocol for clock synchronization since it is already implemented and a single system can not have two daemons changing the system clock.

4.4. Directory Services

Directory services is the naming service of DCE. It provides a universally consistent way to identify and locate people and resources anywhere in the network. The service consists of two major portions, the Cell Directory Service (CDS) which handles naming within

a local network or *cell* of machines and the Global Directory Agent (GDA) which deals with resolution of names between cells.

Applications requiring directory information will initiate a CDS client process on the local machine called a *Clerk*. The Clerk resolves the application's query by contacting one or more CDS Servers. The Servers each physically store portions of the namespace with appropriate redundancy for speed and replication for handling host failures. Queries referencing objects external to the local cell will access the GDA to locate servers capable of resolving the application's request.

When the GDA is resolving inter-cell queries, it uses either the Global Directory Service (GDS) or Domain Name Service (DNS). GDS is a X.500 implementation that comes with the DCE release while DNS is the Internet distributed naming database. Both of these services will locate a host address in a remote cell and pass this value back to the CDS Clerk who will then use it to resolve the application's query.

4.5. Security Services

DCE security services consist of three parts : the Registry service, the authentication service and privilege service. The Registry maintains a database of users, groups, organizations, accounts and policies. The authentication service is a "trusted third party" for authentication of principals. The authentication service is based on Kerberos version five with extensions from HP. The Privilege Service certifies the credentials of principals. A principal's credential consist of its identity and group memberships which are used by a server principal to grant access to a client principal. The authorization checking is based on POSIX Access Control Lists (ACLs). The security service also provides cryptographic checksums for data integrity and secret key encryption for data privacy.

Supporting DCE security on UNICOS can lead to compatibility problems with current and future UNICOS products. Specifically, DCE ACLs are a superset of POSIX ACLs and DCE's Kerberos is based on version five whereas UNICOS's Kerberos is based on version four. We don't believe an application can be part of a V4 realm and a V5 realm. Also, the two protocols aren't compatible, but it is possible to run a Kerberos server that is capable of responding to both version 4 and version 5 requests.

4.6. Distributed File System

The Distributed File System appears to users as a local file system with a uniform name space, file location transparency, and high availability. A log-based physical file system allows quick recovery from server

failures. Replication and caching are used to provide high availability. Location transparency allows easier management of the file system because an administrator can move a file from one disk to another while the system is available.

DFS retains the state of the file system through the use of tokens. Data can be cached on the clients by granting the client a token for the appropriate access (read/write). If the data is changed by another user of the file, the token can be revoked by the server, thus notifying the client that the cached data is no longer valid. This can't be accomplished with a stateless file system, which caches data for some period of time before assuming that it is no longer valid. If changes are made by another user, there is no mechanism for the server to notify the client that its cached data is no longer valid.

DFS supports replication, which means that multiple copies of files are distributed across multiple servers. If a server becomes unavailable, the clients can be automatically switched to one of the replicated servers. The clients are unaware of the change of file server usage.

DFS uses Kerberos to provide authentication of users and an access control list mechanism for authorization. Access Control Lists allow a user to receive permission from the file server to perform operation on a particular file, but at the same time access to other files can be denied. This is an extension of UNIX file permissions, in that access can be allowed or denied on a per user basis. UNIX allows authorization based on group membership, but not to a list of individual users.

DFS is a log-based file system which allows quick recovery from system crashes. Most file systems must go through a file system check to ensure that there was no corruption of the file system. This can occur because much of the housekeeping information is kept in main memory and can be lost across a system crash. In contrast, DFS logs every disk operation which allows it to check only the changes made to the disk since the last update. This greatly reduces the file system check phase and consequently file server restarts.

To summarize, the use of a token manager ensures shared data consistency across multiple clients. A uniform name space is enforced to provide location transparency. Kerberos authentication is used and access control lists provide authorization. DFS allows its databases and files to be replicated, which provides for reliability and availability. It can interoperate with NFS clients through the use of protocol gateways.

Cray's initial port of DFS won't include the Episode file system. This means that log-based recovery and cloning won't be available. Cloning is the

mechanism used for replication.

4.7. LFS

OSF has selected the Episode file system to be the local file system for the OSF/DCE product. In the initial port of DCE it was decided to retain the UNICOS file system instead of LFS. However, there are significant features of the Episode file system that when used with DFS provide reliability and performance enhancements. The most important feature is the ability to support multiple copies of data. This provides redundancy for reliability/availability, and increased throughput and load balancing

CRI will evaluate different log based file systems. Decide on the best alternative for Cray from available log base file systems. The current possibilities include Episode from Transarc, Veritas, write our own, Polyscenter from DEC, and others. This evaluation must first produce a clear set of requirements which will be used to select the best choice for Cray Research, Inc.

5. Comparison to ONC Services

Almost all components of DCE have corresponding ONC services. This section is a quick overview of the technologies available on UNICOS which can be used now to support distributed computing and distributed data.

Both DCE and ONC have an RPC mechanism, which have different protocols. To write a program using ONC RPC, a user makes use of a tool called RPCGEN, which produces stubs. To write a program using DCE RPC, a user makes use of an IDL compiler which also produces stubs. ONC RPC uses XDR for data translation, while DCE RPC uses IDL. DCE RPC has an asynchronous option. User programs may use either DCE RPC or ONC RPC, but not both. The client and server portions of an application must both use either DCE RPC or ONC RPC.

NFS is a stateless file system. DFS relies on state information and uses tokens to control file access. A user program could access files that exist in a DFS file system and files in an NFS file system, however a file must reside in only one file system.

Network Information Service (NIS) is ONC's directory service. It interfaces to the Domain Name Server to extract internet naming and addressing information for hosts. DCE's CDS is similar in this respect. The protocols for NIS and CDS are incompatible.

DCE's User Registry doesn't have a corresponding ONC service, but it will have to coexist or be integrated with the UNICOS User Data Base. Both environments support Kerberos for network security.

ONC's time service is called Network Time Protocol (NTP). DCE's time protocol, DTP isn't compatible at the protocol level. It is possible to tie together an NTP time service with a DTP time service by having the DTP server get its time from NTP.

6. CRI's DCE Plans

The DCE Toolkit has been released. It supports the client implementation of the DCE core services (threads, rpc/idl, security and directory). We rely on ONC's NTP to provide the correct time. The Toolkit passes over 95% of the OSF supplied tests. The test failure are in the area of threads scheduling. Our Toolkit is built on top of libu multi-tasking. Since libu has its own scheduler we removed the OSF supplied scheduler.

CRI does not provide documentation or training for DCE. Both of these services can be obtained from OSF or from third parties.

CRI's next release of the OSF technology will be in two parts : the DCE Client and the DFS Server. The DCE Client will incorporate an updated version of the Toolkit and the DFS client. The DCE DFS Server includes the full OSF/DCE DFS server, providing transparent access to remote files that are physically distributed across systems throughout the network. Implementation of DFS requires UNICOS support for the following new features: pthreads, krpc, and vnodes. These features are available in UNICOS 8.0. Cray DCE DFS Server is planned to be available in mid 1994. In addition to UNICOS 8.0, Cray DCE Client Services is a prerequisite for this product.

The Cray DCE Client Services product provides all of the functionality of the toolkit as well as DFS client capabilities. With the introduction of the Cray DCE Client Services, the Cray DCE Toolkit is no longer be available to new customers. Since Cray DCE Clients Services requires UNICOS 8.0, a transition period has been established for the upgrade of existing Cray DCE Toolkit customers to Cray DCE Client Services. The transition period extends from the release of Cray DCE Client Services until one year after the release of UNICOS 8.0. During this transition period, the Cray DCE Toolkit will continue to be supported on UNICOS 7.0 and UNICOS 7.C systems. Cray DCE Toolkit licenses includes rights to DCE Client Services on a "when available" basis. There is no upgrade fee.

In future product releases, CRI will provide support for a log-based file system (LFS). We will investigate Episode; the OSF supplied file system, and other log based file systems which support the advanced fileset operations. CRI will also integrate DFS will other components of UNICOS (eg. DMF, SFS, accounting etc). Finally, we intend to track all major releases of the

DCE technology from OSF.

7. DFS Advantages

In the DFS distributed file environment, users work with copies of files that are cached on the clients. DFS solves problems that arise when multiple users on different clients access and modify the same file. If file consistency is to be controlled, care must be taken to ensure that each user working with a particular file can see changes that others are making to their copy of that file. DFS uses a token mechanism to synchronize concurrent file accesses by multiple users. A DFS server has a token manager which manages the tokens that are granted to clients of that file server. On the clients it is the cache manager's responsibility to comply with token control.

Caching of information is transparent to users. DFS ensures that users are always working with the most recent version of a file. A DFS file server keeps track of which clients have cached copies of each file. Servers such as DFS servers that keep such information, or 'state' about the clients and are said to be 'stateful' (as opposed to 'stateless' servers in some other distributed file systems). Caching file data locally improves DFS performance. The client computer does not need to send requests for data across the network every time the user needs a file; once the file is cached, subsequent access to it is fast because it is stored locally.

Replication improves performance by allowing read-only copies to be located close to the user of the data. This load balancing of data locations reduces network overhead. All DFS databases (fileset location, backup, update) use an underlying technology which allows replication. This further improves performance and allows more reliability. DFS allows for multiple administrative domains in a cell. Each domain is controlled via a number of administrative lists which can be distributed.

8. DCE and UNICOS Subsystems

8.1.

DMF

Since DFS uses NC1 has its local file system and has its cache the integration is transparent. DMF can migrate and unmigrate DFS files at any time. In future releases we will study the possibility of a special communications path between DMF and DFS.

8.2. NQS

In a future NQS release, NQS will be able to access and use DFS files for retrieving job scripts and returning output.

8.3. SFS

In future releases the Shared File System and DFS will be integrated in order to maintain high performance and network wide consistency. Our initial work will be to synchronize DFS tokens and SFS semaphores. This will ensure that users outside the cluster can access files in the cluster while maintaining data integrity. Next, we will extend a facility already within DFS called the Express path. The Express path will allow DFS clients within the cluster to access data in the cluster without moving the data.

8.4. Security

Our initial release requires users to validate themselves to DCE before using DCE services. This validation occurs during the initial entry into the DCE. If that entry occurs on UNICOS then a second is required. If the entry occurs in the network then no second UNICOS login is required. DCE security is separate and distinct from MLS. However, since DCE makes use of the standard network components that are part of MLS, some of the benefits of MLS apply to DCE. Later releases of DCE will make use of other components of MLS.

9. DCE's impact on UNICOS

Since the reference implementation of DFS is based on the latest file system technology in System V (vnodes), it was necessary to change UNICOS to support vnodes. To support vnodes, all of the old FSS (file system switch) code had to be removed and replaced with vnode code. This required all existing file systems (eg. NC1 and NFS) to change from FSS calls to VFS calls. DFS also requires rpc and threads in the kernel. The rpc code is a copy of some of the user space rpc code. The threads support in UNICOS is completely different from the user space threads code. In the kernel, we implemented threads through direct system calls. DFS and rpc contain large amounts of code. This is much more of an issue for real memory system like CRAYs than it is for other systems. The other components of DCE also contain large amounts of code but since they are in user space they can swap out. The user space libraries require threads. We implemented DCE threads on top of libu multitasking. This has the advantage of easier integration with existing libraries and tools (eg. cdbx). However, there are some restrictions.

10. Current Status

The DCE toolkit is released. The Toolkit supports client implementations of all core components except DFS. The Toolkit allows UNICOS systems to participate in a DCE environment. The Toolkit is at the OSF

1.0.1 level. The components include :

- Threads
- Rpc/idl
- Directory
- Security
- Time API

UNICOS 8.0 contains the infrastructure to support DFS. All DFS products will require 8.0. The infrastructure includes vnodes, kernel rpc and kernel threads.

The DCE client product will include all of the Toolkit components (updated to the 1.0.2 level) and the DFS client. Currently, the DCE Client passes all of the Toolkit tests and the NFS Connectathon tests. More than 90% of the DFS tests are working. The DCE DFS Server product will include support for the DFS servers. It currently passes Connectathon as well. A major undertaking, in conjunction with other OSF members, is underway to multi-thread DFS. LFS (aka Episode) has been evaluated but will not be part of the initial release.

11. Summary

CRI has a DCE Toolkit available and plans for a DFS product in third quarter 1994. CRI is committed to track DCE and enhance it.

SCinet '93 — An Overview

**March 14, 1994, San Diego California
Cray User's Group Meeting**

By: Benjamin R. Peek

**Peek & Associates, Inc.
5765 SW 161st Avenue
Beaverton, OR 97007-4019
(503) 642-2727 voice
(503) 642-0961 fax**

SCinet '93 at the Supercomputing '93 Conference in Portland, Oregon

SCinet '93 — An Overview

SCinet '93

SCinet '93 got underway during SUPERCOMPUTING '92. Bob Borchers had asked me to help locate someone locally (in Oregon) that could handle the SCinet '93 responsibilities. At that point, I was just beginning to understand requirements for the Local Arrangements responsibilities and had not quite received a commitment from Dick Allen to take the Education Chair. I decided to spend time during SUPERCOMPUTING '92 at the SCinet booth in Minneapolis with Dick Kachelmeyer and the SCinet '92 crew. I was most impressed with the level of service that Dick and his crew were able to give to all of the exhibitors and researchers at SC'92. The conference period went well for me but I admit that I was buried in more information than I could handle, especially the SCinet participants and the Local Arrangements scope and organization. I also attended meetings with the SC'92 Education folks, along with Borchers, Crawford, Allen and others, to better understand what they were doing. Basically, my time was spent collecting a lot of information.

Time track

During December 1992, things got even more interesting. Dona Crawford, Bill Boas, and several other folks began organizing the whole idea of the National Information Infrastructure Testbed, spawned out of the success of SCinet '92 and a desire to establish a testbed with the attributes of SUPERCOMPUTING experimentation in terms of industry, academia, and government research participation.. This NIIT idea became an active meeting just before Christmas 1992. On December 17, 1992, Dona Crawford and several interested folks met in Albuquerque to discuss a permanent network testbed and, what that would mean, who might be interested, who would benefit, who and how could such an idea be funded, and so forth. Because demonstrations could result from the NIIT idea that might be showcased at SC'93 on SCinet '93, Dona invited me to participate. I had other commitments and was unable to make the meeting. Clearly, it was a productive meeting.

The NIIT showcase demonstrations for SCinet '93 were massive, impressive, and technically superior. The showcase of technology for ATM over the NIIT infrastructure and SCinet '93 was impressive enough that ATM, as a protocol and technology, moved ahead in its deployment schedule by at least 18 months during 1993, perhaps even more, in my view.

Starting in March 1993, Mike Cox and I began the process of collecting requirements by survey for SCinet '93. The survey was sent by fax, email, U.S. Mail, and by phone to all participants in SUPERCOMPUTING '92, specifically to all of the folks involved in networking or SCinet. The survey was thought, at this early date, to magically collect all of the information that we would need to begin the network design process for all of the networks planned. The survey included questions about Ethernet™, FDDI, ATM, HiPPI, Serial HiPPI, Fibre Channel, and SONET. Some conference calls were made.

Perhaps the next major event for SCinet '93 came in June 1993. Dona Crawford planned a SC'93 Program Committee meeting in Portland for June and it seemed convenient and pertinent that we have the organizing committee meeting for SCinet participants as well, especially since some were traveling to Dona's meeting already. The meeting at the Benson Hotel started at 8:30a with about 30 SCinet folks attending in the morning. Bob Borchers introduced the meeting and gave the group an overview of SUPERCOMPUTING

in general, the expected number of attendees at SC'93, and other general information. I gave an overview of how I planned to organize and run the SCinet '93 committee. We reviewed preliminary plans including permanent infrastructure at the Oregon Convention Center and the fund raising requirements to make such an idea possible.

We heard from Jim McCabe (who committed to be vice chair technology at this meeting), Mark Wilcop, U.S. West (telephone long lines and other issues), Doug Bird, Pacific Datacomm on Ethernet and FDDI, Mark Clinger, Fore Systems on ATM, Bill Boas, Essential Communications on HiPPI, Dona Crawford, Sandia on NIIT, and the Intel folks on experiences at SCinet '92. The meeting was a great success and got the team working on all of the serious issues. Other ideas were represented including the National Storage Labs (NSL) plans and of course, Tom Kitchens, DoE and Fran Berman, UCSD presented the Heterogeneous Computing Challenge and the Le Mans FunRun ideas. A massive amount of activity was initiated and specifically, the connectivity data collection functions went into high gear.

The next major "event" was a meeting two months later in Minneapolis (August.). NetStar was our host and I paid for lunch for around 35-40 people. At the NetStar meeting, we were able to reach consensus on many of the major functions and tasks for SCinet '93. Connectivity requirements were proving hard to come by (no one knows what they will do at a SUPERCOMPUTING conference until a few (very) weeks before the conference (in terms of equipment, projects, etc.). Before the NetStar meeting, email surveys were sent a total of three times, faxes were sent twice (to everyone not yet responding. Finally, in late August, Linda Owen at Peek & Associates, Inc. began the telephone process and called everyone that we knew to collect additional requirements. Jim McCabe and the various project leaders were also calling each participant.

During September and specifically on Labor Day, Jim McCabe and his crew from NASA spent the weekend at the Oregon Convention Center making a physical inventory of the conduits system, inspecting all aspects of the center layout, and getting preliminary implementation plans in place for the physical networks defined at that point.

An October meeting was held in Albuquerque at the same time as the SC'93 steering committee meeting in Portland on October 22 -23, 1993. I was unable to attend due to the Steering Committee meeting in Portland. The purpose of the meeting was final review on all network designs for the conference. Each of project leaders (for Ethernet, FDDI, ATM, HiPPI, FCS, plus external connectivity) gave final reports on the design of their specific network. Issues and problems were discussed, especially those that related to interfaces between networks and those issues dependent on loaned equipment from the many communication vendors that were participants. McCabe reported that the meeting was very productive. Meeting reports were received from most of the project leaders outlining problems, proposed solutions, and unfilled requirements.

Prestaging was also planned for October at Cray Research, Superservers Division in Beaverton, OR. The space was donated, power was arranged, facilities were organized, and no one showed up. Prestaging is just not a concept that will work for SCinet, in my view. The elements of the conference do not come together clearly, in enough time, for the big players in the network like Cray Research, IBM, HP, MasPar, Intel, Thinking Machines, etc. to take advantage of prestaging. The network exhibitors and vendors seemed much more prepared to take advantage of the early timeframe. However, not all of them. There is also the logistics problem of getting all of the people scheduled for a prestaging activity, at the same time. It does not do a lot of good to prestage equipment that must talk to other equipment if the "other folks" cannot be there.

As we moved into mid-October, it became clear that the size, scope, and complexity of SCinet '93 was perhaps four times SCinet '92. I expect a step function for SCinet '94 as well. The costs have also exploded due to the cost of single and multimode connectors, fiber optic cable (the volume of it continues to increase), and the labor (very skilled and expensive) to install and test such networks. Fiber optic installation is also slower than conventional network technology. Rushing a fiber optic installation just results in rework. During October, Peek & Associates, Vertex and RFI were installing networks at the Oregon Convention Center at every available time slot. The convention center had many other conferences that made working in the center really difficult. We would work for 2-4 hours at a time, generally from midnight to early morning, between events.

The real press started the first week of November. We were in the Oregon Convention Center installing networks almost continuously from October 30 until November 15 with the pace becoming more intense with each passing day.

External Networks

SCinet was connected to the outside world in many ways including DS3, OC3, gigabit fiber optics (HiPPI and FDDI), SONET, etc. External networks were developed, managed and setup primarily by Mark Wilcop. Mark worked with many phone companies, research organizations such as NIIT and Sandia, ANS for the Internet connection, and with the SCinet team to establish the finest set of external networks that have been established to date for a technical conference such as SUPERCOMPUTING '93.

The Sandia "Restricted Production Computing Network" connected Sandia, New Mexico to Sandia, California over a 1100 mile DS3 truck. This network was terminated at both ends with AT&T BNS-2000 switching equipment that integrated both local FDDI networks and ATM/SONET testbeds. An additional link was developed at OC3 that covered 1,500 miles of connectivity from Albuquerque to Portland. This connectivity included several players and was arranged primarily by Mark Wilcop (U.S. West).

The NIIT experiment was massive. It included connections to the University of New Hampshire; the Earth, Ocean, Space Institute in Durham; Ellery Systems in Boulder, CO; Los Alamos; Sandia in Albuquerque; Sandia in Livermore, CA; and Oregon State University in Corvallis, OR.

An AT&T Frame Relay capability connected NIIT to University of Berkeley, CA; University of California, Santa Barbara; NASA/Jet Propulsion Lab, Pasadena, CA; High Energy Astrophysics Div., Smithsonian Astrophysical Observatory, Cambridge, MA; and AT&T in Holmdel, NJ.

Private Boeing ATM network to Seattle, WA with Fore Systems equipment and participation. This experiment developed and used switched virtual circuits (SVCs) for the first time in a trial over the switched public network.

Fiber Optic Backbone Network at Oregon Convention Center

A backbone network was installed in the Oregon Convention Center. The backbone network is described elsewhere in this document (Oregon Convention Center Support Infrastructure Letters section). The backbone has from 12 - 48 single and multimode fiber optic connections running throughout the convention center.

The diagram attached shows the network details.

Ethernet

The Ethernet network was very extensive and had hundreds of connections. Ethernet connected to both the FDDI networks and to the ATM backbone. The Ethernet equipment was supplied by ODS with conference support from both Pacific Datacomm in Seattle and ODS in Dallas.

Fiber Distributed Data Interface (FDDI)

FDDI networks interfaced with the ATM backbone. The FDDI equipment was supplied by ODS with conference support from both Pacific Datacomm in Seattle and ODS in Dallas.

Asynchronous Transfer Mode (ATM)

ATM was one of the most interesting ATM experiments to date. The ATM network served as the backbone network for the conference. The ATM activity was managed by Marke Clinger with Fore Systems, Inc. There were many firsts, specifically the first use of switched virtual circuits (SVCs) over the public switched network, the most extensive backbone network based on ATM, and in general, the use of production ATM equipment from a mixed set of vendors over a mixed set of media including the public switched network, private networks, the OCC backbone network, etc.

SCinet'93 Network Diagrams

On the following pages are detail diagrams of each of the networks from SCinet'93 as well as figures describing the network experiments. All of the diagrams go in here. This page number will be used appending A, B, C, D, etc. to each diagram.

High Performance Parallel Interface (HiPPI)

HiPPI is a suite of communications protocols defining a very high-speed, 800 M/bps channel and related services. Using crosspoint switches, multiple HiPPI channels can be combined to form a LAN architecture.

The HiPPI protocol suite consists of the HiPPI Physical Layer (HiPPI-PH), a switch control (HiPPI-SC), HiPPI Framing Protocol (HiPPI-FP), and three HiPPI Upper Layer Protocols (UPLs). The UPLs define HiPPI services such as IEEE 802.2 Link Encapsulation (HiPPI-LE), HiPPI Mapping to Fibre Channel (HiPPI-FC), and HiPPI Intelligent Peripheral Interface (HiPPI-IPI).

SCinet '93 implemented both HiPPI-IPI and HiPPI-LE. There was the possibility of HiPPI-FC but no experiment was conducted due to lack of time. It would have been an interesting experiment and there was a convenient Fibre Channel network that could have been used.

The HiPPI team was headed by Randy Butler with NCSA. Randy did an outstanding job of designing, building and operating the largest HiPPI network built to date. The network extended approximately 28 miles to Beaverton, OR connecting a Paragon at Intel SSD to the Oregon Convention Center with three separate fiber optic networks. There were several private HiPPI networks dedicated to a specific application or demonstration. In

all, there were more than 30 HiPPI connections over single mode fiber, multimode fiber, 25 and 50 meter HiPPI cables, connected by a wide assortment of HiPPI switches, routers, and extenders. The system operated flawlessly throughout the conference.

See the attached figure from Network System describing the HiPPI networks for SC'93.

HiPPI Serial

In addition to the above, there were two separate HiPPI Serial links and one FDDI connection to Intel in Beaverton, approximately 24 miles one way, over three dedicated fiber optic private networks configured using both U S West and General Telephone dark fiber connecting the Oregon Convention Center to Intel in Beaverton, OR. These serial experiments used BCP extenders everywhere. The results were impressive. Thirteen terabytes of information per day moved between Intel's Paragon configuration in Beaverton and Intel equipment at the Supercomputing conference within the Oregon Convention Center.

These fiber optic networks experienced unusually low bit error rates and were functional throughout four days of the conference. Much of the "impossibility" of doing gigabit networks at the WAN level evaporated during the experiment. Engineers and management, across a spectrum of companies and organizations, were convinced that such technology could be implemented

Experiences with OSF-DCE/DFS in a 'Semi-Production' Environment

Dieter Mack

Computer Center of the University of Stuttgart

Talk presented at CUG, San Diego, March 1994

Abstract: RUS has been running a DCE cell since late 1992, and DFS since summer 1993. What was originally a mere test cell is now being used as the day-to-day computing environment by volunteering staff members. The expressed purpose is to obtain the necessary skills and prepare for the transition from an AFS based distributed environment to OSF-DCE. We describe experiences made, difficulties encountered, tools being developed and actions taken in preparation for the switch to DCE/DFS.

Introduction.

The Computer Center of the University of Stuttgart is a regional computer center and its purpose is to provide computing resources to the University of Stuttgart, the other universities of the state of Baden-Württemberg, and to commercial customers.

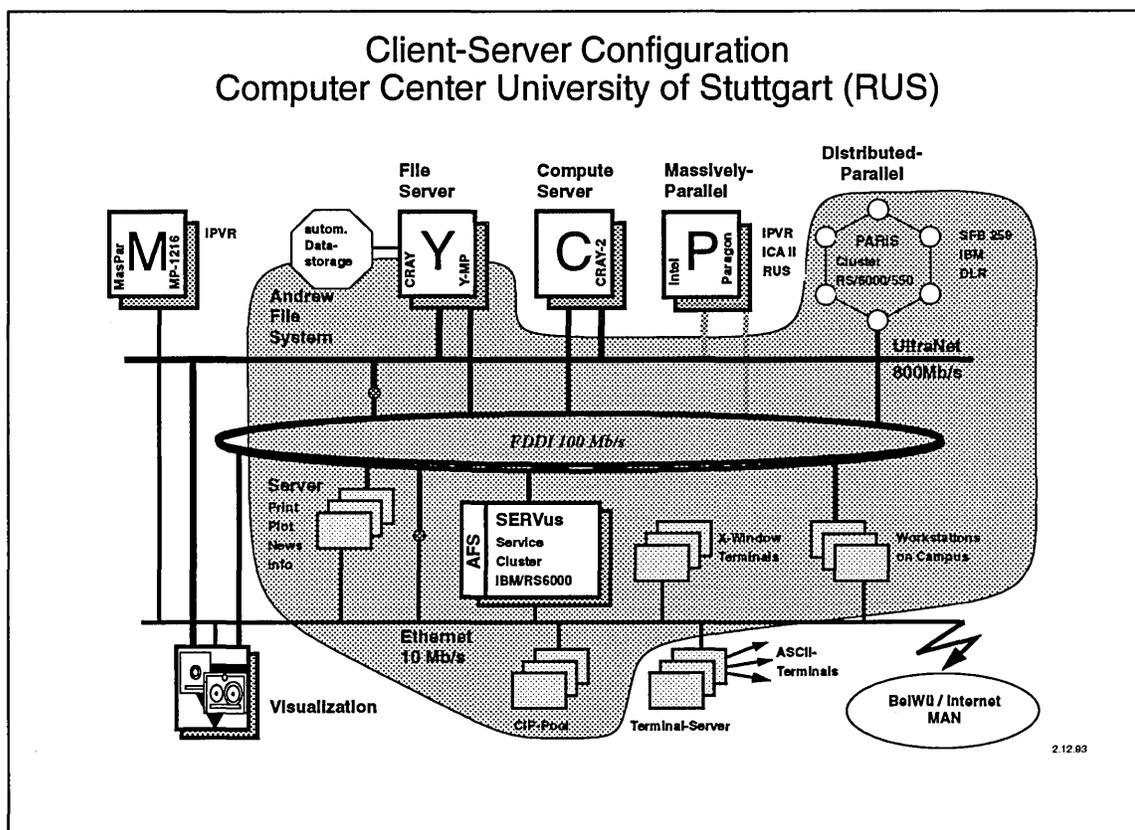


Fig. 1: Central Systems

Figure 1 outlines the central services offered to the user community. The Cray 2, which serves the need for supercomputer power, will be replaced by a C94 shortly. General services and scalar

batch are now being served by the SERVus cluster, a cluster of RISC servers(IBM RS/6000), as shown in figure 2. This cluster has successfully replaced a mainframe based timesharing system at the beginning of last year.

There are about 2800 registered users of the central computers. The LAN of the University of Stuttgart comprises 3200 computer nodes, including more than 1600 workstations. As the users and owners of these local systems become aware of the work involved in their administration and maintenance, they start to demand new central services to take this work off their shoulders.

The RUS-DCE project.

In response to these needs, in summer 1991, with some support from IBM, we started a project to investigate the technologies of distributed computing. At this time OSF-DCE was but an interesting blue print, so we decided to look into and evaluate the available technologies, and to make them available to the members of the project team. The ultimate goal of course was and still is to offer these as new services to the users.

The main parts of the project were:

- Time-Services - xntpd
- Kerberos
- AFS - Andrew File System
- NQS - Network Queuing System
- Distributed Software Maintenance and Software Distribution

The time service is just a part of the infrastructure of distributed computing, without any directly visible impact on the users.

The AFS cell rus.uni-stuttgart.de was installed in November 1991 as the first cell in Germany. AFS is one of the key components of the SERVus cluster in order to achieve a 'single system image'. Today AFS is in full production use at the university, and it is hard to imagine, how we could get our work done without it. AFS is the mature and widely used production Distributed Computing Environment available today.

But AFS lacks a finer granularity of administrative rights, as would be desirable at a university in order to keep departments under a 'common roof', but give them as much autonomy as possible

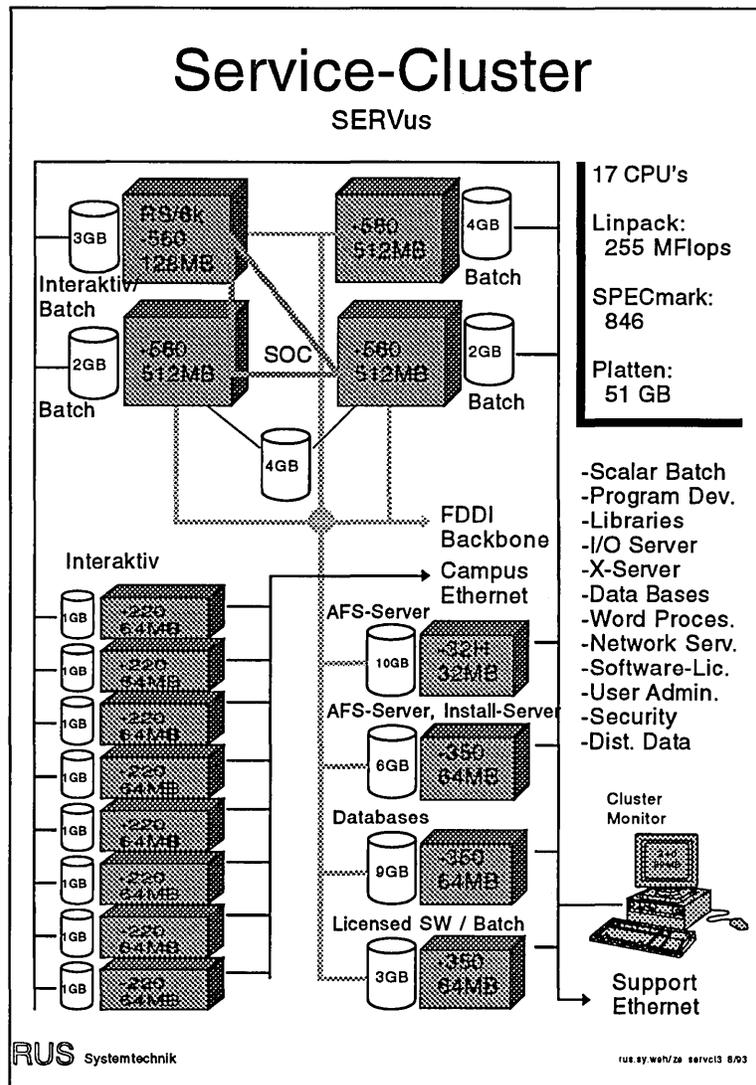


Fig. 2: SERVus RISC Cluster

without sacrificing security. This is the reason for having multiple AFS cells (currently 4) on the campus, as departments have to run their own cell, if they want to run and administer their own file servers.

OSF-DCE and DFS at Stuttgart.

While the main component technologies of OSF-DCE are in production use at RUS, it was obviously desirable to get hold of DCE at an early stage. We had the opportunity to take part in IBM's EPP (beta test) for AIX-DCE. We first configured our DCE test cell in November 1992, and we have DFS running since summer 1993. One does not expect everything to run absolutely smooth in a beta test, but one hopes to get ones hands onto a new product at an early stage. So we started off with one server machine and a handfull of clients, the working machines of participating project members.

Today we have three server machines with the following roles:

rusdce: CDS, DTS, DFS simple file server

zerberus: Security, CDS, DTS

sysrv2: DTS, DFS file server, FLDB server, system control machine

The clients are essentially the same, but besides the RS/6000's there now is a Sun Sparc machine.

There are 30 user accounts in the registry, and these people have their own DFS filesets, containing their home directories. We plan to install the DFS/AFS translator as soon as we can get hold of it.

Experiences with DCE and DFS.

The first impression with DCE is that this is a really monolithic piece of software. Nevertheless configuring server and client machines is relatively straight forward with the supplied shell scripts and especially with the SMIT interface supplied by IBM.

Next one realizes that the security is much more complex to administer, but on the other hand this offers one the finer granularity, which AFS is lacking. The cell_admin account of course still is master of the universe, but there are numerous ways to delegate well defined administrative tasks. This is mostly due to fact that one can put ACLs on almost everything from files to directories in the security space.

The Cell Directory Service (CDS) is intended to locate resources transparently. From this it is clear, that it is one of the key components of DCE. If the CDS is not working properly, all the other services may no longer be found and contacted, and everything will eventually grind to a halt. Its performance should be improved, before one can really use DCE in large production environments.

DFS essentially still is AFS with different command names. From the users perspective it is as transparent as AFS or, may be, even more transparent due to the synchronisation between ACL's and mode bits. And the administration is still the same. As an example: I just replaced the command names etc. in an old AFS administrative shell script of mine, and it just ran on DFS. And DFS appears to be quite stable and robust, but we have not yet really stressed it.

Nevertheless it is a good idea to copy ones files from DFS back to AFS, just to be on the safe side. This has proven to be a wise measure, not because of DFS failure but due to CDS problems, which can prevent one from getting at ones data. Mounting a DFS fileset locally on the file server can also help in case of problems.

Another problem worth mentioning was encountered, when we tried to move the security server

to a new machine. We succeeded in the end, but only by having the old and the new security server running simultaneously for takeover. Had the old security server been broken, we would not have been able to bring up a new security server from a database backup, and would have had to reconfigure the cell.

This brings us to one very important remark: as all objects in DCE are internally not represented by their names, but by their UUID, it is crucial to have the cells UUID written down somewhere. It is possible to configure a cell with a given UUID. Reconfiguring a cell with a new UUID makes all data in DFS inaccessible, until one forcibly changes the ACLs by hand.

What is missing in DCE.

There are a some features missing in DCE, which are absolutely necessary.

The most important of these is a Login program, which allows a user to log into DCE and the local machine in one step. This is the only way to be able to have ones home directory in DFS, and thus have the very same home directory on every machine one works on. AFS has this indispensable feature, without which a single system image on a workstation cluster may not be achieved.

We are currently using a 'single login' program developed by IBM banking solutions, which has some very interesting features for limiting access to machines to specified users.

DCE versions of utilities like ftp and telnet (or rlogin) are needed. As long as there are machines which are not DCE/DFS clients, there will be a need for shipping files the conventional way. And there will always be a need for running programs on the computer most suited for the problem at hand. One of the goals of distributed computing is for the user to see the same data on whichever computer he deems appropriate for solving his problems, and to allow him access to these in a secure manner, i.e. no passwords on the network, etc.

Batch processing in a distributed environment.

The other field which appears to have been forgotten by those developing DCE and DME, is batch processing. Despite the personal computing revolution most computing is still done in batch. Users use their desktop systems for program development etc., but the long production runs are usually beeing done on more powerful systems in batch.

Batch in the context of DCE faces two problems:

1. credential lifetime
2. distributed queue management

The lifetime problem can be overcome by renewable tickets, and by a special 'batch ticket granting service', which hands a new ticket for the batch user to the batch system in return for a ticket granted for the 'batch service' (possibly valid only for this jobs UUID), ignoring this tickets lifetime. This is, at least in the given framework, the only way to overcome the problem of a job being initiated after the ticket of the user, who submitted it, has already expired. This means accepting an expired ticket, but if, hours later, I do no longer believe, that this user submitted the job, I should better not run it anyway. Batch processing means doing tasks on behalf of the user in his absence, this is the very nature of it.

The second problem, namely distributed queue management, is at the heart of all currently available distributed batch systems. Their problem is: how to get the job input to the batch machine, and how to get the output back to the user. In 1976 we ran two Control Data mainframes with shared disks and, due to special software developed at RUS, shared queues. In its very essence this was a distributed file system, if only shared between two machines. The

existence of a distributed file system makes the problem of sending input and output obsolete. The data is there, at least logically, no matter from where you look at it. Queues then are essentially special directories in the distributed file system, with proper ACLs to regulate who is allowed to submit jobs to which queues. On the other side the batch initiating service on a batch machine knows in which queues to look for work, like on a stand alone system. And it anyway is best suited to have each machine decide when to schedule a new job. The only thing necessary might be a locking mechanism to prevent two machines from starting the same job at the same time. A centralized scheduler, which decides which batch machine should run which jobs, and then sends them there, is no longer needed, thus eliminating an other single point of failure.

A model for a campuswide DCE cell.

The envisaged goal of a campuswide DCE cell is to provide a transparent single system image to the users of computing equipment on the whole university campus. Of course the Computer Center cannot force departments into becoming part of this cell, but if we offer and provide them with a good service without too much administrative hassle, they will accept our offer. This is at least our experience from offering the AFS service to the campus.

There are two requirements for campuswide distributed computing: The user should be registered in only one place, with one user name, one password, one home directory, but be able to get at all available resources, for which he is authorised. And departments should be able register there own users and run their own servers, without opening their services to uncontrolled access.

The requirements on user management can easily be achieved in DCE due to the fact that the object space of the security service is not flat, but a true tree structure. Principals can be grouped in directories in the security space, and ACLs on these directories allow for the selective delegation of the rights to create or remove principals from a specific directory. Hence, by creating a separate directory for each university department under the principal and groups branches of the security space, it is possible to have department administrators do their own user administration, and still have a common campuswide security service.

In the same fashion it is possible to have departments mount their DFS filesets in a common file tree. By using DFS administrative domains, departments wanting to run their own file servers can safely do so, and still be part of the same cell. They may thus use centrally maintained software, share data with users in other departments, even other universities, and do so in a secure manner. And they may nevertheless have their filesets backed up centrally. This is a definite improvement of DFS over AFS, where the only way to securely run departmental file servers was to have multiple cells.

One of the principles of DCE, to which one should stick under all circumstances, is that authorisation should be regulated by ACLs. A DCE single login, whether invoked directly or via a DCE telnet, should thus grant access only to principals listed in an ACL for the machines interactive service. The hooks for this may be found by defining appropriate attributes for the corresponding objects in the CDS name space.

Conclusion.

To summarize our experiences thus far: DCE/DFS is a good secure system with great potential, but it is not yet mature and stable enough to be used in a true production environment. But we can use it today to learn about all its new and rich features, especially its powerfull security mechanisms. When it will be more mature, and supported by more vendors, we will be ready to use it as a truely distributed environment for scientific computing at the university.

ATM — Current Status

March 14, 1994, San Diego California
Cray User's Group Meeting

By: Benjamin R. Peek

Peek & Associates, Inc.
5765 SW 161st Avenue
Beaverton, OR 97007-4019
(503) 642-2727 voice
(503) 642-0961 fax

ASYNCHRONOUS TRANSFER MODE (ATM)

Overview

Today's Local Area Networks (LANs) do not support emerging high-bandwidth applications such as visualization, image processing, and multimedia communications. Asynchronous Transfer Mode (ATM) techniques brings computational visualization, video and networked multimedia to the desktop. With ATM, *each* user is guaranteed a *nonshared* bandwidth of 622M bps or 155M bps.

Conceived originally for Wide Area Network (WAN) carrier applications, ATM's capability to multiplex information from different traffic types, to support different speeds, and to provide different classes of service enables it to serve both LAN and WAN requirements. A detailed specification for developing very high-speed LANs using ATM technology was published in 1992. The UNI Specification is now at V.3.0, as of last fall. It describes the interface between a workstation and an ATM hub/switch and the interface between such a private device and the public network. Signaling is being developed to enable multipoint-to-multipoint conferencing.

Local ATM products are available now, and many more are expected by mid- to late-1994. In fact, 1994 is the year that ATM comes of age. UNI V.3.0 will allow the technology to be deployed into large scale production environments.

Bandwidth-intensive applications including multimedia, desk-to-desk videoconferencing, computational visualization, and image processing are now appearing as business applications. Existing 1M to 16M bps LANs (Ethernet and token-ring), and 100M bps LANs (Fiber Distributed Data Interface — FDDI, FDDI II), can marginally support these applications and their expected growth. Peek & Associates, Inc. forecasts indicate that there could be 1,000,000 multimedia PCs in business, manufacturing, education, and other industries by 1995.

Peek & Associates, Inc. estimates that the total market for ATM-based equipment and services will grow from approximately \$50 million in 1992 to more than a \$1.3 billion market by mid- to late-1995.

The shortcomings of these shared-medium LANs are related to the limited effective bandwidth per user, and to the communication delay incurred by users. A more serious problem involves the potential delay variation. For example, a 10M bps LAN may have an effective throughput of only 2M to 4M bps. Sharing that bandwidth among 10 users would provide a sustained throughput of only 200K to 400K bps per user, which, even if there was no delay variation, is not adequate to support quality video or networked multimedia applications.

Originally developed for Wide Area Network (WAN) carrier applications, ATM's capability to effectively multiplex signals carrying different network traffic and support different speeds makes it ideal for local (LAN) and for remote (WAN) applications. The term private ATM has been used to describe the use of ATM principles to support LAN applications.

Progress is being made on two fronts:

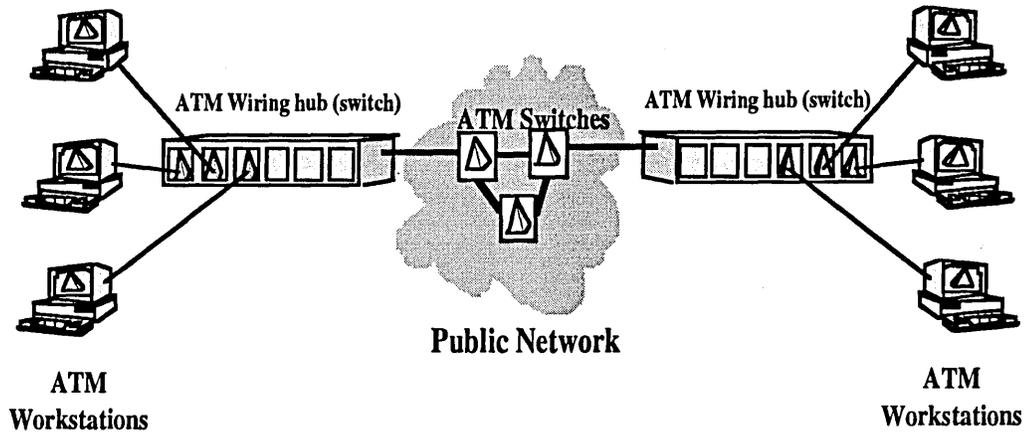
1. Within the Exchange Carriers Standards Association (ECSA) T1 Committee, ANSI, and Consultative Committee on International Telephony and Telegraphy (CCITT), for WAN and carrier applications; and
2. Within the ATM Forum for LAN applications. In general, good harmonization exists between the CCITT/ECSA work and the ATM Forum's work. If the ATM Forum is an example, vendors are very interested ATM. There are now (3/94) 485 member organizations internationally cooperating together in the ATM Forum.

Current Status

ATM and related standards have been developed in the past eight years under the auspices of Broadband Integrated Services Digital Network (BISDN), as the blueprint for carriers' broadband networks for the mid-1990s and beyond. BISDN is positioned as the technology for the new fast packet WAN services, such as cell relay and frame relay. Vendors have selected ATM-based systems for three key reasons:

- Local ATM enables major synergies between the LAN and the WAN, since both networks rely on the same transport technology. This allows a LAN to be extended through the public switched network, transparently. With seamless transparency, the LAN extends to the WAN, then to the GAN. This idea has the power to integrate communication protocols globally establishing a ubiquitous marketplace of private virtual networks (PVNs).
- ATM and related standards are nearly complete. ATM has already had an eight-year life. By 1993, the carrier-driven standards were fully published, and through the activities of the ATM Forum, a full complement of Local ATM specs was published in the fall of 1993. The ATM standard itself was stable as of December 1990, but support standards such as signaling (to enable call control, such as setup and teardown), are still being developed though much progress was made in 1993. The other (non-ATM) standards are only beginning to be developed, and it might take several years for them to reach maturity, particularly FFOL.
- Local ATM allows the delivery of 155M bps signals to the desktop over twisted-pair cable. FDDI has only recently made progress in that arena, and FCS relies on fiber of coax cables. Several documented technical studies have shown the viability of 155M bps on twisted pair, without exceeding the FCC's radiation limits. Chipsets exist today (3/94), priced at \$20 each in quantities of 1000, that implement ATM 155M bps over Class V copper infrastructure.

ATM LAN/WAN PVN



The ATM Forum provides the focal point for this activity. The Forum is a consortium of equipment vendors and other providers with a mandate to work with official ATM standards groups, such as ECSCA and CCITT, to ensure interoperability among ATM-based equipment. Vendors supporting the development include manufacturers of workstations, routers, switches, and companies in the local loop. The Forum was formed in late 1991, and membership has grown to the current level of 485 organizations in the short intervening time. Based on these activities, a number of proponents claim that ATM will become a dominant LAN standard in the near future.¹

BISDN standards address the following key aspects:

- User-Network Interface (UNI)
- Network-Node Interface (NNI)
- ATM
- ATM Adaptation Layer (AAL), to support interworking ("convergence" with other systems (such as circuit emulation)
- Signaling, particularly for point-to-multipoint and multipoint-to-multipoint (conferencing) calls
- End-to-end Quality of Service (QoS)
- Operations and maintenance

Connections can either be supported in a Permanent Virtual Circuit (PVC) mode or in a Switched Virtual Circuit (SVC) mode. In the former case, signaling is not required, and the user has a permanently assigned path between the sending point and the receiving point. In the latter case, the path is activated only for the call's duration; signaling is required to support SVC service.

ATM-based product vendors realize that the market will go flat if the inconsistencies characteristic of early ISDN products resurface in the broadband market. The first

¹ M. Fahey, "ATM Gains Momentum," *Lightwave*, September 1992, pp. 1 ff.

"implementers' agreement" to emerge from the ATM Forum was the May 1992 UNI Specification.¹ The interface is a crucial first goal for designing ATM-based equipment that can interoperate with equipment from other developers. The development of a 622M bps UNI is also important for applications coming later in the decade.

A related start-up consortium is the SONET-ATM User Network (Saturn). Saturn's mandate was to create an ATM UNI chipset. The goal was achieved during 1993 and there are now several chipsets from which to select.

The ATM Forum's UNI Specification

ATM is a telecommunications concept defined by ANSI and CCITT standards for carrying voice, data, and video signals, on any UNI. On the basis of its numerous strengths, ATM has been chosen by standards committees (such as ANSI T1 and CCITT SG XVIII) as an underlying transport technology for BISDN. "Transport" refers to the use of ATM switching and multiplexing techniques at the data link layer (OSI Layer) to convey end-user traffic from a source to a destination.

The ATM technology can be used to aggregate user traffic from existing applications onto a single access line/UNI (such as PBX trunks, host-to-host private lines, or video-conference circuits) and to facilitate multimedia networking between high-speed devices (such as supercomputers, workstations, servers, routers, or bridges) at speeds of 155M to 622 M bps range. An ATM user device, to which the specification addresses itself, may be either of the following:

- An IP router that encapsulates data into ATM cells, and then forwards the cells across an ATM UNI to a switch (either privately owned or within a public network).
- A private network ATM switch, which uses a public network ATM service for transferring ATM cells (between public network UNIs) to connect to other ATM user devices.

The initial Local ATM Specification covers the following interfaces:

1. Public UNI — which may typically be used to interconnect an ATM user with an ATM switch deployed in a public service provider's network; and
2. Private UNI — which may typically be used to interconnect an ATM user with an ATM switch that is managed as part of the same corporate network.

The major distinction between these two types of UNI is physical reach. Both UNIs share an ATM layer specification but may utilize different physical media. Facilities that connect users to switches in public central offices must be capable of spanning distances up to 18,000 feet. In contrast, private switching equipment can often be located in the same room as the user device or nearby (such as within 100 meters) and hence can be limited-distance transmission technologies.

¹ ATM Forum, UNI Specification, May 1992.

ATM Bearer Service Overview

Carrying user information within ATM format cells is defined in standards as the ATM bearer service involves specifying both an ATM protocol layer (Layer 2) and a compatible physical media (Layer 1).

The ATM bearer service provides a connection-oriented, sequence-preserving cell transfer service between source and destination with a specified QoS and throughput. The ATM physical (PHY) layers are service independent and support capabilities applicable to (possibly) different layers residing immediately above them. Adaptation layers, residing above the ATM layer, have been defined in standards to adapt the ATM bearer service to provide several classes of service, particularly Constant Bit-Rate (CBR) and Variable Bit-Rate (VBR) service.

An ATM bearer service at a public UNI is defined to offer point-to-point, bi-directional virtual connections at either a virtual path (VP) level and/or a virtual channel (VC) level; networks can provide either a VP or VC (or combined VP and VC) level services. For ATM users desiring only a VP service from the network, the user can allocate individual VCs within the VP connection (VPC) as long as none of the VCs are required to have a higher QoS than the VP connection. A VPC's QoS is determined at subscription time and is selected to accommodate the tightest QoS of any VC to be carried within that VPC. For VC level service at the UNI, the QoS and throughput are configured for each virtual channel connection (VCC) individually. These permanent virtual connections are established or released on a subscription basis.

ATM will initially support three categories of virtual connections:

1. Specified QoS Deterministic
2. Specified QoS Statistical
3. Unspecified QoS

The two specified categories differ in the Quality of Service provided to the user. Specified Deterministic QoS connections are characterized by stringent QoS requirements in terms of delay, cell delay variation, and loss and are subject to Usage Parameter Control (UPC) of the peak rate at the ATM UNI. Specified Deterministic QoS connections could be used to support CBR service (T1/T3 circuit emulation) or VBR services with minimal loss and delay requirements. Specified Statistical QoS connections are characterized by less stringent QoS requirements and are subject to Usage Parameter Control of the peak rate at the ATM UNI. Typically, Specified Statistical QoS connections would be used to support variable bit rate services (data) that are tolerant to higher levels of network transit delay compared to applications such as voice, which may require CBR.

When ATM equipment evolves to require dynamically established ("switched") virtual connections, the ATM bearer service definition must be augmented to specify the ATM Adaptation Layer protocol (in the Control Plane — C-plane), required for User-Network signaling.

The ATM bearer service attributes to be supported by network equipment conforming to the Local ATM UNI specification are summarized in the following table.

Summary of Required ATM Functionality

<u>ATM Bearer Service Attribute</u>	<u>Private UNI</u>	<u>Public UNI</u>
Support for Point-to-Point VPCs	Optional	Required
Support for Point-to-Point VCCs	Required	Required
Support for Point-to-Multipoint VPCs	Optional	Optional
Support for Point-to-Multipoint VCCs/SVC	Optional	Optional
Support for Point-to-Multipoint VCCs/PVC	Optional	Optional
Support for Permanent Virtual Connections	Required	Required
Support for Switched Virtual Connections	Required	Required
Support for Specified QoS Classes	Optional	Required
Support of an unspecified QoS Class	Optional	Required
Multiple Bandwidth Granularities for ATM Connections	Optional	Required
Peak Rate Traffic Enforcement via UPC	Optional	Required
Traffic Shaping	Optional	Optional
ATM Layer Fault Management	Optional	Required
Interim Local Management Interface	Required	Required

Physical Layer Specification

The private UNI connects customer premises equipment, such as computers, bridges, routers, and workstations, to a port on an ATM switch and/or ATM hub. Local ATM specifies physical layer ATM interfaces for the public and private UNI. Currently, a 44.736M bps, a 100M bps, and two 155.52M bps interfaces are specified. Physical layer functions in the "User Plane" are grouped into the Physical Media Dependent (PMD) sublayer and the Transmission Convergence (TC) sublayer. The PMD sublayer deals with aspects that depend on the transmission medium selected. The PMD sublayer specifies physical medium and transmission characteristics (such as bit timing or line coding) and does not include framing or overhead information. The transmission convergence sublayer deals with physical layer aspects that are independent of the transmission medium characteristics.

SONET Physical Layer Interface

The physical transmission system for both the public and private User-Network Interface is based on the Synchronous Optical Network (SONET) standards. Through a framing structure, SONET provides the payload envelope necessary for the transport of ATM cells. The channel operates at 155.52M bps and conforms to the Synchronous Transport Signal Level 3 Concatenated (STS-3C) frame. The UNI's physical characteristics must comply with the SONET PMD criteria specified in ECSA T1E1.2/92-020. Given that SONET is an international standard, it is expected that SONET hierarchy-based interfaces will be a means for securing interoperability in the long term for both the public and private UNI. For various availability and/or economic reasons, however, other physical layers have been specified to accelerate the deployment of interoperable ATM equipment.

The following table depicts the Physical Layer functions which need to be supported by ATM-based equipment, such as private switches and hubs.

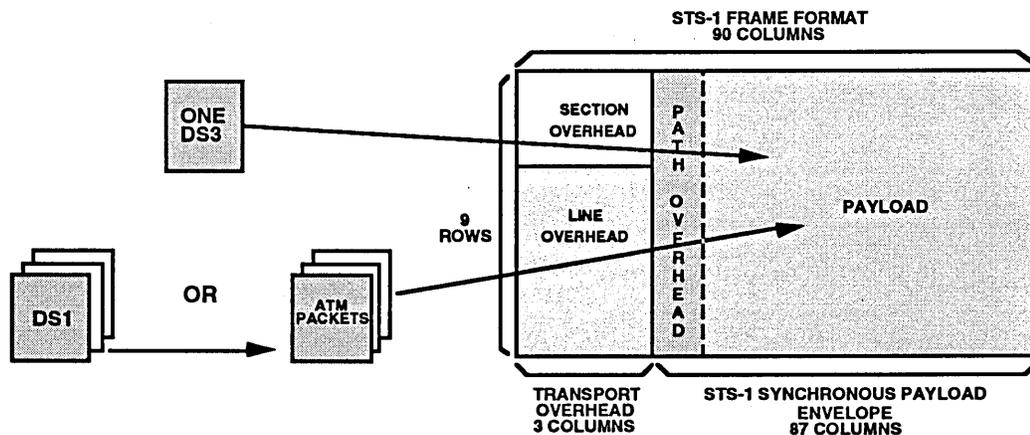
Physical Layer Functions Required for SONET-Based ATM Equipment

Transmission Convergence Sublayer	Header Error Control generation/verification Cell scrambling/descrambling Cell delineation Path signal identification Frequency justification and pointer processing Multiplexing Scrambling/descrambling Transmission frame generation/recovery
Physical Media Dependent Sublayer	Bit timing Line coding Physical interface

Most of the functions comprising the TC sublayer are involved with generating and processing overhead bytes contained in the SONET STS-3c frame. The "B-ISDN independent" TC sublayer functions and procedures involved at the UNI are defined in the relevant sections of ANSI T1.105-1991, ECSA T1E1.2/92-020, and ECSA T1Sa/92-185. The "B-ISDN specific" TC sublayer contains functions necessary to adapt the service offered by the SONET physical layer to the service required by the ATM layer (these are the top four functions depicted in the previous table under the TC sublayer).

SONET Frame Format

Payload Options



Physical Layer Functions Required to Support DS3-Based Local ATM Equipment

Transmission Convergence Sublayer	Header Error Control generation/verification Physical layer convergence protocol framing Cell delineation Path overhead utilization Physical layer convergence protocol timing Nibble stuffing
Physical Media Dependent Sublayer	Bit timing Line coding Physical interface

The 44.736M bps interface format at the physical layer is based on asynchronous DS3 using the C-Bit parity application (CCITT G.703, ANSI T1.101, ANSI T1.107, ANSI T1.107a, and Bellcore TR-TSY-000499). Using the C-Bit parity application is the default mode of operation. If equipment supporting C-Bit parity interface with equipment that does not support C-Bit parity, however, then the equipment supporting C-Bit parity must be capable of "dropping back" into a clear channel mode of operation.

To carry ATM traffic over existing DS3, 44.736M bps communication facilities, a Physical Layer Convergence Protocol (PLCP) for DSC is defined. This PLCP is a subset of the protocol defined in IEEE P802.6 and Bellcore TR-TSV-000773. Mapping ATM cells into the DS3 is accomplished by inserting the 53-byte ATM cells into the DS3 PLCP.

Extracting ATM cells is done in the inverse manner; that is, by framing on the PLCP and then simply extracting the ATM cells directly.

Physical Layer for 100M bps Multimode Fiber Interface

The Local ATM specification also describes a physical layer for a 100M bps multimode fiber for the private UNI. The motivation is to re-utilize FDDI chipsets at the physical layer (with ATM at the Data Link Layer). The private UNI does not need the link distance and the operation and maintenance complexity provided by telecom lines; therefore, a simpler physical layer can be used, if desired. The Interim Local Management Interface (ILMI) specification provides the physical layer Operations and Maintenance functions performed over the local fiber link. As with the previous case, the physical layer (U-plane) functions are grouped into the physical media dependent sublayer and the transmission convergence sublayer. The network interface unit (NIU), in conjunction with user equipment, provides frame segmentation and reassembly functions and includes the local fiber link interface.

The ATM Forum document specifies the rate, format, and function of the 100M bps fiber interface. The fiber interface is based on the FDDI physical layer. The *bit rate* used refers to the logical information rate, before line coding; the term *line rate* is used when referring to the rate after line coding (such as a 100M bps bit rate resulting in a 125M

baud line rate if using 4B/5B coding). This physical layer carries 53-byte ATM cells with no physical layer framing structure.

This physical layer follows the FDDI PMD specification. The link uses 62.5-micrometer multimode fiber at 100M bps (125M baud line rate). The optical transmitter and fiber bandwidth adheres to the specification ISO DIS 9314-3. The fiber connector is the MIC duplex connector specified for FDDI, allowing single-connector attachment and keying if desired.

The fiber link encoding scheme is based on the ANSI X3T9.5 (FDDI) committee 4 Bit/5 Bit (4B/5B) code. An ANSI X3T9.5 system uses an 8-bit parallel data pattern. This pattern is divided into two 4-bit nibbles, which are each encoded into a 5-bit symbol. Of the 32 patterns possible with these five bits, 16 are chosen to represent the 16 input data patterns. Some of the others are used as command symbols. Control codes are formed with various combinations of FDDI control symbol pairs.

Physical Layer for 155M bps Multimode Fiber Interface Using 8B/10B

The ATM specification also supports an 8B/10B physical layer based on the Fiber Channel Standard. This PMD provides the digital baseband point-to-point communication between stations and switches in the ATM LAN. The specification supports a 155M bps (194.4M baud), 1300-nm. multimode fiber (private) UNI.

The PMD provides all the services required to transport a suitably coded digital bit stream across the link segment. It meets the topology and distance requirements of building and wiring standards EIA/TIA 568. A 62.5/125-micron, graded index, multimode fiber, with a minimum modal bandwidth of 500 MHz/km., is used as the communication link (alternatively, a 50-micron core fiber may be supported as the communication link). The interface can operate up to 2 km. maximum with the 62.5/125-micron fiber; the maximum link length may be shortened when 50-micron fiber is utilized. The non-encoded line frequency is 155.52M bps, which is identical to the SONET STS-3c rate. This rate is derived from the insertion of one physical layer for every 26 data cells (the resultant media transmission rate is 194.40M baud).

ATM Cell Structure and Encoding at the UNI

Equipment supporting the UNI encodes and transmits cells according to the structure and field encoding convention defined in T1S1.5/92-002R3.

ATM Cell Fields

Generic Flow Control (GFC): This field has local significance only and can be used to provide standardized local functions (such as flow control) on the customer site. The value encoded in the GFC is not carried end to end and can be overwritten in the public network. Two modes of operation have been defined for operation of the GFC field: uncontrolled access and controlled access. The uncontrolled access mode of operation is used in the early ATM environment. This mode has no impact on the traffic which a host generates.

Virtual Path/Virtual Channel Identifier (VPI/VCI): The actual number of routing bits in the VPI and VCI subfields used for routing is negotiated between the user and the network, such as on a subscription basis. This number is determined on the basis of the lower requirement of the user or the network. Routing bits within the VPI and VCI fields are allocated using the following rules:

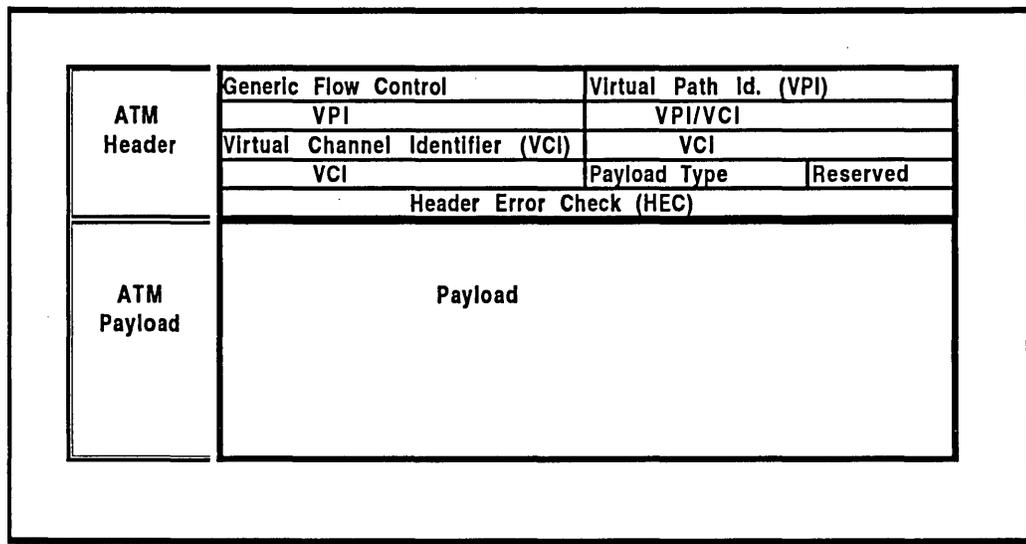
- The VPI subfield's allocated bits are contiguous
- The VPI subfield's allocated bits are its least significant, beginning at bit 5 of octet 2
- The VCI subfield's allocated bits are contiguous
- The VCI subfield's allocated bits are the least significant, beginning at bit 5 of octet 4

Any VPI subfield bits that are not allocated are set to 0.

Payload Type (PT): This is a 3-bit field used to indicate whether the cell contains user information or Connection Associated Layer Management information. It is also used to indicate a network congestion state or for network resource management.

Cell Loss Priority (CLP): This is a 1-bit field allowing the user or the network to optionally indicate the cell's explicit loss priority.

Header Error Control (HEC): The HEC field is used by the physical layer for detection/correction of bit errors in the cell header. It may also be used for cell delineation.



ATM Cell Structure

Late Breaking News

Federal legislation enabling communications companies to develop a national information highway made its first step through Congress March 2, 1994. The House telecommunications subcommittee unanimously backed a bill that would overhaul the

nation's 1934 Communications Act now that industry alliances and technological advances are blurring boundaries between telephone and cable industries, Joanne Kelley of Reuters news service reported.

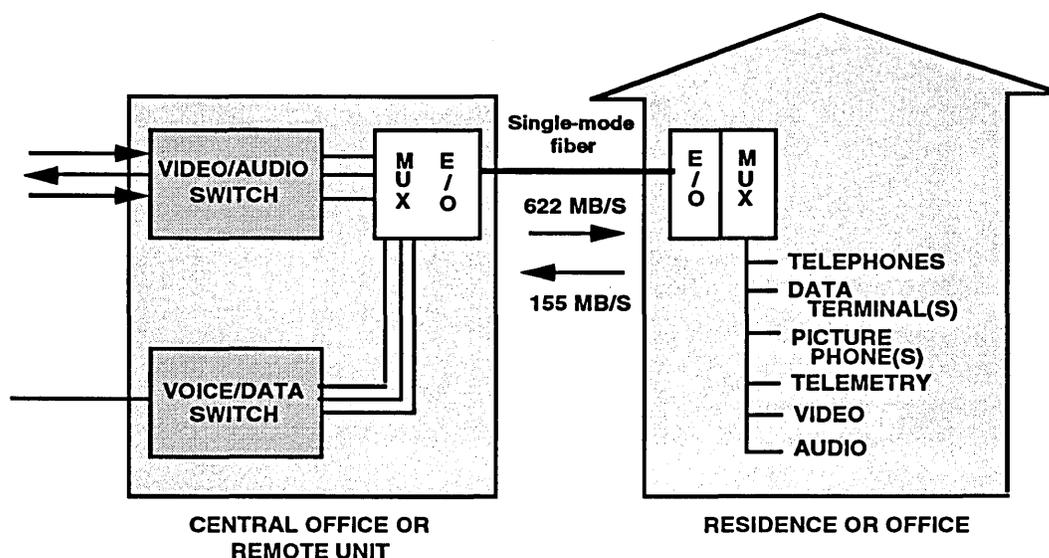
The information superhighway, championed by Vice President Al Gore, could link the nation's homes, businesses and public facilities on high-speed networks to allow them to quickly send and receive information. "This is the most significant change in communications policy in 60 years," said Massachusetts Democrat Edward Markey, who heads the House panel. "This legislation will pave the way for the much-anticipated information superhighway."

The bill, which next faces a vote by the Energy and Commerce committee, would break down barriers that currently separate the phone and cable television industries, freeing them to invade each other's turf. Several revisions to the bill dealt with issues raised by various industry factions and consumer groups during lengthy hearings during the past several weeks, Kelley reported.

"Legislation to create a competitive telecommunications marketplace should and can be completed this year," said Decker Anstrom, president of the National Cable Television Association. Lawmakers left some key issues unresolved in hopes of maintaining the tight schedule they set to ensure passage of the legislation this year. Two lawmakers were persuaded to withdraw controversial amendments until the full committee takes up the legislation.

The deferred measures include a provision that would allow broadcasters to use existing spectrum licenses to provide new wireless communications services. That raised fears among some lawmakers that this spring's first ever auctions of spectrum for such services might command lower bids by rivals. The lawmakers also supported a measure pending before another House judiciary panel that would relax restrictions on regional telephone companies, which are currently barred from entering long distance and equipment making businesses.

Broadband Network Future - 1996



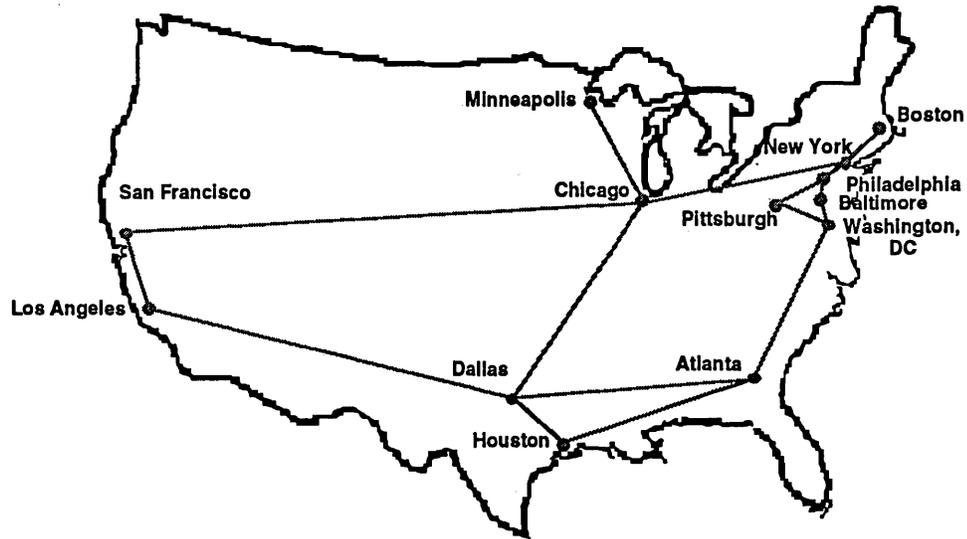
ATM Nationally and Internationally

MFS Datanet, Inc., an operating company of MFS Communications Company, Inc., launched the first end-to-end international ATM service in February 1994, making it possible to send data globally. Availability of MFS Datanet's ATM service solves the problem for multinational businesses of how to address increasing requirements for speed and reliability. More than half of all communications by international companies today are data, rather than voice.

MFS Datanet -- which launched the first national ATM service on Aug. 4, 1993 -- will initially offer the service between the U.S. and the United Kingdom, with expansion planned in Western Europe and the Pacific Rim.¹

MFS Datanet's High-speed Local Area Network Interconnect services (HLI), a range of services allowing personal computer networks to intercommunicate, will now be offered on an international basis. "MFS Datanet's global expansion will mean customers around the world will have access to a flexible, economical, high-speed medium to accommodate emerging high-volume applications such as multimedia," said Al Fenn, president of MFS Datanet.

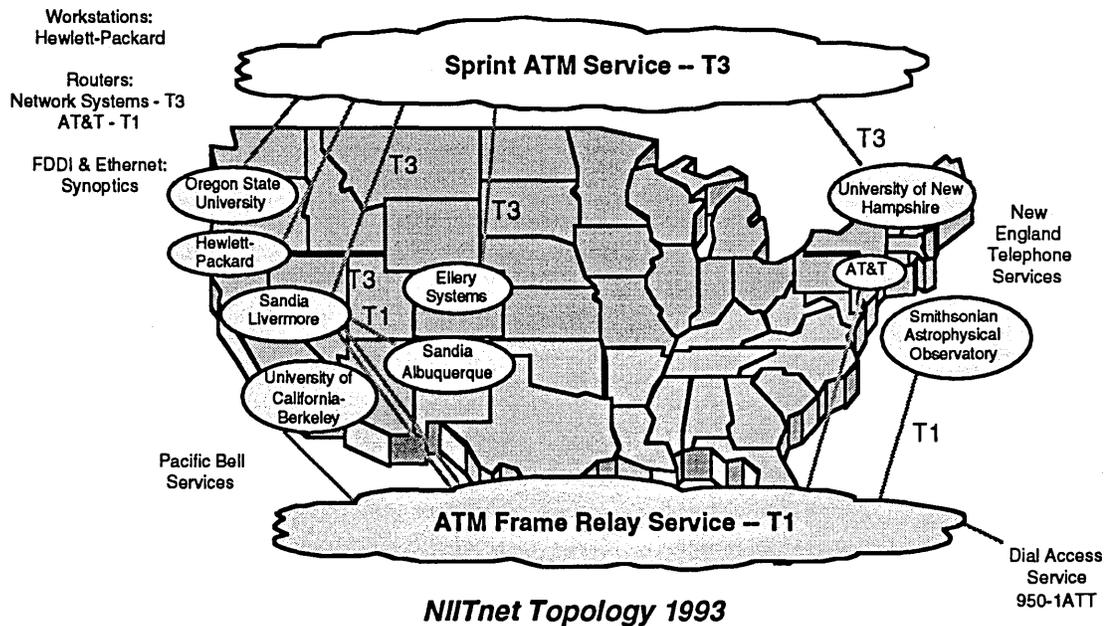
¹ Lightwave, September 1993.



The international network will be managed and controlled by the Network Control Center in MFS Datanet's headquarters office in San Jose, Calif., with parallel control for the U.K. operation in the London offices of MFS Communications Limited. In the U.S., the ATM service is offered in 24 metropolitan areas.

Sprint ATM Service

SCinet '93 was the prototype for the NIITnet and all of the facilities that were prototyped are currently a part of the NIIT. This briefing was delivered at this conference in a different session. Additional detail is available in the conference proceedings.



ATM Links Supercomputing and Multimedia in Europe

A cluster using ATM (Asynchronous Transfer Mode) technology has been put into operational use at Tampere University of Technology, Finland.¹ "We have tested ATM technology with real applications," said project manager Mika Uusitalo. "The results show the good feasibility and scalability of ATM. It really seems to offer a new way to distribute and use supercomputing facilities."

Used together, high-speed ATM networks and clustering give a platform for the new generation of scientific applications in which visual information plays the key role, Uusitalo said. He added that ATM is a solution for distributed audio-visual communication applications requiring high network bandwidth and also low predictable delays, satisfying the requirements by transferring data extremely fast in small, fixed-length packets called cells. ATM also has properties that guarantee real time transmission of data, video, and audio signals.

Tampere University of Technology's cluster, connected directly to an ATM network, can be used for traditional supercomputing and running distributed multimedia applications. Currently, the ATM network at the university operates at 100 Mb/sec., and the cluster consists of eight Digital Equipment (DEC) Alpha AXP machines. Users noted that the ATM cluster has much better price/performance than traditional supercomputers.

Among the first applications run at Tampere University of Technology was the computation and visualization of 3D magnetic and electrical fields, an application typical of today's needs: computationally very demanding and generating a lot of data for visualization.

"Earlier, the visualization of simulations included two phases," said Jukka Helin, senior research scientist. "First, the results of the simulation were generated on a supercomputer and later post processed and visualized by a workstation. Now the user in the ATM network sees how the solution evolves in real time." For further information, contact Jukka Helin at helin@cc.tut.fi or Mika Uusitalo at uusitalo@cc.tut.fi.

¹ 2130 ATM Links Supercomputing and Multimedia in Finland Oct. 4, 1993 HPCwire

APPENDIX

Background

A number of special interest groups address broadband services and technologies, serving the information needs of both end users and vendors. Four important groups are the ATM forum, Frame Relay Forum, the Fibre Channel Association and the SMDS Special Interest Group. The primary function of these groups is to educate, not to set standards. Their objective is to promote their respective technologies by expediting the development of formalized standards, encouraging equipment interoperability, actuating implementation agreements, disseminating information through published articles and information. These groups also provide knowledgeable speakers. These groups were seldom chartered with defining technology standards. However, some of these groups have become more active in the standards process. The ATM Forum has become most active in making specifications work, according to Fred Sammartino, President and Chairman of the Board of the ATM Forum.

Established Forums Sponsored by INTEROP

Three of these four separate forums, Asynchronous Transfer Mode (ATM), Switched Multi-megabit Data Service (SMDS), and frame relay, selected INTEROP Co. (Mountain View, CA) to perform as their secretary. INTEROP Co. was founded in 1985 as an educational organization to further the understanding of current and emerging networking standards and technologies. INTEROP sponsors the annual INTEROP Conference and Exhibition in the spring (Washington, DC) and in the fall (San Francisco, CA). Additionally, INTEROP publishes the technical journal *ConneXions*. The Fibre Channel Association is a separate organization and functions differently.

Interop's Director of Associations, Elaine Kearney, is responsible for the strategic development of each new forum. Each new forum must be formally incorporated and seek legal support so that antitrust laws are not violated. INTEROP handles membership databases, mailings, and trade show representation, and participates on each forum's Board of trustees as a nonvoting member.

The ATM, SMDS, and Frame Relay Forums, with the assistance of INTEROP Co., perform as conduits for distributing information relating to their representative technologies. Additionally, these forums promote cooperative efforts between industry members and technology users.

All of these groups are described in more detail in the Appendix. These descriptions include a location (address) to acquire more information, the structure of the management team (where available) and a short description of the charter and organization objectives available.

ATM Forum

ATM Forum

480 San Antonio Road

Suite 100

Mountain View, CA 94040

Phone: (415) 962-2585; Fax: (415) 941-0849

Annual Membership Dues: Full membership \$10,000 per company.

Auditing membership: \$1,500 (observers may attend general meetings free of charge).

Board of Directors

Chairman and President — Fred Sammartino, Sun Microsystems

Vice President, Business Development — Charlie Giancarlo, ADAPTIVE Corp., an N.E.T. Company

Vice President, Committee Management — Steve Walters, Bellcore

Vice President, Marketing & Treasurer — Irfan Ali, Northern Telecom

Vice President, Operations & Secretary — Dave McDysan, MCI Communications.

Executive Director — Elaine Kearney, INTEROP Co.

Charter

The ATM Forum's charter is to accelerate the use of ATM products and services through a rapid convergence and demonstration of interoperability specifications, and promote industry cooperation and awareness. It is not a standards body, but instead works in cooperation with standards bodies such as ANSI and CCITT.

Overview of the ATM Forum

The ATM Forum was announced in October 1991 at INTEROP 91 and currently represents 350+ member organizations. The ATM Forum's goal is to speed up the development and deployment of ATM products and services. Its initial focus was to complete the ATM User Network Interface (UNI), which was accomplished June 1, 1992. The UNI isolates the particulars of the physical transmission facilities from upper-layer applications. The ATM Forum Specification was based upon existing and draft ANSI, CCITT, and Internet standards.

The ATM forum has two principal committees: *Technical* and *Market Awareness and Education*. The Technical committee meets monthly to work on ATM specifications documents. Areas targeted include Signaling, Data Exchange Interface (DXI), Network-to-Network Interface, Congestion Control, Traffic Management, and additional physical media (such as twisted-pair cable). The ATM Market Awareness and Education committee's goal is to increase vendor and end-user understanding of, and promote the use of ATM technology.

In response to an increasing European interest in ATM technology and the ATM Forum, the Forum is expanding its activities outside of North America to the European

Community. It held a meeting in Paris in November 1992 to form its European activities. More meetings occurred in 1993.

Frame Relay Forum

The Frame Relay Forum
480 San Antonio Road
Suite 100
Mountain View, CA 94040
Phone: (415) 962-2579; Fax: (415) 941-0849
Annual Membership Dues: Full membership \$5,000 per company.
Joint membership for company affiliates: \$2,000.
Auditing level: \$1,000 per year, reserved for universities, consultants, users, and nonprofit organizations.

Board of Trustees and Officers

Chairman and President — Alal Taffel, Sprint
Vice President — Sylvie Ritzenthaler, OST
Treasurer — John Shaw, NYNEX
Secretary — John Valentine, Northern Telecom
Trustees — Richard Klapman, AT&T Data Comm. Services; Lawrence J. Mauceri, Hughes Network Systems; Holger Opderbeck, Netrix
Executive Director — Elaine Kearney, INTEROP Co.

Publications

The Frame Relay Forum News
UNI Implementation Agreement
NNI Implementation Agreement

Charter

The Frame Relay Forum is a group of frame relay service providers, equipment vendors, users, and other interested parties promoting frame relay acceptance and implementation based on national and international standards.

Overview of the Frame Relay Forum

The Frame Relay Forum was established in January 1991 and currently has 107 member organizations. It is a nonprofit corporation dedicated to promoting the acceptance and implementation of frame relay based on national and international standards. The Forum's activities include work on technical issues associated with implementation, promotion of interoperability and conformance guidelines, market development and education, and round tables on end-user experiences. Forum membership is open to service providers, equipment vendors, users, consultants, and other interested parties. By participating in the Forum, members can have an active role in developing this new technology.

The Frame Relay Forum is organized with a Board of Trustees and four committees. The committees are the following:

- Technical Committee
- Interoperability & Testing
- Market Development and Education
- Inter-Carrier Committee

In addition, the Forum maintains an International branch (FRF International), with European and Australian Chapters:

European Chapter

Paris, France

Contact: Sylvie Ritzenthaler

Phone: +33-99-32-50-06; Fax: +33-99-41-71-75

Australia/New Zealand Chapter

Allambie Heights, New South Wales

Contact: Linda Clarke

Phone: +612-975-2577; Fax: +612-452-5397

The Frame Relay Forum has organized User Roundtables so that users of frame relay can exchange experiences and ideas among each other. The Frame Relay Forum intends User Roundtables to serve as the basis for a wide array of other user activities.

The Frame Relay Forum Speakers Program

The Frame Relay Forum sponsors a speaker program free of charge. An expert on frame relay will be made available to visit a company's site and present a 30- to 45-minute presentation on frame relay. The presentation will introduce frame relay technology, products, and services. Contact the Frame Relay Forum for further details and scheduling.

SMDS Special Interest

The SMDS Interest Group

480 San Antonio Road

Suite 100

Mountain View, CA 94040-1219

Phone: (415) 962-2590; Fax: (415) 941-0849

Annual Membership Dues: \$4,999.

Annual Affiliate Membership: \$3,000.

Annual Associate Membership: \$800.

Annual Observer Membership: Free.

Board of Trustees & Officers

Chairman and President — Steve Starliper, Pacific Bell

Vice President — Scott Brigham, MCI

Treasurer — Robyn Aber, Bellcore
Executive Director — Anne M. Ferris, INTEROP Co.
Board Members: Tac Berry, Digital Link; Bill Buckley, Verilink; Joe DiPeppe, Bell Atlantic; Jack Lee, ADC Kentrox; Hanafy Meleis, Digital Equipment Corp.; Allen C. Millar, IBM; Connie Morton, AT&T Network Systems; David Yates, Wellfleet Communications

Publication

SMDS News (published quarterly)
Sharon Hume, Editor
3Com Corporation
5400 Bayfront Plaza
Santa Clara, CA 95052
Phone: (408) 764-5166; Fax: (408) 764-5002

Charter

The SMDS Interest Group is a group of SMDS providers, equipment vendors, users, and other interested parties working toward the proliferation and interoperability of SMDS products, applications, and services. The group's activities include advancing SMDS and providing SMDS education, fostering the exchange of information, ensuring worldwide interoperability, determining and resolving issues, identifying and stimulating applications, and ensuring the orderly evolution of SMDS.

Overview of the SMDS Interest Group

The SMDS Interest Group (SIG) was established in the fall of 1990 and currently has 58 member organizations. The SIG provides a central point for discussing and focusing on SMDS. Members who are SMDS users can influence products and services, gain exposure to a wide array of vendor offerings, and receive timely information for network planning. Likewise, SMDS vendors receive timely information for planning and developing products and services. Vendors also benefit from having the SIG's marketing efforts to supplement their own. Each SIG member can participate in resolving issues raised in SIG forums by exercising their membership voting privileges. Each SIG membership is allowed one vote. Besides having a vote, members can help shape the direction of SIG activities by participating in several working groups which the SIG sponsors. These working groups include the following:

- Technical Working Group
- Inter-Carrier Working Group
- PR and Market Awareness Working Group

Each of these working groups meet at the quarterly SIG meeting and at various other times. The meetings are open to all interested parties.

Members receive meeting minutes outlining the discussions and actions taken at these forums. Members are kept informed of industry developments via a monthly clipping service dedicated to tracking published news items on SMDS and competitive services

that appear in the trade press. A quarterly newsletter tracks the SMDS activities of SIG members. Members can use the newsletter for product announcements and profiles.

The SIG is planning several projects that will further SMDS market awareness. These activities include the development of a speakers package, an SMDS training course, and a compendium of SMDS applications.

Fibre Channel Association

Fibre Channel Association
12407 MoPac Expressway North 100-357
P.O. Box 9700, Austin, TX 78766- 9700
(800) 272-4618.

Initiation Fee: \$2,500
Annual Principal Membership Dues: \$5,000.
Annual Associate Membership: \$1,500.
Annual Documentation Fee: \$1,000.
Annual Observer Membership: \$400 per meeting
Annual Educational Membership: \$250.

Board of Trustees & Officers

President — Jeff Silva (needs to be verified).
Vice President — no information.
Treasurer — no information.
Executive Director — no information.
Board Members: Jeff Silva, no information.

Publication

No information.

Charter

Promote industry awareness, acceptance and advancement of Fibre Channel. Encourage article publication, educational seminars and conferences along with trade shows, round tables, and special events. Accelerate the use of Fibre Channel products and services. Distribute technical and market-related information and maintain a products and services database. Foster a Fibre Channel infrastructure that enables interoperability. Develop application-specific profiles and test specifications and environments.

Committees

- Marketing Committee
- Strategic Committee
- Technical Committee

Information About the Fibre Channel Association

Twenty organizations were originally involved in the formation of the Fibre Channel Association (FCA) in January 1993. The idea is to promote Fibre Channel technology as a high-performance interconnect standard for moving large amounts of information. Fibre Channel can transmit large data files bi-directionally at one gigabit per second.¹

"While computer processors have become increasingly faster and capable of handling larger amounts of data, the interconnects between these systems, and the input/output devices that feed them data, are unable to run at speeds necessary to take advantage of these more powerful products," said Jeff Silva, FCA board member.

"Fibre Channel offers an excellent solution for these data-intensive environments. At the same time, when links within and between companies, schools, hospitals, governments and other organizations are created, Fibre Channel will provide standards-based, high-speed on-ramps necessary for these digital information highways of the future."

Under development for more than four years, the Fibre Channel standard was initially endorsed by the Fibre Channel Systems Initiative (FCSI), a joint effort of Hewlett-Packard, IBM and Sun Microsystems Computer Corp. "One of the primary missions of the FCSI was to kick-start the development of profiles and use of Fibre Channel technology for the workstation arena," said Ed Frymoyer, FCSI's program manager. "The FCA's wide spectrum of industry players and their varied interests helps ensure the development of a broad array of Fibre Channel profiles for products aimed at a multitude of applications."

Dataquest, a San Jose-based market research firm, estimates that total revenue for the Fibre Channel adapter market alone will grow from under \$2 million in 1993 to \$1.2 billion in 1998. It estimates that by 1998, Fibre Channel adapter shipments will reach more than 500,000 units for workstations, PCs, midrange, mainframe and supercomputer systems. Similar trends are forecasted for other Fibre Channel products, like ports and switches.

INTEROP is not the key sponsor of the Fibre Channel Association. For this reason, it looks and acts a bit different from those Forums that are sponsored by INTEROP. The following information should give the reader an overview of the Fibre Channel Systems Initiative.

Fibre Channel Systems Initiative (FCSI)

The Fibre Channel Systems Initiative, a joint effort between Hewlett-Packard, IBM and Sun Microsystems, announced the first prototype implementation of the high-speed Fiber Channel standard in August, 1993. The companies said the new fiber-optic technology "dramatically reduces the time it takes to transfer large, complex files between computers." The first implementation of the prototype technology is at Lawrence

¹ 1297 Fibre Channel Association to Advance Interconnect Standard Aug. 16, 1993, HPCwire.

Livermore National Laboratory, the interoperability test site for the Fibre Channel Systems Initiative (FCSI).¹

Launched in February, 1993, the FCSI is working toward the advancement of Fibre Channel as an affordable, high-speed interconnection standard for workstations and peripherals. Because the results of its efforts will be open and available to the public, the companies said the eventual impact of the technology will enhance the way all computers are used in business, medicine, science, education and government.

Fibre Channel simplifies the connection between workstations and supercomputers, and its speed is not affected by additional connections. It allows data to be transmitted bi-directionally at 1 gigabit per second at distances up to 10 kilometers.

"I wanted an interconnect technology that could transfer data as fast as the human eye could visualize it, and Fibre Channel was exactly what I was looking for," said Paul R. Rupert, manager of the Advanced Telecommunications Program at Lawrence Livermore National Laboratory.

Lawrence Livermore is planning to use Fibre Channel in complex computer simulations of occurrences such as fusion experiments. Because these models are so complex, they often cannot be completed on a supercomputer without first being manipulated on a workstation. This requires transferring the data from the supercomputer to a workstation for manual correction and then back to the supercomputer for completion.

Transferring this data takes up to 40 minutes using an Ethernet connection. With the prototype Fibre Channel interconnect, it will take eight minutes; with future gigabit-speed interconnects, it will take two seconds, the companies said.

"Lawrence Livermore's needs were ideally suited for Fibre Channel interconnects because their applications involve so many different computers, ranging in scale from workstations to supercomputers, and include several major brands," said Dr. Ed Frymoyer, program manager for the Fibre Channel Systems Initiative.

Frymoyer said that Livermore's wide-ranging workstation inventory has given the FCSI much-needed input for developing "profiles," or specifications, that will be available to the public. These specifications will assist computer manufacturers in designing products that have Fibre Channel technology built into them, he said.

¹ 1280 Livermore Hosts First Fibre Channel Gigabit Implementation Aug. 11, 1993, HPCwire.

Network Queuing Environment

Robert. E.Daugherty

Daniel M. Ferber

Cray Research, Inc.
655-F Lone Oak Drive
Eagan, Minnesota 55121

ABSTRACT

This paper describes the Cray Network Queuing Environment. NQE consists of four components - Network Queuing System (NQS), Network Queuing Clients (NQC), Network Load Balancer (NLB), and File transfer Agent (FTA). These are integrated to promote scheduling and distribution of jobs across a network of systems. The systems can be clustered or mixed over a wide area network or both. The Network Queuing Environment offers an integrated solution that covers an entire batch complex - from workstations to supercomputers.

Topics covered include features, platforms supported, and future directions.

CraySoft's Network Queuing Environment (NQE) is a complete job management facility for distributing work across a heterogenous network of UNIX workstations. NQE supports computing in a large network made up of clients and servers. Within this batch complex, the NQE servers provide reliable, unattended batch processing and management of the complex. The NQE clients (NQC) submit jobs to the NQE batch and monitor the job status. The Network Load Balancer (NLB) provides status and control of work scheduling seeking to balance the network load and route jobs to the best available server. The File Transfer Agent (FTA) provides intelligent unattended file transfer across a network using the `ftp` protocol.

NQE Configuration

From an administrative standpoint, an NQE configuration

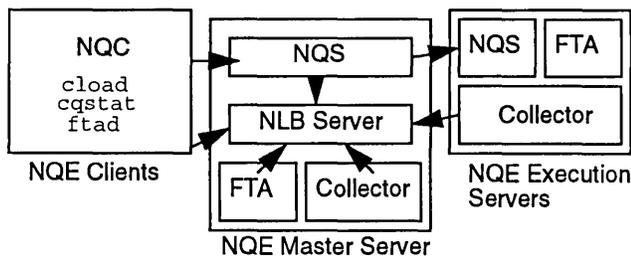


Figure 1: Sample NQE configuration

could be represented by the figure above. The NQE Master Server is the host which runs all of the software. Cray

NQS, the Network Load Balancer, FTA, and the collectors used by NLB to monitor machine workload. The NQE Execution servers which are replicated on several machines in the network contain everything needed to execute jobs on the host. The NQE clients, running on numerous machines, provide all the necessary interface commands for job submission and control.

NQE license manages the server hosts but provides you with unlimited access to NQE clients. Configuration options allow you to build redundancy in your system. Multiple NLB servers may be installed in order to provide access to the entire system in the event of the failure of on NLB server.

Network Queuing System

Cray NQS provides a powerful, reliable batch processing system for Unix networks. The systems can be clustered (tightly coupled) or mixed over a wide area network (loosely coupled) or both. NQE offers an integrated solution covering an entire batch network, from workstations to supercomputers.

Using the `qmgr` command, the NQE manager defines the systems in the network and the system operators, creates NQS queues, and defines other system parameters related to queues and how the jobs are processed.

NQS operates with batch and pipe queues. Batch queues accept and process batch job requests. Pipe queues receive

batch job requests and route the requests to another destination for processing. In creating queues, options are provided to allow control over queue destinations, the maximum number of requests that can be processed at one time, and the priority in which the queues are processed. Permissions can be created to give user access/restriction to specified queues. Limits can be set up for each queue to control the number of requests allowed in a queue at one time, the amount of memory that can be requested by jobs in the queue, the number of requests that can be run concurrently, and the number of requests one user can submit. Queues may be configured into queue complexes. In addition to the limits that may be specified in each queue, global limits can be implemented to restrict activity throughout the NQS system.

NQE provides both command-based and graphical user interfaces for job submission/control information. The `cqstat` utility provides status information of requests throughout the batch complex.

The screenshot shows a window titled "Complex cqstat display" with a menu bar (File, Actions, Filter) and a CRAY logo. The main content is a table titled "Network Job Status".

Queue	NIS id	Run User	Status	CPU Used	CPU Limit	Memory Used	Memory Limit
b_30_18e:loudy	257,mvs	zoe	W	0	30	0	1e6
b_30_18e:loudy	261,mvs	zoe	W	0	30	0	1e6
b_30_18e:loudy	265,mvs	zoe	W	0	30	0	1e6
b_30_18e:loudy	266,mvs	zoe	W	0	30	0	1e6
b_30_18e:loudy	267,mvs	zoe	W	0	30	0	1e6
batch@cloudy	125,snake-fddi	jbadger	A	0	*	0	*
b_14400_16@gust	1298,gust	c1306	R03	71	7200	8e6	2e7
b_14400_8@gust	1299,gust	c1306	R03	62	7200	4e6	9e6
batts@gust	14333,sequoia	batts	Q24	0	1000	0	2e6

At the bottom of the window, it says "Server: yankee".

Figure 2: cload display window

Network Load Balancer

The Network Load Balancer (NLB) makes load balancing recommendations to the NLB server. NLB also provides an interface for the collection and display of workload information about the network. This information, in addition to being used to set NLB job destination policies, is useful to the administrator as a graphical tool for network load display.

On each host in the network where NQE is installed, collectors are installed. These collectors send load and status information to the NLB server, installed on the NQE master server. The server stores and processes this information. NQE then queries the recommended host. The job is then sent to the selected host recommended by the load balancer. The destination selection process can be configured to take into account site-specific factors. Users may tailor the balancing policy and algorithm; exits are provided so that jobs can override the destination selection

process.

The utility `clload` provides a continually updated display of machine and load status. This tool enables you to easily compare machine loads.

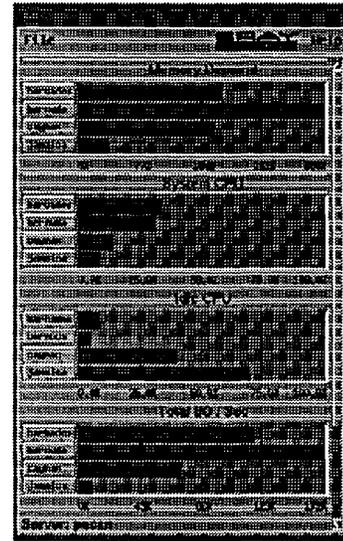


Figure 3: cload display main window

Popup windows are provided to give additional information about any host selected. This tool provides a visual way to determine which machines are heavily used and if a machine is not providing new data.

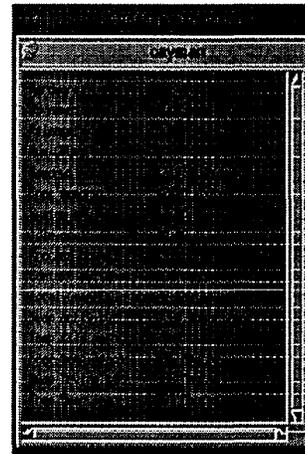


Figure 4: Host load display

The Network Load Balancer allows you to use your heterogeneous network of systems from various vendors as a single computing resource. By utilizing the NLB destination selection process, running jobs on the least loaded system will in many cases provide the best turnaround time for the user and make best use of the system resources.

File Transfer Agent

The File Transfer Agent (FTA) provides reliable (asynchronous and synchronous) unattended file transfer across the batch complex. The `ftad` daemon services FTA requests issued by users and NQE itself. The `ftua` and `rft` commands are provided for file transfer. Built upon the TCP/IP `ftp` utility, FTA provides a reliable way to transfer files across the network from your batch requests. Using `ftp`, all file transfers take place interactively; you must wait until the transfer is completed before proceeding to another task. With FTA, file transfer requests can be queued. FTA executes the transfer; if the transfer does not occur due to an error, FTA automatically requeues the transfer. The `ftua` facility is similar to `ftp` in its user interface and offers the full range of file manipulation features found in `ftp`. The `rft` command is a one-line command interface to FTA. It is a simplified interface to the `ftua` feature of copying files between hosts. In general it is a reliable mechanism for transferring files in simple operations, especially from within shell scripts or NQE.

Also FTA allows for network peer-to-peer authorization, which enables you to transfer files without specifying passwords in batch request files, or in `.netrc` files or by transmitting passwords over the network. It requires use of FTA on the local system and support for the peer-to-peer authorization on the remote system. It can be used to authorize both batch and interactive file transfers.

NQE Clients

The Network Queuing Client (NQC) provides a simplified user interface for submitting batch jobs to the network. Typically the NQC is installed on all user systems, providing a submit-only access to NQE. No NQE jobs will be run on the NQC system. NQC also provides access to the GUI job status tools, FTA, and the job submission/deletion and control commands. With each NQE license, NQC stations may be installed in unlimited quantities at no additional charge.

UNICOS NQE

A new product, NQX, will make available the following NQE components on UNICOS 8 systems; NLB collector and associated libraries, NQS interface to network load balancer, NQS server support for NQC clients, NQE clients, and the Network Load Balancer.

With the addition of NQX, UNICOS systems can

be part of the NQE batch complex. CraySoft NQE for workstations is not required. But both UNICOS and workstation systems may belong to the NQE.

Summary

The Cray Network Queuing Environment runs each job submitted to a network as efficiently as possible on the available resources. This translates into faster turnaround for users.

When developing NQS for use on Cray Research supercomputers, we made it as efficient, reliable, and production oriented as possible. With NQE we've created a high-performance integrated computing environment that is unsurpassed in performance and reliability.

This environment is now available to Cray and workstation systems in an integrated package.

Operating Systems

Livermore Computing's Production Control System, 3.0*

Robert R. Wood

Livermore Computing, Lawrence Livermore National Laboratory

Abstract

The Livermore Computing Production Control System, commonly called the PCS, is described in this paper. The intended audiences for this document are system administrators and resource managers of computer systems.

In 1990, Livermore Computing, a supercomputing center at Lawrence Livermore National Laboratory, committed itself to convert its supercomputer operations from the New Livermore TimeSharing System (NLTSS) to UNIX based systems. This was a radical change for Livermore Computing in that over thirty years had elapsed during which the only operating environments used on production platforms were LTSS and then later NLTSS. Elaborate facilities were developed to help the lab's scientists productively use the machines and to accurately account for their use to government oversight agencies.

UNIX systems were found to have extremely weak means by which the machine's resources could be allocated, controlled and delivered to organizations and their users. This is a result of the origins of UNIX which started as a system for small platforms in a collegial or departmental atmosphere without stringent or complex budgetary requirements. Accounting also is weak, being generally limited to reporting that a user used an amount of time on a process. A process accounting record is made only after the completion of a process and then only if the system does not "panic" first.

Thus, resources can only be allocated to users and can only be accounted for after they are used. No cognizance can be taken of organizational structure or projects. Denying service to a user who has access to a machine is crude: administrators can beg users to log off, can "nice" undesirable processes or can disable a login.

Large computing centers frequently have thousands of users working on hundreds of projects. These users and the projects are funded by several organizations with varying ability or willing-

ness to pay for the computer services provided. With only typical UNIX tools, the appropriate delivery of resources to the correct organizations, projects, tasks and users requires continual human intervention.

UNIX and related systems have become increasingly more reliable over the past few years. Consequently, resource managers have been presented with an attendant problem of accurate accounting for resources used. Processes can now run for days or months without terminating. Thus, a run-away process or a malicious or uninformed user can use an organization's budgeted resource allocation many times over before an accounting record to that effect is written. If a process's accounting record is not written because of a panic, the computer center is faced with possibly significant financial loss.

The PCS project, begun in 1991, addresses UNIX shortcomings in the areas of project and organizational level accounting and control of production on the machines:

THE PCS PROVIDES THE BASIC DATA REPORTING MECHANISMS REQUIRED FOR PROJECT LEVEL ACCOUNTING SYSTEMS. THIS RAW DATA IS PROVIDED IN NEAR-REAL TIME.

THE PCS PROVIDES THE MEANS FOR CUSTOMERS OF UNIX BASED PRODUCTION SYSTEMS TO BE ALLOCATED RESOURCES ACCORDING TO AN ORGANIZATIONAL BUDGET. CUSTOMERS ARE THEN ABLE TO CONTROL THEIR USERS' ACCESS TO RESOURCES AND TO CONTROL THE RATE OF DELIVERY OF RESOURCES.

THE PCS PROVIDES THE MECHANISMS FOR THE AUTOMATED DELIVERY OF RESOURCES TO ALL PRODUCTION BEING MANAGED FOR CUSTOMERS BY THE SYSTEM.

THE PCS DOES MORE THAN MERELY PREVENT THE OVERUSE OF THE MACHINE WHERE NOT AUTHORIZED. IT ALSO PROACTIVELY DELIVERS RESOURCES TO ORGANIZATIONS THAT ARE BEHIND IN CONSUMPTION OF RESOURCES TO THE EXTENT POSSIBLE THROUGH THE USE OF THE UNDERLYING BATCH SYSTEM.

CUSTOMERS ARE ABLE TO MANAGE THEIR USERS' ACCESS DIRECTLY TO A LARGE EXTENT WITHOUT REQUIRING HEAVY OR CONTINUAL INVOLVEMENT OF THE COMPUTER CENTER STAFF.

* This work was performed under the auspices of the United States Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48

It helps to state what the PCS is NOT. It is not a CPU scheduler; rather, it relies on a standard kernel scheduler to perform this

function. It is not a batch system. However, it may be regarded as a policy enforcement module which controls the functioning of a batch system. Finally, it does not do process level accounting.

While the PCS is not a memory or CPU scheduler, it does adapt the production demand on the machine to present an efficiently manageable workload to the standard memory and CPU schedulers. The PCS monitors memory load, swap device load, idle time, checkpoint space demand (if applicable), etc. to keep resource demands within bounds that are configurable by site administrators.

Overview

There are two major components of the PCS. They are the Resource Allocation & Control system (RAC) and the Production Workload Scheduler (PWS).

Resource Allocation & Control system (RAC)

Generally, the RAC system is used to manage recharge accounts, to manage allocation pools and to manage user allocations within the allocation pools. A recharge account should not be confused with a user "login" account. So that the term "group" not continue to be overused, the PCS has borrowed another term to mean an allocation pool or group. This term is "bank".

As resources are consumed on the machine the RAC system associates those resources with the users who are consuming them, a bank and a recharge account. The user and the bank are debited the resources used and an accounting report is made for the purpose of charging the account. All of this is done in near-real time. If the RAC system determines that a user, recharge account or bank has consumed as much of the resources as has been permitted, the RAC system takes steps to prohibit further resource consumption by that user, recharge account or bank via the use of graceful denial of service. Denial of service includes automatically "nicing" processes, suspension of interactive sessions and checkpointing of batch jobs.

A recharge account, or simply account, is essentially a credit that represents an amount of usable resources (which may be unlimited). Users may be permitted to charge an account for resources used. Some users, called account coordinators, may be permitted to manage the account. That is, account coordinators may grant and deny other users access to an account. Accounts are independent from each other; that is, accounts have no sub-accounts.

The primary purposes of accounts are 1) as a mechanism by which budgetary charges may be determined for organizations whose personnel use the computer and 2) a "one stop" limit on resource accessibility to users.

A primary purpose of banks is to provide a means by which the rate of delivery of allocated resources is managed. Also, the PWS uses the banking system to prioritize and manage production on the machine. This is further described in the PWS section below.

A bank represents a resource pool available to sub-banks and users who are permitted access to the bank. As implied, banks exist in a hierarchical structure. There is one "root" bank which "owns" all resources on a machine. Resources of the root bank are apportioned to its sub-banks. The resources available to each bank in turn may also be apportioned among its sub-banks. There is no limit to the depth of the hierarchy.

Some users, called bank coordinators, may create and destroy sub-banks and may grant and deny other users access to a bank. The authority of coordinators extend from the highest level bank at which they are named coordinator through out that bank's sub-tree.

Users are permitted access to a part or all of bank's resources through a user allocation. There is no limit on the number of allocations a user may have.

Production Workload Scheduler (PWS)

The Production Workload Scheduler schedules production on the machine. Production requirements are made known to the PWS in the form of batch requests. When the PWS is installed, users do not submit their requests directly to the batch system, but rather submit them to the PWS which then submits them to the batch system. When users submit a batch request, they must specify the bank from which resources are to be drawn and the account to be charged for the request's resources.

One important function of the PWS is to keep the machine busy without overloading it. Interactive work on the machine is not scheduled by the PWS. However, the PWS does track the resource load presented by interactive usage and adjusts the amount of production to "load level" the machine accordingly.

At any point, there is a set of production requests being managed by the PWS. This set is called the production workload. Requests in this workload are prioritized according to rules and allocations laid out by system administrators and coordinators. High priority requests are permitted to run insofar as the machine is not overloaded.

The PWS uses a mechanism called adaptive scheduling to prioritize a set of sibling banks. Simply stated, to schedule a set of sibling banks adaptively is to schedule from the bank (or its sub-tree) which has the highest percentage of its allocated time not yet delivered in the current shift. Each bank has associated with it a scheduling priority. This priority is a non-negative, floating-point number. The scheduling priority is a multiplier or accelerator on the bank's raw adaptive scheduling priority.

Each request has associated with it several priority attributes. They are coordinator priority, intra-bank priority, also called the group priority and individual priority. These attributes establish a multi-tiered scheme used to prioritize requests drawing from the same bank.

Some requests may be declared to be short production by a user. Requests with this attributes are scheduled on demand (as if they

were interactive). A coordinator may grant or deny any user permission to use this feature.

Vendor Requirements

The PCS requires an underlying batch processing system. The ability to checkpoint batch jobs is highly recommended where feasible, but not required.

System functionality required of the platform vendor is as follows:

The processes of a session (in the POSIX sense of that word) must be identifiable at the session level. For instance, when a user logs in, the login process (or telnetd process) should call the POSIX routine `setsid()`. Every process spawned from any child of that login (or telnetd) process should be considered to be a member of the same session (unless one of them calls `setsid()` again in which case a new session is created for that process subtree). The locality of processes in a session is not material to membership in the session. This is meant to address the issue of “parallel” processes executing in several nodes of a system. Each session existing simultaneously on a platform must have a unique identifier. This identifier must be independent of the node(s) on which the processes of the session are executing.

The resources used by processes on the platform must be collected by the underlying system at the session level. The most critical pieces of information are the session owner user id and the amount of user CPU, system CPU and user induced idle CPU time consumed by the processes of the session. User induced idle CPU time results when a CPU is dedicated to a user process which then sleeps. Other data are useful as well such as characters of I/O, paging (on virtual machines), swap space consumed, etc.

The resources used by the system’s idle processes must be collected as “idle time” and the resources used by other system processes that are not part of any session must be collected as “system time.”

There must be a way to periodically extract the data collected by the system. The periodicity of data extraction must be configurable. The manner of extraction (reading a kernel table, report by vendor supplied software, etc.) must be efficient.

Various session management functions are required. These functions take a session id and then apply the function to every process in the session. (Usually, these functions are written to allow several classes of process identifier sets such as process, process group leader, session, etc.) These functions are:

- Send any signal to all processes in the session.

- Set the nice value or priority of all processes in the session. Temporarily suspend (i.e., deny any CPU time to) all processes in the session. (This function must not be revocable by the user.)

- Revoke the action of the suspend function for all processes in the session.

Addressing the Future

Several development needs have been identified for PCS. These enhancements will make the use of PCS by administrators more flexible and will encompass a larger environment (more platforms, for instance). Users will also find the PCS to be more of an aid in the handling of their production.

PCS Support of Distributed Computing

Support should be provided for multi-host execution of a single job where the hosts are not necessarily of the same type. For instance, a portion of a job might execute on an MPP while another portion is executing on a large vector machine and finally, a supervisor processor for the job may be executing on a work station.

Distributed Management of the PCS

One problem with the current PCS in an environment with more than one host is that system administrators and coordinators must work with each PCS system independently. We need to reduce the management workload by managing the PCS for all hosts from a single platform. Furthermore, coordinators and system administrators should be able to manage the entire PCS system from any host rather than being required to manage it from a single platform.

Cross Host Submission and Management of Production

Users should be able to submit their production requests on one machine and have it execute on another. The results should be available at the submitting host or the executing host as requested by the user. Users should be able to establish cross-host dependencies for their production.

Enhanced Dependent Execution of Production

PCS already supports a dependency type wherein a request is prohibited from running until a designated job terminates. New dependency types should be supported. Some of these are:

1. Don’t run a request until and unless a designated request terminates with a specified exit status. (If a designated request terminates with an exit status other than that specified in the dependency, the dependent request is removed.)
2. Don’t run a request until a designated request has begun executing.
3. Begin the execution of a set of production requests simultaneously.
4. Don’t run a request until an “event” is posted by some other request. (This would require mechanisms by which “event definitions” are declared, the occurrence of the event is posted and event definition removed when no longer needed.)
5. Don’t run a request until specified resources (files, etc.) are staged and available.

Enhancement of PCS to Use Various Batch & Operating Systems

The PCS has been written to use the Network Queuing System provided by Cray Research, Inc. and Sterling Software. It should be extended to use other batch systems as appropriate. It has been written to run on UNICOS 7.0 and Solaris 2.1.3. It should, of course, be maintained to remain compatible with future releases of vendor’s systems.

ALACCT: LIMITING ACTIVITY BASED ON ACID

Sam West

Cray Research, Inc.

National Supercomputing Center for Energy and the Environment

University of Nevada, Las Vegas

Las Vegas, NV

Abstract

While UNICOS provides a means to account for a user's computing activity under one or more Account IDs, it doesn't allow for limiting a user's activity on the same basis. Nor does UNICOS provide for a hierarchy of Account IDs as it does with Fair-Share Resource Groups.

Alacct is an automated, hierarchically organized, SBU-oriented administrative system that provides a mechanism to limit a user's activity based on the SBU balance associated with an Account ID which the user must select.

Implementation comprises an acidlimits file along with utilities and library routines for managing and querying the file.

1.0 Introduction

NOTE: In the following the term *account* is used synonymously with Login ID. When reference is made to an 'account', in the accounting sense, Account ID or ACID will be used.

In this paper we will be discussing Alacct, an administrative system for automatically enforcing access limitation to the NSCEE's Cray Y-MP2/216 based on year-to-date accumulation of SBUs within an Account ID. We will discuss the motivation for such a system and the limitations of UNICOS that led to its development. An earlier solution to the problem, along with its shortcomings, is also described.

1.1 Motivation

The user base at NSCEE is, like many computing centers, a dynamic entity. Our users come from the University of Nevada System, private industry, government agencies, and universities outside of Nevada. Their accounts evolve over time as projects come and go and affiliations change and, likewise, the bill for their computing activity must get charged to different funding sources as those projects and affiliations change.

This dynamic nature creates problems for the system manager who has to deal with the changes. At NSCEE, since account creation and deletion is done manually, automation of any portion of that process is a boon. Fortunately, accounting for a particular user's computing activity is a feature that is already provided by Unix, and, by extension, UNICOS. In fact, UNICOS goes a step further and allows for accounting by UID-ACID pair, to facilitate accounting for one user working on more than one project. Also, CSA, Cray Research System Accounting, which is an extension to Standard Unix Accounting, provides a site-configurable System Billing Unit, or SBU, to express in a single number a user's resource utilization.

1.2 Limitations of UNICOS

However, important capabilities of system management associated with user accounts that are not directly provided by UNICOS are the ability to automatically force a user to select a project (Account ID) against which to bill a session's computing, and then to limit his or her computing activity on that same basis. These capabilities are important for, at least, two reasons: to the extent that it is possible you want to be able to correlate the consumption of resources to actual projects; and, like a 1-900 telephone number, you don't want users consuming resources for which they may be unable, or unwilling, to pay.

Currently, UNICOS provides no way to force Account ID selection. With respect to the second capability, the only means provided, other than manually turning off a user's account, is the cpu resource limitation feature of the `limit(2)` system call (with the user's limit set in the `cpuquota` field of the UDB.) This mechanism suffers from two inadequacies, however: cpu utilization does not completely represent the use of a complex resource like a supercomputer; and, even worse, since there is only one lnode per user the `cpuquota` is tied only to the UID, not to a UID-ACID pair. So, disabling a user on this basis means disabling him or her entirely, not just for activity on a particular project.

2.0 Background

2.1 Multiple UID Accounting

An early attempt at solving this problem at NSCEE, Multiple UID Accounting, involved the creation of multiple accounts per individual user, with each account being a member of a class of account types: Interim, Class, DOE, etc. The individual accounts were distinguished from one another by a suffix letter that indicated their class membership: i for Interim, c for Class, d for DOE, etc. Likewise, each account type had its own Account ID. Users would then use the Login ID that was associated with the particular project he or she wanted to work on for that session. UIDs were allocated on multiple-of-5 boundaries to allow for future account types.

This mechanism allowed for billing to be tied to a particular user's activity on a particular project, insofar as was possible, while guaranteeing that, when a user had reached his or her limit of resources for that project the system administrator could disable the user's account associated with that project without disabling any of the user's other, viable, accounts. Since, at NSCEE, a user's bill is generated directly from his or her SBU charges, the aforementioned limit was expressed in SBUs.

The implementation of this scheme involved a file of per-Login ID limits and a daily report that cross-referenced each Login ID's year-to-date system utilization with its limit to produce an exception list. The system administrator would use that exception report to manually disable any account that was over its SBU limit. Again, the primary reasons for solving the problem in this way were that it allowed NSCEE to take advantage of existing mechanisms for forcing users to consciously choose a project and for allowing the system administrator to limit activity based on SBU utilization on one project without disabling the user entirely.

Needless to say, this became something of a nightmare to administer and the users liked it even less. Limits had to be set on a per-user basis, instead of a class of users. Also, it should be pointed out that this scheme essentially subverted the purpose of allowing multiple Account IDs per Login ID as a means of accounting for the multiple projects of a single user.

2.2 A New Way

Therefore, a New Way was sought. NSCEE's Director, Dr. Bahram Nassersharif, had recently come from Texas A&M University, where Victor Hazelwood had implemented a strategy for automatically limiting, at login, a user's access to the system when the user had exceeded an SBU Quota. This sounded like a much better way and we decided to head in that direction and begin work on implementing it immediately.

After some discussion, however, we concluded that we needed some features that were not provided by the TAMU system, especially a hierarchically organized Account ID framework.

So, we decided to write the system from scratch incorporating the ideas we got from TAMU and the experience we already had with Multiple UID Accounting.

3.0 Requirements and Constraints

After further discussion, a list was formulated of features and capabilities the NSCEE needed out of this project, as well as the constraints under which we should proceed.

Alacct should:

- Be easy to implement.
- Limit users based on the aggregate SBU consumption by all users within a particular Account ID.
- Allow for hierarchical Account ID management.
- Force Account ID selection at login, etc.
- Be easy to administer.
- Be portable to new releases of UNICOS.
- Be robust.

Alacct should not:

- Require kernel mods.
- Require multiple accounts per user.
- Have significant system impact.
- Have significant user impact.

4.0 Implementation

Alacct is implemented at the user level. It is not real-time, in the sense that new usage is reflected only after daily accounting has been run. It is fairly simple: a central file keeps track of Account ID limits and usages; a library of user-callable routines queries that file and modifications to critical UNICOS commands make calls to those routines; and administrative utilities are used to maintain that file.

4.1 Acidlimits

The focal point of the implementation of alacct is the:

```
/usr/local/etc/acidlimits
```

file. This file contains limit, usage and hierarchy information for each Account ID on the system. It is queried whenever an Account ID must be verified to have a balance greater than 0.00. This happens during login, execution of the su(1) and newacct(1) commands and also in nqs_spawn, prior to starting a user's NQS job.

The acidlimits file is symbolic and made up of entries like:

```
21100:102.00:0.00:5.72:20000
```

Where the five fields represent:

```
acid:limit:usage:progeny_usage:parent_acid
```

The acid field is an integer representing the numeric Account ID.

The `limit` field is a float representing the maximum number of SBUs that can be consumed, collectively, by all users that process under the specified Account ID and its progeny (see the `progeny_usage` field.)

The `usage` field is a float representing the year-to-date, collective SBU consumption by all users who have processed under the specified Account ID (as well as optional historical usage - see `al_usages`.)

The `progeny_usage` field is a float representing the recursively computed sum of all `usage` field entries for all progeny in the hierarchy of the Account ID represented in the `acid` field.

The `parent_acid` field is an integer representing a numeric Account ID. This field defines a precedence relation on the entries in the file to establish a hierarchy of Account IDs. This hierarchy takes the form of an inverted tree like that in figure 1.

In figure 1 we see a list of users at the bottom. Each of these users has the Account ID `GrCP13` as one of their Account IDs. `GrCP13`'s parent Account ID is `GrantE`, whose parent is `Grant`, whose parent is `Internal`. `Internal` has, in its parent field, 0, which, when appearing as a parent Account ID, represents the meta-root Account ID (since 0 is a valid Account ID and must also have a line in the `acidlimits` file.) All top-level Account IDs, including root, have the meta-root as their parent, to complete the hierarchy.

Note that, although in figure 1, users only appear on a leaf of the tree, there is nothing that prevents users from having one of the hierarchy's internal Account IDs as one of their Account IDs.

This file contains an entry for all Account IDs currently in use

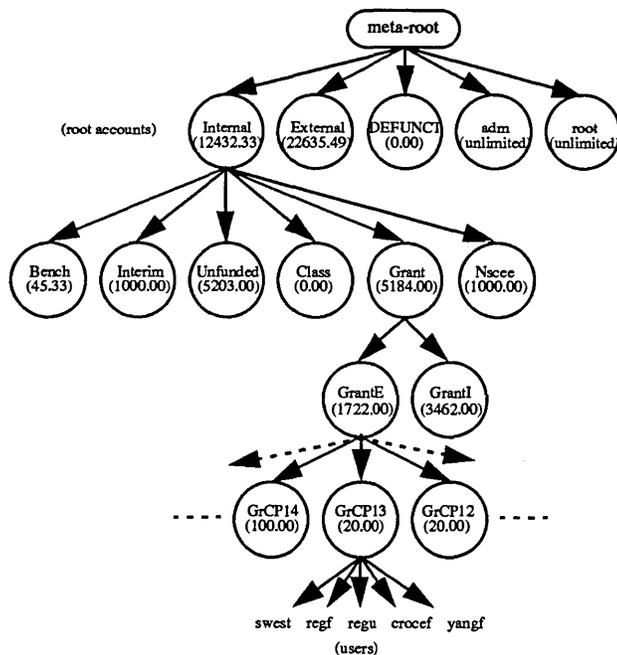


figure 1

by any user on the system. If a new Account ID is added to the

`/etc/acid`

file, then, the next time the `acidlimits` file is created, that new Account ID will be inserted with `meta-root` as its parent and a limit of unlimited. As you will see later, a new `acidlimits` file is created from, among other sources, the old `acidlimits` file. In the bootstrap process, when the system is first installed and there is no old `acidlimits` file, an `acidlimits` file is created from `/etc/acid` with a one-level hierarchy, `meta-root` as every Account ID's parent, and unlimited limits for every Account ID.

Note that since `acidlimits` is a symbolic file, it can be created or modified by hand with your favorite editor. In a system with few Account IDs this might be feasible. However, there is an `acidlimits` editor, `aledit(8l)`, that understands the structure of the file and makes changing the file a simple task (see `aledit`).

4.2 Local System Modifications

In this implementation there are two times when the user is forced to select the Account ID to which his or her computing activity will be billed (and, by virtue of which, access may be denied): at login and when `su'ing`. Likewise, it is at these times, and also when requesting a change of Account IDs with `newacct(1)` and starting an NQS job with the user's current Account ID, that the Account ID is validated for a positive balance. To achieve these actions modifications were made to the following source files:

- `login.c`
- `newacct.c`
- `su.c`
- `nqs_spawn.c`

To keep the number of mods small this list is, intentionally, short. Notably absent are `cron(1)` (and, by extension, `at(1)`) and `remsh(1)`. The user's default Account ID is used in these cases. In practice, this has not created problems. Also unmodified is the `acctid(2)` system call, since kernel mods were specifically avoided and, anyway, only root can change a user's Account ID.

4.3 libal.a

The aforementioned mods involve calls to a library, `libal`, which was written to support the `alacct` system. That library contains the following routines:

- `alget()` Get a single entry from the `acidlimits` file
- `alput()` Update a single entry in the `acidlimits` file
- `alread()` Read the entire `acidlimits` file
- `alwrite()` Write the entire `acidlimits` file
- `algetbal()` Retrieve the current balance of a particular Account ID
- `alselect()` Display, and query for selection, the user's current Account IDs and their respective balances

```

localhost
Cray UNICOS (clark,nscee,edu) (tty005)

National Supercomputing Center for Energy and the Environment
University of Nevada Las Vegas

Use of this system is restricted to authorized users.

login: swest
Password:
Valid account ids and balances for user swest:
 1. root - unlimited SBUs
 2. adm - unlimited SBUs
 3. Nscee - 908.35 SBUs
 4. Class - 0.00 SBUs
 5. GrCP13 - 17.81 SBUs
   please select by line number.
account id? 3
Last successful login was : Thu Mar 10 09:33:29 from localhost

```

figure 2

An Account ID's balance is recursively computed as the lesser of: a) the difference of its limit and the sum of its usage and its childrens' usage, and b) the difference of its parent's limit and the sum of its parent's usage and its parent's childrens' usage, with the meta-root as the limiting case for the recursion (meta-root's limit is unlimited.)

In other words, an Account ID's balance is the least of all balances of all Account IDs on the path back to the meta-root. This means that if any Account ID has exhausted its allocation, then all its progeny have, effectively, exhausted theirs.

It is the requirement that the predecessors' balances be positive that gives function to this hierarchy. If, for example, it were the case, referring again to figure 1, that Internal accounts (i.e. Account IDs whose predecessor path includes the Internal Account ID) had an aggregate limit which was greater than the limit of the Internal Account ID itself, then it would be possible to exhaust the allocation to *all* Internal accounts before any *one* of those accounts had reached its own limit. Also, this scheme allows for special users, perhaps the principal investigator on a project, to have access to one of the Account IDs that is an internal node of this hierarchy, and draw on its balance directly, using it up before the users who have access to Account IDs further down the hierarchy have a chance to use theirs.

5.0 User Interaction

As mentioned, there are four instances when a user would be aware that there is something affecting his or her processing. At login, when issuing the su and newacct commands, and when his or her NQS job is started.

5.1 login(1) and su(1)

When a user logs in, or issues an su command, he or she will see a screen like that in figure 2. This is the output from the else-lect() library call.

The user must select one of the line numbers corresponding to an Account ID with a positive balance. If a valid Account ID is selected, then that Account ID becomes the current Account ID and the login process (or su) is successfully completed. If an invalid Account ID is selected, the user is notified and logged off (or the su fails).

5.2 newacct(1)

In the case of the newacct(1) command, its invocation is unchanged. To change Account IDs the user still specifies the new Account ID to which he or she wishes to change. With alacct in operation, however, the user's selection is not only verified as a legitimate Account ID for that user, but the balance is also verified for a balance greater than 0.00. Examples of newacct's operation appear in figure 3.

5.3 nqs_spawn

NQS jobs carry the Account ID of the qsub'ing user. It is possible that, by the time the user's job is selected for processing by the nqsdaemon, the limit of the requested Account ID will have been exceeded. For this reason, the check for positive balance is deferred until the job is actually spawned by NQS. If at that time the Account ID is over its limit, the job is rejected and e-mail is sent to the user.

6.0 Administration

The day-to-day administrative requirements of this system are minimal. In addition to any Account ID additions, deletions or modifications (that would be made with the aforementioned aledit program) the acidlimits file must be revised daily to reflect the current year-to-date usage of the various Account IDs on the system. Of course, this means that the site must produce a year-to-date accounting file at least as often as the site wishes the usage information stored in the acidlimits file to be accurate. UNICOS provides a fairly easy method of doing this without requiring that the system administrator maintain on-line all accounting data for the year.

6.1 CSA

CSA produces summary data for the current accounting

```

localhost
/u1/cr1/swest 102=> newacct -a
root (0), account balance - unlimited
adm (7), account balance - unlimited
Nscee (21400), account balance - 908.35
Class (21600), account balance - 0.00
GrCP13 (23513), account balance - 17.81
/u1/cr1/swest 103=> newacct -l
Current account name: Nscee, account balance - 908.35
/u1/cr1/swest 104=> newacct Class
newacct: account balance < 0.0, account unchanged
/u1/cr1/swest 105=>

```

figure 3

period when daily accounting (csarun) is run (it is referred to as daily accounting, but it can be run on any boundary desired.) This data is 'consolidated' by the csacon(8) program into cacct format. CSA also allows for 'periodic' accounting to be run which adds together, with csaaddc(8), an arbitrary collection of cacct files to produce a file that represents accounting for a specified period. By default, csaperiod stores its output in the fiscal subdirectory of the /usr/adm/acct directory.

At NSCEE, csaperiod is run on the first day of every month for the previous month to produce monthly accounting. These monthly files are kept on-line and the daily files that produced them are cleaned off to start the new month. Year-to-date accounting then, would be the summation of all the fiscal files along with the current month's daily files. To accomplish this, a local variation of csaperiod was written, csaperiod_year. Csaperiod_year is like csaperiod, except that, in addition to the /usr/adm/acct/sum directory, it also knows about the /usr/adm/acct/fiscal directory to add in monthly data, as well.

Figure 4 illustrates the year-to-date accounting process which

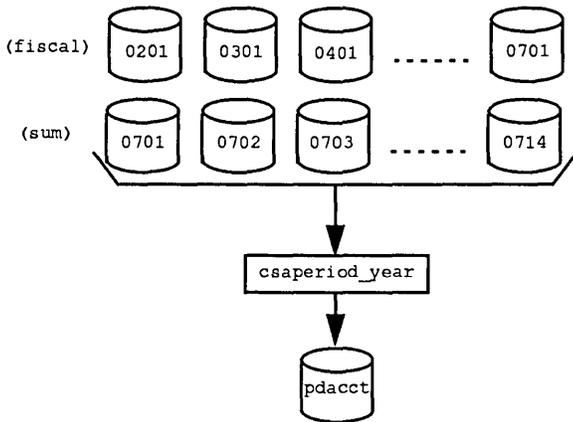


figure 4

would take place on July 14. Periodic (fiscal) data for each of the months from January through June along with the Daily (sum) accounting files for the month of July, which are all in cacct format, would be added together to produce the current year-to-date accounting file - pdacct.

6.2 mkal

As mentioned before, the year-to-date accounting file contains the information that will be used to keep the usage fields in the acidlimits file current. For this purpose, there is the mkal(81) program. Mkal updates the acidlimits file with current usage information by reading the year-to-date accounting data file,

pdacct (see figure 5.)

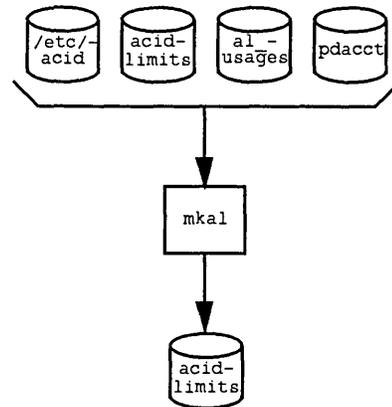


figure 5

The program operates in the following steps:

1. Read /etc/acid to get a current list of valid Account IDs. All Account IDs start with the default limit of unlimited.
2. Read /usr/local/etc/acidlimits to bring forward the most recent limit information.
3. Read /usr/local/etc/al_usages to update the Account IDs' usage field to reflect historic usage (optional).
4. Read /usr/local/adm/acct/pdacct, summing all usage for each Account ID and then update the Account IDs accordingly.
5. Rewrite /usr/local/etc/acidlimits.

6.3 crontab

The accounting and mkal operations have been automated and figure 6 shows the relevant crontab entries which, each day, run daily accounting, year-to-date accounting, and mkal.

```

#
# accounting stuff
#
0 * * * * /usr/lib/acct/ckpacct
0 * * * * /usr/lib/acct/ckdacct nqs
30 0 * * * /usr/local/lib/acct/daily
0 1 * * * /usr/lib/acct/dodisk -a -v 2> /usr/adm/acct/nite/dk2log
0 2 * * * /usr/lib/acct/csarun 2> /usr/adm/acct/nite/fd2log
0 3 1 * * /usr/local/lib/acct/csaperiod_mont
10 3 * * * /usr/local/lib/acct/csaperiod_year
0 4 * * * /usr/local/lib/acct/mkal
#
# air stuff
#
15 * * * * /usr/air/bin/airdchk
0 0 * * 0 /usr/air/bin/mvfiles
#
# sa stuff
#
0 * * * * /usr/lib/sa/sa1 600 6 &
30 1 * * * /usr/lib/sa/sa3 -ubuqv
#
# data migration
#

```

figure 6

6.4 aledit

If resource limits change, or a new Account ID has been created, the acidlimits file must be rewritten to reflect this change or addition.

The aledit(8l) program is the primary maintenance tool for the alacct system. It operates in two modes, Interactive and Display, and allows a suitably privileged user to interactively view and edit the acidlimits file, or to display a formatted, hierarchically arranged report of the contents of that file.

In Display Mode the aledit command accepts options to produce reports showing:

- The entire acidlimits hierarchy, appropriately indented (a portion of an example of which is shown in figure 7.)

```

/usr/local/lib/acct/aledit -t l more
root unlimited 0,00 unlimited
  Internal 12432,33 4383,17 8049,16
    Bench 45,33 13,00 32,33
      BenBal 4,00 3,85 0,15
      BenChem 41,33 9,15 32,18
    Interin 1000,00 495,18 504,82
    Unfunded 5203,00 1974,07 3228,93
      UnfP25 200,00 0,03 199,97
      UnfP24 200,00 0,00 200,00
      UnfP23 300,00 0,00 300,00
      UnfP22 22,00 7,17 14,83
      UnfP21 150,00 0,49 149,51
      UnfP20 300,00 1,53 298,47
      UnfP19 183,00 0,19 182,81
      UnfP18 40,00 26,51 13,49
      UnfP17 200,00 0,00 200,00
      UnfP16 10,00 9,81 0,19
      UnfP15 4,00 0,00 4,00
      UnfP14 20,00 20,44 -0,44
      UnfP13 400,00 407,20 -7,20
      UnfP12 170,00 173,97 -3,97
      UnfP11 811,00 741,20 69,80
      UnfP10 200,00 11,57 189,43
      UnfP9 10,00 10,18 -0,18
      UnfP8 100,00 110,62 -10,62
      UnfP7 40,00 41,58 -1,58
      UnfP6 800,00 0,00 800,00
      UnfP5 293,00 33,49 259,51
      UnfP4 200,00 16,46 183,54
      UnfP3 200,00 3,62 196,38
      UnfP2 250,00 251,44 -1,44
      UnfP1 100,00 106,58 -6,58
    Class 0,00 0,00 0,00
    Grant 5184,00 1737,69 3446,31
      GrantE 1722,00 206,18 1515,82
        GrCP18 40,00 0,00 40,00
        GrCP17 22,00 0,00 22,00
        GrCP16 217,00 0,00 217,00
        GrCP15 32,00 0,00 32,00
        GrCP14 100,00 101,13 -1,13
        GrCP13 20,00 2,61 17,39
        GrCP12 20,00 19,10 0,90
        GrantCP9 20,00 0,00 20,00
        GrantCP8 145,00 7,35 137,65
        GrantCP7 20,00 0,09 19,91
        GrantCP6 0,00 19,35 -19,35
        GrantCP5 20,00 24,31 -4,31
        GrantCP4 44,00 0,00 44,00
        GrantCP3 207,00 0,00 207,00
        GrantCP2 207,00 0,00 207,00
        GrantCP1 608,00 32,23 575,77
      GrantI 3462,00 1531,52 1930,48
        GrUcp19 60,00 2,10 57,90
        GrUcp18 105,00 2,00 103,00
        GrUcp17 100,00 0,00 100,00
        GrUcp16 10,00 0,00 10,00
        GrUcp15 58,00 0,00 58,00
        GrUcp14 425,00 17,29 407,71
          GrUcp14a 150,00 17,19 132,81
        GrUcp13 34,00 0,00 34,00
        GrUcp12 21,00 0,00 21,00
        GrUcp11 24,00 0,00 24,00
        GrUcp10 102,00 0,00 102,00
        GrUcp9 11,00 0,00 11,00
  
```

figure 7

```

localhost
Internal 12432,33
External 22635,49
DEFUNCT 0,00
adm unlimited
root unlimited

Account = root (unlimited) (k)up (j)down (h)left (l)right (i)higher (w)lower
Page 1 of 1 (5 accounts) (/)srch (+,-)pg+-1 (^U)users (b,B)balance (?)help
account total = unlimited (x)mark (p)put (P)Put (U)unlimit (U)update (q)quit
  
```

figure 8

- The entire acidlimits hierarchy along with users
- A single Account ID, including users, progenitor path and children

Executing aledit with no options places the user in Interactive Mode. When you execute aledit in Interactive Mode, the program reads and assembles, into a tree, the Account ID hierarchy defined by the predecessor relation found in the acidlimits file. The terminal display is then initialized to show the children of the 'meta-root' account, and input is accepted directly from the keyboard (see figure 8).

As you can see, the normal display mode shows a list of Account IDs down one column, and their corresponding limits in another. Notice the special limit 'unlimited'. Account IDs with an unlimited limit are not subject to usage deductions when calculating a balance.

The accounts on the screen at any given time constitute a peer group of accounts, in that they are at the same level in the hierarchy and have the same parent. The parent is shown, along with some other information, and help messages, at the bottom of the display (users of the UNICOS shrdist(8) command may recognize the format - portions of the display code for aledit were derived from shrdist.)

There are several major functions provided within aledit:

- Movement within the Hierarchy
- Changing Limit Values
- Changing the Hierarchy
- Changing Display Modes

Notice that in figure 8 the cursor is resting on the limit value for the Internal Account ID. Changes to that limit can be made by simply typing in a new value. To move up or down the list, the 'k' or 'j' keys are pressed. If this peer group had a parent Account ID, which in this case it doesn't, pressing the 'i' key would move the display up one level to display the peer group of the parent. Also, if Account ID Internal had children, which it does, pressing the

'm' key while the cursor is resting on Internal will take the user down the hierarchy to display the children of that Account ID (see figure 9.)

```

localhost
Bench      45.33
Interim    1000.00
Unfunded   5203.00
Class      0.00
Grant      5184.00
Nscee      1000.00

Account = Internal (12432.33) (k)up (j)down (h)left (l)right (i)higher (m)lower
Page 1 of 1 (6 accounts) (/)srch (+,-)pg+,-1 (^U)users (b,B)balance (?)help
account total = 12432.33 (x)mark (p)put (P)Put (U)unlimit (u)update (q)quit

```

figure 9

The Account IDs in figure 9 are peers of one another and are children of the Internal Account ID. Notice that the total of all limits for all Account IDs on this display is 12432.33 SBUs, which matches the limit on the Internal Account ID itself. This is not a requirement. That total may be greater or less, than the parent limit, depending on the circumstances. As an aid, however, since those values will be frequently be the same, there is a balance function that propagates the total up one level to the parent, or all the way up the hierarchy to meta-root.

Continuing with this example, we see, in figures 10 and 11 the displays for the children of the Grant Account ID, followed by the

children of the GrantE Account ID.

```

localhost
GrCP18     40.00
GrCP17     22.00
GrCP16     217.00
GrCP15     32.00
GrCP14     100.00
GrCP13     20.00
GrCP12     20.00
GrantCP9    20.00
GrantCP8    145.00
GrantCP7    20.00
GrantCP6     0.00
GrantCP5    20.00
GrantCP4     4.00
GrantCP3    207.00
GrantCP2    207.00
GrantCP1    608.00

Account = GrantE (1722.00) (k)up (j)down (h)left (l)right (i)higher (m)lower
Page 1 of 1 (16 accounts) (/)srch (+,-)pg+,-1 (^U)users (b,B)balance (?)help
account total = 1722.00 (x)mark (p)put (P)Put (U)unlimit (u)update (q)quit

```

figure 11

Finally, in figure 12, if we change display modes to show users, we see the list of users who have Account ID GrCP13 as one of their Account IDs:

As mentioned before, aledit also allows for the manipulation of the hierarchy itself. This is accomplished in three steps by:

1. Marking with an 'x' an Account ID, or list of Account IDs,
2. Positioning the cursor on the parent (or, alternatively, the peer group) where the Account ID is to be moved, and
3. Pressing 'p' (or 'P').

Note that if the hierarchy is changed in any way, mkal must be rerun to correctly compute the new progeny_usages.

6.5 al_usages

As mentioned earlier the al_usages file is an optional feature of alacct that allows for mapping SBU usage by a particular UID to a new Account ID against which this usage must be deducted.

```

localhost
GrantE     1722.00
GrantI     3462.00

Account = Grant (5184.00) (k)up (j)down (h)left (l)right (i)higher (m)lower
Page 1 of 1 (2 accounts) (/)srch (+,-)pg+,-1 (^U)users (b,B)balance (?)help
account total = 5184.00 (x)mark (p)put (P)Put (U)unlimit (u)update (q)quit

```

figure 10

```

localhost
acid: GrCP13  limit: 20.00  usage: 2.19  balance: 17.81
users:
  swest regf regu crocef yangf

more? █

```

figure 12

This mechanism allows us to bring forward from previous years historical usage that must be tied to a particular Account ID. The `mka_usages(8)` program is usually used at year-end to create from the final year-to-date accounting file an `al_usages` file for the coming year which contains Account IDs that will be carried forward into the new year.

7.0 Future Plans

Just like the `shrdist(8)` command allows a point-of-contact (POC) for each resource group to control the allocation of resource shares in his or her group, a POC should be able to distribute limits to children of the Account ID he or she controls.

Improvements to the timeliness of the `acidlimits` file could be made with a new routine to summarize the current accounting period's data and update a new field in the `acidlimits` file - `today's_usage`.

Approximation to real-time operation could be implemented with a daemon process which would scan the process list as often as desired and, using the aforementioned improvement to the `acidlimits` file, kill processes with Account IDs which are over the newly computed usages.

8.0 Summary

`Alacct` is an administrative extension to UNICOS that allows for the automatic limitation of computing activity based on SBU consumption by individual user accounts within an Account ID.

`Alacct` requires modifications to selected commands in UNICOS. These modifications cause the commands to make calls to a library of routines, `libal.a`, to query a file that contains limit information for individual Account IDs. This '`acidlimits`' file contains an entry for each Account ID on the system and is organized hierarchically such that limits are imposed from the top down while usage is propagated from bottom up.

Administratively, `alacct` is maintained by manipulating the `acidlimits` file directly with the `aledit` program, which allows for changes to limits and to the hierarchy itself, and indirectly with the `mka` program which, using current year-to-date accounting information reconstructs the file when necessary to reflect current usages.

Users interact with this system directly, and are forced to select an Account ID under which to compute, when logging in and when executing the `su(1)` and `newacct(1)` commands. Users interact with this system indirectly when their NQS job is started.

Since the implementation of `alacct` is fairly simplistic and it does not operate in real-time, the potential exists for users to subvert its purpose. However, the safety net of a robust UNICOS accounting system guarantees that no SBU will go unbilled, making the greatest risk, therefore, that a user might have to be turned off manually.

System impact is minimal: approximately 45 seconds of real time (10 cpu seconds) were required on a lightly loaded system to compute year-to-date accounting for March 12 (2 fiscal files and 11 daily files). `mka` operates on the year-to-date accounting file in under 1 second to produce the `acidlimits` file. Disk requirements at NSCEE are approximately 700 Blocks (~3MB) per `cacct` file, whether monthly or daily, so on December 31, we would need approximately $(11 + 30) * 700$ Blocks = 28700 Blocks (~120MB). These values will, of course vary from system to system depending on the number of active Account IDs and UIDs.

9.0 Acknowledgments

Thanks very much to Joe Lombardo and Mike Ekedahl at NSCEE for a final reading of the paper. Also, thanks to Joe for using `aledit` while I got it right. Thanks to Victor Hazelwood for suggestions and for already figuring out that this was a reasonable thing to do.

10.0 References

- 1 Cray Research, Inc., *UNICOS System Administration Volume 1*, SG 2113 7.0, A Cray Research, Inc. Publication.
- 2 Cray Research, Inc., *UNICOS System Calls Reference Manual*, SR 2012 7.0, A Cray Research, Inc. Publication.
- 3 Cray Research, Inc., *shrdist v. 7.0*, Unpublished Source Code.
- 4 Cray Research, Inc., *UNICOS 7.0*, Unpublished Source Code.

Appendix 1 - Alacct(3l) man page

alacct(3l) LOCAL INFORMATION

NAME
alacct - Introduction to local alacct UNICOS enhancement

DESCRIPTION
alacct is the name given to an NSCEE enhancement to UNICOS that permits NSCEE to automatically limit computing activity based on SBU utilization within an account.

alacct comprises the following elements:

- o /usr/local/etc/acidlimits File of acid limits, usages and hierarchy information. Usage is updated on a daily basis by the mkal utility.
- o /usr/local/etc/al_usages (optional) File of historical usages.
- o /usr/local/lib/libal.a Library of acidlimits access and update routines.
- o /usr/local/lib/acct/mkal Administrative utility to update usages in the acidlimits file.
- o /usr/local/lib/acct/aledit Administrative utility to maintain the acidlimits file.
- o /usr/local/adm/acct/pdacct Year-to-date accounting information.
- o modifications to
 - /bin/login
 - /bin/su
 - /bin/newacct
 - /usr/lib/nqs/nqsdaemon

NOTES
This extension to UNICOS requires mods to the following system source files (and their respective Nmakefiles):

```
/usr/src/cmd/newacct/newacct.c (mod 00cmd00007a)
/usr/src/cmd/su/su.c (mod 00cmd00008a)
/usr/src/cmd/login/login.c (mod 00cmd00009a)
/usr/src/net/nqs/src/nqs_spawn.c (mod 00nqs00010a)
```

For further information regarding these mods, please see their source in /usr/src/rev/local.

FILES
/usr/local/etc/acidlimits
/usr/local/adm/acct/pdacct
/usr/local/etc/al_usages
/usr/local/lib/libal.a
/usr/local/acct/lib/mkal
/usr/local/acct/lib/aledit

SEE ALSO
libal(3l)
acidlimits(5l)
mkal(8l)
aledit(8l)

Priority-Based Memory Scheduling

Chris Brady, Cray Research, Inc.

Don Thorp, Cray Research, Inc.

Jeff Pack, Grumman Data Systems, Inc.

Fleet Numerical Meteorology and Oceanography Center
Monterey, California

Abstract

The Fleet Numerical Meteorology and Oceanography Center (FNMOC) is a production center, providing environmental forecasts for the US Navy and the other DOD services around the globe. As the development of the environmental models progressed and the production job mix became more predictable, it became apparent that existing scheduler features would not adequately handle our particular job mix of high priority production work and lower priority development work on a C90 with memory over-subscription.

Production jobs run at various times throughout the 24 hour day. When a production job is queued, it must run immediately and get all necessary resources. This requires Fair Share, large swap areas to hold the preempted development work and scheduler mods to compute swap priorities based on Fair Share usage.

This paper describes the changes that were made to the scheduler and the rationale behind the mods. We hope to have the new features included as a design feature of UNICOS 9.0.

Introduction

FNMOC is the U. S. Navy's primary numerical prediction center for automated oceanographic and atmospheric analyses and forecasts and applied products. FNMOC is one of a half-dozen centers worldwide running global and regional atmospheric models on an operational basis, and is the world leader in performing oceanographic and coupled air-ocean modeling operationally.

The current production system at FNMOC is anchored with a CDC Cyber 205 that has been in use since 1982. In order to improve and enhance the capabilities of FNMOC, a program was started to install new large scale computing resources and associated support hardware and software.

Grumman Data Systems was awarded the contract to provide and integrate the new program. A Cray Research, Inc. Y-MP 2E was installed in November 1991 to operate the Empress database management system (DBMS) for the larger compute server, the Cray C90, which was installed in September 1992. Presently, the new systems are running a "pseudo" operational job mix concurrently with additional model and DBMS development work. The new systems are scheduled to be performing the complete operational workload by June 1994.

In order to provide the operational jobs the resources they require in the current configuration, we first utilized the Fair Share Scheduler and assigned the operational user accounts the highest share of the resources. This provided satisfactory CPU scheduling but we were unable to obtain consistent results with the memory scheduler. We found that operational jobs were being swapped out with development jobs. We decided to look at the memory scheduling algorithm and see if it could be modified to provide prioritized memory scheduling that was consistent with CPU scheduling priorities in order to accomplish our scheduling objectives.

Discussion

System Configuration

A brief description of the system configuration will help in understanding the following discussion. The compute server is a Cray Research, Inc. (CRI) C90 with 8 CPUs and 128 MW of memory. A 256 MW SSD is used for swap, lldcache and SDS space for jobs. The C90 is running UNICOS 7.C.3. The database server is a CRI Y/MP 2E/232 with a 128 MW SSD used for lldcache on the database disk devices. The 2E operates with UNICOS 7.0.5.1. The two machines are linked by a direct double-wide HiPPI and an Ultraset HiPPI connects each machine to the LAN.

Scheduling Requirements

FNMOC is a production site that operates on a strict twelve hour job schedule. Each job has a start and end time that should not vary by more than a few minutes each time it runs. This schedule is necessary in order to transmit environmental products to end-users in a timely manner. Some of the jobs depend on results from a previous job so overall success depends on all jobs finishing in a prescribed manner. The run times are from five minutes to ninety minutes, with most processes multitasked over 6 CPUs. The total number of production jobs run in a twelve hour period is currently around 90, with more planned for the final production configuration.

Along with the normal production jobs is a mix of unscheduled batch and interactive development work. The development jobs can use any spare resources that are available any time of the day, but when a production job requires resources, the development jobs must be swapped. The developers are aware of the operational requirements and accept the fact that their batch or interactive job may be swapped for a long time, sometimes even hours.

Configuration Settings

The Fair Share Scheduler was implemented to enforce the job scheduling policy. Currently, 99% of the C90's resources are allocated to the production jobs, which at first look may be overkill but accurately demonstrates the scheduling requirements of FNMOC. The combination of shares allocated and other scheduling parameters determine the operation of the memory scheduler.

We have created a separate NQS queue for production jobs that has no limits on resources and allows any number of jobs to be running. The development jobs use a traditional queue structure with memory and run limits. The number of development jobs running at any one time is usually small and the jobs normally begin execution upon submission. It is rare to see a job waiting in an NQS queue.

The swap space is configured large enough to hold all possible development and production jobs that could normally be running at any one time.

The `nschedv(8)` parameter `hog_mem_max` is set so that interactive workload is allowed, but no more than 10% of available memory is allocated to interactive processes. Occasionally, a large memory interactive process (`cdbx(1)`, for example) is observed but these processes have not been frequent enough to cause a problem with scheduling or

swap space. Also, the `memhog` value in `nschedv` is set to allow most normal interactive processes to not be classified as hogs, whereas most production jobs are hog processes.

Problems Encountered

As our load of production jobs increased, we noticed that, even with Fair Share weighted towards the production jobs, a large development job would cause a production job to swap. Our goal is to force development processes to swap when production jobs need memory and eliminate swapping of production jobs. We started testing various settings of the priority factors (*pfactors*) on `nschedv` and found that these were ineffective. The *pfactors* use the process priority (*p_upri*) to compute swap priorities. However, the *p_upri* is adjusted many times a second making it much too volatile to be useful for memory scheduling.

Another factor in swapping is the *hard sleeper* processes classification. In our case, we were concerned with processes waiting on network I/O, since all of the production and development jobs do network I/O in the form of requests to the database server. The scheduler assigns any processes waiting on a socket with a priority of 26 which designates those processes as *hard sleepers*. When memory is overcommitted and the scheduler is looking for processes to swap out, the scheduler swaps *hard sleepers* first regardless of swap priority. This behavior had a severe negative impact on our priority based scheduling goals.

We also noticed that occasionally a hog process that should have fit in the available memory was held out because the system had incorrectly calculated that the available hog memory was insufficient for that process.

Finally, SDS space was being used by development jobs while a request for SDS space from a production job was blocked. We wanted to see if we could force the development job to release SDS space without having to restrict the developers' use of SDS.

Solutions

We decided to change the method that `nschedv` uses to calculate the priority of swapped processes, normally referred to as swap priority. Normally, the formula uses the process priority *p_upri* as a variable in the calculation. We replaced *p_upri* with the Fair Share variable *kl_usage* which is relatively static and is an accurate indicator of Fair Share priority.

The results of this change essentially gave the production jobs “dedicated” memory due to their 99% share of the resources. When the c90 is oversubscribed for memory, production processes stay resident in memory and development processes compete with each other for the available memory or remain swapped until enough memory is available.

After this change was implemented, we were still seeing production jobs swapped that were doing network I/O. The *hard sleeper* status was overriding our new calculation of swap priority and defeating the purpose of the new `nschedv` calculation. A two-line mod to the scheduler was made to change the *hard sleeper* threshold to 27 for all non-root processes. This allows non-root processes that are doing network I/O to remain in memory, but allows system daemons that are inactive for long periods of time to get swapped. After this mod was applied, the production jobs doing network I/O were not marked as *hard sleepers* and were not swapped.

As mentioned earlier, we noticed that there was a miscalculation in the amount of available hog memory. We discovered that when the `vfork(2)` system call was called by a hog process, the new child was incorrectly marked as a hog process. Then *hogmem*, the amount of memory allocated to hog processes, was incremented to account for the new hog process. We corrected `vfork` so that child processes of hogs are not marked as hogs. An SPR was filed with CRI for this problem.

Unresolved Issues

The SDS space issues are still not resolved to our satisfaction. It seems to be impossible to get the operating system to release SDS space once it has allocated it to a process, especially on a priority basis. This issue has not impacted our production run yet because most development jobs use a small amount of SDS. We will most likely end up reserving SDS for production jobs and allowing development users limited access.

Summary

In an effort to allow higher priority operational jobs to receive resources on demand, yet still allow lower priority development jobs to utilize remaining resources, we modified the memory scheduling algorithm to compute swap priorities using a process priority value that was stable and accurately reflected Fair Share priorities. We changed the definition of *hard sleeper* to not include user processes doing network I/O. We also fixed a small bug in `vfork` that was

miscalculating *hogmem*. We feel that, for the most part, our modifications have been successful and our goals have been met. We are still looking for a better solution to SDS space allocation. As the job mix changes towards more operational jobs, we will have to modify our scheduling parameters, but we now feel that we have a better idea on how to tune the scheduler and have more control over the overall job load. If there is further interest in the modifications we made, please contact the authors via the email addresses listed below.

Acknowledgments

Thanks goes to the Systems Support Division at FNMOC for their support of this experiment and to CRI for allowing us to utilize the considerable talents of Chris Brady to help us understand the scheduler.

Author Information

Chris Brady is the AIC with Cray Research, Inc. at the National Center for Atmospheric Research in Boulder, CO. He also has a temporary assignment in Software Product Support division for Cray Research, Inc. He can be reached at cbrady@denver.cray.com.

Don Thorp is the AIC with Cray Research, Inc. at the Fleet Numerical Meteorology and Oceanography Center in Monterey, CA. He can be reached at djt@craywr.cray.com.

Jeff Pack is a system analyst with Grumman Data Systems at the Fleet Numerical Meteorology and Oceanography Center in Monterey, CA. He works on workstation, network and supercomputer administration. He can be reached at jpack@fnoc.navy.mil.

References

Cray Research, Inc., *UNICOS System Administration for Source Releases*, SG-2113 7.0, 1992.

Cray Research, Inc., *UNICOS Tuning Guide*, SR-2099 7.0, 1993.

PLANNING AND CONDUCTING A UNICOS OPERATING SYSTEM UPGRADE

Mary Ann Ciuffini

National Center for Atmospheric Research
Scientific Computing Division
Boulder, Colorado

Abstract

For large UNICOS-licensed sites that support many local mods and local codes, a UNICOS operating system upgrade is a major undertaking. A detailed plan, flexible time line, and communication with staff, Cray Research Incorporated (CRI) analysts, third party vendors and users are all valuable aids to accomplishing a successful and painless upgrade. Often the reorganization of file systems is combined with an upgrade. This paper will present the methodology used at the National Center for Atmospheric Research (NCAR) for planning and conducting UNICOS operating system upgrades.

Introduction

All sites must periodically upgrade their computers' operating systems in order to support new hardware and software products, to obtain software enhancements, and to apply bug fixes. In fact, since Cray Research's Software Release Policy was put into place in April 1992, fixes are now provided in an Update to the latest Revision or in the next Revision. This means that sites which formerly applied individual CRI mods to fix known problems and then rebuilt just the affected codes, must now apply an upgrade and rebuild the entire system. However, in Software Field Notice (SFN) #1072 dated February 11, 1994, CRI resumed listing individual mods. Listing individual mods in SFNs for 7.0 is something that CRI had stopped doing. Since CRI has not announced a change to their 1992 Software Release Policy, it is not clear if CRI is now releasing individual mods in addition to the

revisions or updates.

Each Major Operating System release level, according to CRI's Software Release Policy, is supported for one year beyond the introduction of the next Major release level. Major Operating System releases are 12 to 18 months apart. Consequently, many sites will be conducting a major operating system upgrade every year to year-and-a-half.

For computer installations that administer several Crays of varying architectures and configurations, that have applied local mods to the UNICOS source code, and that support locally developed codes and third party vendor products, upgrading the UNICOS operating system is not a trivial task. The development of a detailed plan and a flexible time line can serve to facilitate this process, as can the creation of an upgrade team.

Materials

One of the first things to be done when upgrading to a Major Operating System release level is to check all licensing requirements. At large computer centers changes to existing license agreements or the purchase of new licenses must go through the center's contracting department. This process can be time consuming, so it is wise to initiate it well ahead of the planned upgrade. UNICOS licensing requirements are listed in CRI's *Release Preview*.

UNICOS documentation that should be procured and carefully reviewed before an upgrade plan and time line can be fully developed are: *Release Preview*, *Release Overview*, *Release Letter*, *Update System Installation Addendum*, *Installation Guide*, *Errata*, and *Publications Errata*. The *UNICOS System Administration Manual* should be available. If a Major Release is being installed, the full set of UNICOS documentation for the release level should be ordered for the systems staff, the consulting staff and the documentation library.

Tapes for the release, updates, revisions and asynchronous products must be ordered. If CRI has recommended that the operating system (OS) on the Operator Workstation (OWS) should be upgraded, the tapes and documentation for accomplishing this must also be ordered.

Component Identification

All major components, CRI, locally developed and third party vendor supplied, that must be rebuilt and/or tested, should be identified. Examples at NCAR include: Network Queueing System (NQS), Network File System (NFS), I/O Subsystem (IOS), OWS, file system quotas, networking, mass storage access, NCAR Graphics, climate models, local and third party libraries. For local, public domain and third party applications and libraries the make files, source code and man pages need to be available. Establishing a

well documented and organized source code directory tree structure for local commands and libraries can greatly facilitate the build and install of non-UNICOS components.

The "Software Enhancements" and "Compatibilities and Differences" sections of the *Release Preview* and *Release Overview* should be carefully scrutinized by the systems staff and consultants for new features that will need to be tested and for differences that will impact users. Enhancements and differences that are determined to affect the user community should be advertised to them well in advance of the cutover to the new OS. At NCAR we created a file under the directory `/usr/news` to communicate this information. As new upgrade issues arose they were appended to this news file. Additionally, users were able to retrieve the *Release Overview* with the on-line documentation retrieval system, Docview.

Create the Upgrade Team

Once the components have been identified, the persons responsible for building and/or testing them comprise the upgrade team. The upgrade team includes systems staff, consultants, operators, CRI analysts and end users. The upgrade team is vital to the success of the project.

Each team member should be notified of the components he or she will be expected to build and/or test. To facilitate communication amongst the team members an e-mail alias can be created and the project head should be identified. All communications should be bidirectional between the project head and the team.

Time Line Development

Before the upgrade materials are obtained a crude time line can serve to get the upgrade project started in the right direction. Once the manuals are in-hand and the various components and their

points-of-contact have been identified, the time line can be refined. The time line should show all major tasks and their dependencies. It should also show tasks that can occur concurrently. By each task box the personnel involved in performing that task should be listed. Each task box may also be labeled with a start and end date, and the duration in days.

The time line should be flexible. Its main purpose is to keep the project on track and to ensure that major tasks are not omitted. Each member of the upgrade team should receive a copy of the time line and should be allowed to suggest necessary modifications to it.

File System Layout

The load, build and install of the new release can occur while the host is running the production system. To accomplish this, separate root, usr and src file systems for the new release are needed. CRI also recommends a tmp file system of 50,000 blocks. However, if disk space is tight, tmp can be a directory on the root file system. We have found that a /tmp directory is sufficient if the build is done in stages and /tmp/OUT is cleaned out after each stage.

Some weeks before the upgrade, monitor and log the disk space usage and I/O rates of the current production system's file systems. This information will show if current file system sizes and placement are adequate or if adjustments need to be made. CRI supplies disk space requirements in the *UNICOS Installation Guide* but we have found their sizes to be overly generous. A method that can be used to determine file system size is to compute the percentage increase between CRI's disk space requirements for the currently running release and for the new release. Increase file systems from their current adequate size by this percentage. As an example, if it is determined that an adequate size for the production root file system is 90,000 blocks and the percentage increase is 10%, the root file system for

the new release should be 99,000 blocks.

If disk space for the upgrade is limited and there is available space on another host, such as a workstation, the current UNICOS source can be moved and NFS mounted. The source file system on the Cray disks can then be reused for the new release or upgrade. If this option is selected, the decision to freeze changes to the current source, except for critical bug fixes, should be made.

Once the file systems sizes and layout on disk have been determined, develop a detailed checklist of the steps needed to create the new file systems layout and develop scripts to perform the tasks. A checklist will ensure that tasks are performed in order and are not omitted. Several hours of system test time will be required to revamp the file systems. Use of logical device caching (ldcache), use of the dd command, and having planned for no more than one move per file system helps to reduce the time required. This time savings could be as much as fifty percent. An example of using the ldcache, dump and restore commands to copy file system \$1 to file system \$2 follows:

```
mount /dev/dsk/$2 /mnt
ldcache -n 500 -s 96 -t SSD -l /dev/dsk/$1
ldcache -n 500 -s 96 -t SSD -l /dev/dsk/$2
cd /mnt
dump -t 0 -f - /dev/dsk/$1 | restore -x -f -
ldcache -n 0 -l /dev/dsk/$1
ldcache -n 0 -l /dev/dsk/$2
cd /
umount /mnt
fsck -u /dev/dsk/$2
```

When using the dd command, the file systems must be the same size. The dd command is particularly useful for creating backup root and usr file system. Here is an example of how the dd command can be used to copy file system \$1 to file system \$2:

```
dd if=/dev/dsk/$1 of=/dev/dsk/$2 bs=409600
labelit /dev/dsk/$2 $2 sn1036
```

Keep in mind that the main goals for remaking file systems are to achieve optimal sizes and I/O performance.

Backups

Identify critical times when full system backups should be executed. Before a load and before a cutover are examples. After the new release is built, include it in the daily system backup. At NCAR a locally developed Practical Extraction and Report Language (PERL) script backs up the software on the Y-MPs to NCAR's Mass Storage System (MSS). Incremental system backups are executed six days per week and a full system backup is executed once per week. The MSS is comprised of an IBM 3090, a 120 gigabyte IBM 3380 disk farm, a 4.8 terabyte Storage Tek Automated Cartridge System and an IBM 3480 cartridge system. Access to NCAR's MSS from the Crays is via the ANSI High Performance Parallel Interface (HIPPI) and the Network Systems Corporation (NSC) HYPERchannel.

Test Schedule

Production down time is only required for file systems reconfiguration, system testing and cut-over. At NCAR, we try to advertise production system down time one week in advance and schedule it during our normal system test periods, 0600-0830. The agenda for each test and a list of the upgrade team members needed to conduct the test should be prepared in advance. Each test of the new release should be well planned to avoid wasting expensive production time. After each system checkout, the test team should share their findings which may lead to problem resolution and follow up testing.

Load

If sufficient space is not available to build and test the new release, the file systems will need to be reconfigured. For a new release, the base root

and usr are loaded first. Source, products, revisions and updates are loaded next. If an update release is applied to the current UNICOS source, local mods and CRI mods may have to be deleted first. Having a list of the mods that comprise the upgrade and the codes they affect, facilitates this. Before adding the UNICOS source for a new release make sure that local mods and CRI mods that were added to the current production system do not get overwritten.

Mods

We have added approximately forty-five local mods to the production base release. More than half of these local mods pertain to system accounting. NCAR local mods and CRI mods that have been added to the base release are logged and stored in the standard UNICOS source .USM directories and in a separate directory tree structure that we have named /usr/src/local. Having copies of these mods and logs stored in a separate directory tree makes it easier to determine how the base release was altered. It facilitates the process of deciding which local mods need to be applied to the new release. Should the UNICOS source file system be reused for the new release, having the local mods stored under a separate directory tree makes it easier to move them to an area where they will not be overwritten by the new release. We have developed local scripts "addmod/delmod" that accomplish this. The addmod script adds mods to the UNICOS source tree via the UNICOS Source Manager (USM). It then places a copy of the mods under a tree structure separate from the UNICOS source tree and updates the log files located under this tree. It then mails an informative message to the systems staff and appends this message to a log of system changes.

Once the new release is loaded, each local mod must be reviewed for compatibility with the release. Some local mods may need to be rewritten before they can be added. Others may no

longer be valid. CRI mods that had been added to the current base release should be in the new release but it should be verified that this is the case.

Configuration

The configuration information for the machine and site should be updated via the install tool. Files that the install tool converts, imports, creates or updates should be identified. After the configuration process has been activated via the install tool, verify that the files have been changed correctly. There usually are a few configuration files that slip through the cracks, i.e. `ntp.conf`. These will have to be identified and converted manually. Review configuration changes that were made directly to files, i.e. files that were not modified by applying USM mods, and make sure that these changes are applied to the files on the new system. All changes made to system files on NCAR's Crays are logged and communicated to the systems staff via e-mail. This local change log is an extremely useful reference.

Build and Install

Build and install the new system under `chroot`. Review all configuration files to ensure that they are set up properly for the site and machine and that they have not been inadvertently overwritten by the build/install process. If required, upgrade the operating system on the Operator Workstation. As a precaution, the current OS on the OWS should be backed up to tape just prior to the upgrade. Transfer the new IOS and UNICOS kernels to the OWS.

Test Environment

Before booting the new system, a test

environment should be set up. Ensure that the correct file systems will be mounted and that production system files, such as accounting records, will not be modified while the test system is running. Step through the startup scripts and check files, such as `crontabs`, to verify that the appropriate components get started. The first test of the new system should be conducted by the systems staff to ensure that the system boots and transitions into multi-user mode properly. Major UNICOS components, such as NQS, should be tested. The user data base (UDB) will need to be populated. If the first test is a success, subsequent tests will involve the other members of the test team.

Access during system testing should be restricted to members of the test team. We have developed a local modification that enables us to control access to the system via setting the `sitebit` in the UDB. If the `sitebit` setting for a user matches the value that is entered in the file `/etc/restrict`, that user is allowed to logon. To open access to all users, the `/etc/restrict` file is removed. Users whose access to the system is blocked can be notified that the system is in restricted mode by placing an informative message in `/etc/issue`. When the user attempts to logon the message in `/etc/issue` is echoed. We have found this locally developed procedure for restricting system access to be quicker and more flexible than the UNICOS command `udbrstrict`.

Non-UNICOS components that are built under the test system must be kept separate from their production system counterparts. This is particularly important if the `/usr/local` file system, home to the non-UNICOS utilities, is shared by both the test and production system. Through the use of mirror directories on the `/usr/local` file system, these binaries can be kept separate. Scripts executed prior to system startup and after system shutdown can handle the renaming of the mirror directories appropriately for the production or test system. If space is an issue and all files can not be duplicated, codes that do not need to be rebuilt under the new release, such as scripts, can be linked instead of copied to the mirror

directories. If hard rather than soft links are used, at cutover linked files under the old directories can simply be removed.

When switching to the test system from the production system and visa versa, instead of entering the same commands over and over again, put them in scripts. Setup scripts can simplify the transition between production and test mode. They reduce the number of key strokes at the console and thus minimize errors and save time. Setup scripts help to avoid omissions and serve as a record. As more components are built and tested under the new system, they can be added to the setup scripts.

Cutover

At cutover to the new UNICOS release there should be no major surprises as long as the new system had been thoroughly checked out. CRI enhancements, local codes, network connections, user codes and etc., all should have been tested under the new UNICOS release prior to cutover. The user community should have been notified well in advance of the changes that will affect them and at this point their scripts and codes should have been modified accordingly. Also, the current production system should have been backed up as a precaution.

Sites that use the batch system extensively will probably want to move the work load from the current production system to the new system. To do this all NQS jobs must be put into a queued state. As long as there have been no changes to internal NQS structures, the current NQS information under `/usr/spool/nqs` can be copied to the new system. CRI has provided "qdump" and "qload" to accomplish this in the UNICOS 8.0 Release. The entire procedure is defined in the *UNICOS 8.0 Installation Guide*. In the UNICOS 7.0 Release, the internal NQS (raw request) structures changed. CRI supplied a "qconvert" utility to convert queued NQS jobs from 6.1 to 7.0.

For sites that require system accounting information, quiet the production system and run a final accounting. We have found that running a final accounting on the production system is easier than converting accounting records over to the new system. At this point time critical files, such as the UDB, should be moved to the new system. Before bringing the new system up in multi-user mode the UDB must be updated. Finally, changes should be made on the test system to convert the test environment to a production environment. For example, local utilities that were rebuilt under the test system should become the default, permanently replacing their previous release counterparts.

Once again it may be necessary to reconfigure file systems. Using the `dd` command or the `dump` and `restore` commands, the new root and `usr` file systems should be copied to backup root and `usr` file systems. On several occasions, we have had to rely on the backup root to boot our production system. If `/usr/adm` and `/usr/spool` were part of the `usr` file system under the test system but were separate file systems under the production system, they will need to be split from `usr` at this time.

Recommendations for CRI

CRI's April 1992 Software Release Policy of not releasing individual mod fixes is not acceptable. For most sites applying a revision or update and testing it is a major task, while adding one or two mods to fix a bug is not. SFN #1072, which lists individual mods, may indicate that CRI has recently reversed their 1992 policy. If this is the case, CRI needs to officially announce this change in their Software Release Policy.

Due to the internal NQS (raw request) structures change in UNICOS 7.0, CRI supplied the `qconvert` utility to convert queued NQS jobs from 6.1 to 7.0. Should CRI make changes to NQS in the future that would affect transferring the NQS queued workload to the new release, we request

that CRI continue to supply a qconvert utility. Furthermore, we would like CRI to provide a method for transferring NQS checkpointed jobs from one release to another. At NCAR there are a great number of long running batch jobs. To get running jobs into a queued state so they can be transferred to the new release, they must be rerun. This wastes the system resources that had already been accumulated by these jobs.

CRI used to list all the mods in an upgrade package. We would like them to continue this as it helps us to decide which local mods need to be pulled before the upgrade mod set can be applied.

We would like CRI to provide an install tool and source release that can be used on one Cray platform to build systems for other Cray platforms. This would reduce disk space requirements for source and would eliminate the need to have a source license for each Cray host. Local mods could then be applied to Cray machines that are not licensed for source.

Future Directions

On NCAR's two Y-MP super computers, the UNICOS upgrades have been conducted separately and at different times from source located on each machine. With the recent acquisition of a Cray EL, the concept of treating the three Cray hosts as a cluster has emerged. For this concept to work, all NCAR Crays must run the same version of the UNICOS operating system. This means that OS upgrades must be done concurrently. Most of the UNICOS source for the Y-MPs and EL is the same. Consequently we are planning to store the source and build UNICOS on one Cray host for the other Cray hosts. Machine specific source, such as the UNICOS and IOS kernels, will need to be stored and built separately from the common source. If this scheme works, system build/install and test time should be reduced significantly. Disk space usage by source code should also be reduced.

Conclusion

A painless and successful UNICOS upgrade can only happen with careful preparation and planning. The adage "Haste makes waste" certainly applies here. There are many items to attend to in a UNICOS upgrade. The project leader must pay attention to detail and must be highly organized. The skills of staff and users should be drawn upon, especially when testing the new system. Communication with project team members and end users is vital. There should be no surprises for anyone.

References

1. Cray Research Service Bulletin, Volume 2, Number 5, May 1992
 2. UNICOS 7.0 Release Preview, PV-5000 7.0, Cray Research, Inc., 1991
 3. UNICOS 7.0 Release Overview, RO-5000 7.0, Cray Research, Inc., 1992
 4. UNICOS Installation Guide, SG-2112 7.0, Cray Research, Inc., 1992
 5. Frisch, Aeleen, *Essential System Administration*, O'Reilly and Associates, Inc., Sebastopol, Ca., 1991
-

UNICOS Release Plans (1994-1995)

Janet M. Lebens
UNICOS Release Project Leader
Cray Research, Inc.

March, 1994

ABSTRACT

This paper gives an overview of recent and upcoming UNICOS releases. Specific focus is placed on the basic feature content and timing of UNICOS 7.0, 8.0, 8.x and 9.0 release levels for 1994 and into 1995.

1 Introduction

This paper is divided into three parts. The UNICOS Release Policy section briefly covers the UNICOS Release Policy adopted in April of 1992, describes the different release types provided under the policy, and explains, based on the policy, what releases are planned for 1994 and the first part of 1995. The Release Content section covers the content and release dates for recent and upcoming releases for the UNICOS 7.0, 8.0, 9.0 and 8.x (the next Restricted Feature Release) levels. The paper concludes with an overall summary.

2 UNICOS Release Policy

2.1 Release Types

The Software Release Policy was adopted in April of 1992. There are two release levels provided for under the policy: Major releases, and Restricted Feature releases. Both of these can be further divided into Revisions and Updates. Each of these release types has a different purpose, a different frequency, and a different level of support.

2.1.1 Major Releases

Major releases are intended for all customers. They focus on providing new functionality as well as fixes. Major operating system releases are 12 to 18 months apart, and are released in the first or third quarter. Each Major release is supported for one year beyond the introduction of the next Major release level to provide sites with sufficient time for upgrading to the next Major release. Both source and binary systems are supported with Major releases.

2.1.2 Restricted Feature Releases

Restricted Feature releases are available on the latest architectures to satisfy contractual commitments for customers who need new features before the next major release. In the past, they have been released on an as-needed basis with no set frequency. In the future, CRI sees an ongoing need to satisfy contractual com-

mitments. As a result, starting with 8.x, Restricted Feature releases will occur every 15-20 weeks. Support is limited for the Restricted Feature release: a UNICOS source license is required, and support for the Restricted Release ends at the next Major release of the product.

2.1.3 Revision and Update Releases

Revisions and Updates primarily contain fixes. Priority is given to fix critical and urgent problems. There are two to four Revisions per year per product level, continuing throughout the support period and decreasing in frequency over the life of the release. Major-release Revisions and Updates contain features by rare exception and only when those features are in broad demand and do not impact other customers' operations.

Updates are a snapshot of the next Revision under development. They are the fastest release method and smallest, tested releasable package. They provide a delivery mechanism for fast response to critical problems and Software Field Notices. Updates are released every 4-6 weeks for the Major release level. Both Revisions and Updates are available for source and binary systems for the Major release level.

2.2 Policy Layout 1994-1995

Release Schedule		SCHEDULE 1994-1995							
		1994				1995			
		1	2	3	4	1	2	3	4
7.0					7.0.7 *				
7.C			*						
8.0	8.0.2				8.0.3		8.0.4		
8.x		8.1	8.2			8.3			
9.0									9.0

* = support ends

UNICOS 8.0 is the current Major release level. Revisions to UNICOS 8.0 will occur approximately every 4 months into 1995. As the release level matures, the need for Revisions will decrease and so will their frequency. Updates will occur every 4-6 weeks for UNICOS 8.0.

UNICOS 7.0 is the previous Major release. Because UNICOS 8.0 was released in March of this year, support for UNICOS 7.0 will end in March of 1995. One more revision is planned for 7.0. Updates will continue every 4-6 weeks for 7.0 until support ends.

UNICOS 7.C is the Restricted Feature release level. According to the Release Policy, support for UNICOS 7.C should have ended with the release of UNICOS 8.0. However, support has been extended an additional six months to allow time for sites to upgrade to UNICOS 8.0. Please upgrade as soon as you can; in the interim, CRI will be providing fixes to critical and selected urgent problems.

UNICOS 8.x will be the next Restricted Feature release level, which will be a superset of UNICOS 8.0. CRI is planning three releases, 8.1, 8.2 and 8.3, for this release level. These releases will be similar to UNICOS 7.C with one main difference: rather than being a separate release leg, this leg will become UNICOS 9.0. In other words, features for UNICOS 9.0 will be incrementally integrated into the 8.x releases, about one third of UNICOS 9.0 features into each 8.x release. As a result, there will be more features in Restricted Feature releases than in the past. Also, upgrading to UNICOS 9.0 will be easier.

3 Release Content

3.1 UNICOS 7.0 Major Release Level

3.1.1 UNICOS 7.0.7 Revision Release

One more revision to the 7.0 leg is planned in fourth quarter of this year. UNICOS 7.0.7 will continue focus

on reliability, mainly in the way of critical and urgent fixes.

In addition to focusing on reliability, UNICOS 7.0.7 will contain support for the UNICOS under UNICOS feature. This feature allows you to run two copies of UNICOS on a single mainframe. UNICOS under UNICOS supports YMP and C90 mainframe systems. The intent of providing UNICOS under UNICOS is to allow sites to test new versions of UNICOS prior to upgrading.

In order to support this feature, both versions of UNICOS must contain the UNICOS under UNICOS code. UNICOS under UNICOS support will also be contained in UNICOS 8.0.4 and the UNICOS 8.2 Restricted Feature release. Therefore, any combination of 7.0.7/8.0.4/8.2 or later releases in each of these legs will be able to be run together. Upcoming issues of the Cray Research Service Bulletin will contain more information on this feature.

3.2 UNICOS 8.0 Major Release Level

3.2.1 UNICOS 8.0 Major Release

UNICOS 8.0 was released in March of this year. There are five major themes for this release: Robustness and Resiliency, Performance, Security, Standards, and New Hardware. Significant functionality for UNICOS 8.0 will now be discussed as they relate to these five themes.

3.2.1.1 Robustness and Resiliency

There are many features in the area of Robustness and Resiliency.

Checkpoint/restart has been improved, including NQS automatic checkpointing jobs at regular intervals, which allows you to restart NQS requests after an unplanned system outage.

UNICOS 8.0 provides interactive session protection, by giving you the ability to reconnect to an interactive session in the event of a network interruption.

Improvements have been made to the tape daemon for 8.0 surrounding tape daemon aborts, as well as in the area of kernel memory management and error handling.

A mechanism is provided to automatically switch to an alternate path to disk when one is available and the system has detected a hard failure on the currently active path.

Automatic power on/off allows you to power on/off your CRAY C90 system without corrupting data or losing work in progress.

3.2.1.2 Performance

In the area of performance, the UNICOS kernel has been multithreaded to reduce semaphore wait time. The result is a reduction in system overhead and system idle time for multiple-CPU systems. Those with heavily utilized systems should see a substantial improvement in system throughput as a result.

Dynamic tuning options for autotasked codes automatically adjust to changes in system workload. This includes dynamic processor scheduling based on system load average.

UNICOS 8.0 increases NFS performance by providing an option for the server to do asynchronous I/O to disk and by providing client control over invalidating its cache entries when a file is closed. Subsequent reads can then be satisfied out of the cache rather than over the network.

3.2.1.3 Security

Many features have been added for UNICOS 8.0 to support a B1 evaluation of the UNICOS system. One particular configuration of Multilevel Security is known as "Trusted UNICOS". The UNICOS 8.0 release version of Trusted UNICOS has been given a B1 MDIA rating by the US Department of Defense.

3.2.1.4 Standards

With the exception of a few commands like `lex` and `awk` on the CRAY-2 and internationalization, UNICOS 8.0 implements POSIX P1003.2, the "shells and utilities" standard. Full compliance is planned for UNICOS 9.0.

To provide the advantages of full ANSI C prototyping to all users, full function prototypes are on in all UNICOS 8.0 include files. ANSI C prototypes permit the compiler to diagnose errors at compile time.

Vnodes replace the current file system switch to allow for easier importing of 3rd party file systems, such as Distributed File System (DFS) for the Distributed Computing Environment (DCE).

3.2.1.5 Hardware

In the hardware area, UNICOS 8.0 supports all CRAY M90, C90 and EL90 series products. Underlying UNICOS support for the CRAY T3D is included.

Disk array support for the DA-60/62 and the new DA-301 drive are provided, as well as disk support for the DD-301 disk, and the CRIND-12/ND14 HiPPI disk array products. Support for the DD5 for the EL is also provided. RAID 10 provides disk striping and disk mirroring across multiple IOSs, providing scalability, fault tolerance, and performance improvements.

In the area of tapes, the EMASS ER90 helical scan tape and associated tape autoloader, known as the Data-

tower, are supported. The 3490, a drive similar to the 3480 but with IDRC (Improved Data Recording Capacity) (i.e., data compression), is supported, as well as the 3490E, a 36 track medium with double-length tape capability. Digital Audio Tape support was also added for the EL.

For model E platforms, support has been added for the SSD-E 128i. These are the SSDs with 128MW of memory and are included in the model E mainframe cabinet.

In the channel area, support for the FCA-1, the channel adaptor connected directly to FDDI for model E systems, is included.

3.2.1.6 Other Significant Features

Much work was done for 8.0 in the way of user exits. Exits were added for security, USCP, tapes, NQS, accounting, as well as a site-reserved system call for customer's own use.

The Unified Resource Manager facility is a centralized workload manager that dynamically adjusts to changing system workloads, based on input from NQS, telnet, ftp and others, as well as resource targets established by the system administrator.

An asynchronous multiplexed swapper increases interactive user response times by concurrently swapping out multiple processes.

UNICOS 8.0 was field tested at six sites, to test a combination of hardware platforms (XMP, YMP, EL and C90) and major software functionality (MLS, Multithreading, DMF, tapes and binary). The results were good from CRI's perspective: better stability and fewer SPRs than field tests for 7.0.

3.2.1.7 Compatibility/Support Issues

UNICOS 8.0 will be the last major release for the CRAY-2 and XMP platforms. These platforms will continue to be supported through 1996 by UNICOS 8.0 Revisions and Updates.

To implement POSIX 1003.2, we changed the default shell. It is based on `ksh`, which executes nearly all Bourne shell scripts unchanged. See the UNICOS 8.0 Release Overview for the list of changes associated with POSIX 1003.2. Some libraries and utilities, such as `libm` and `cdbx`, will now be released with the programming environment releases such as FORTRAN and C.

3.2.2 UNICOS 8.0.3 Revision Release

UNICOS 8.0.3 is planned for fourth quarter 1994 release. UNICOS 8.0.3 will include SPR fixes as well as a fair amount of new functionality.

UNICOS 8.0.3 will contain support for T3D Phase II I/O. With Phase II I/O control for I/O is still done by the Cray mainframe, but data transfer can occur directly between disks and the T3D.

Optimizations have been made to buffer cache management routines. These enhancements have show a significant speed-up.

In addition, enhancements have been made to the file-system allocation routines. The filesystem changes have demonstrated significant speed-ups in adding data to long files, because the routines now keep track of the last extent to a file, rather than searching for the end of a file.

A new product, NQX 1.0, will make available the following NQX components on UNICOS 8 systems: The Network Load Balance, NLB collector and associated libraries, NQS interface to the Network Load Balancer, NQS server support, and NQE clients. With the addition of NQX, UNICOS systems can now be part of the NQE batch complex.

FTA 5.0 will add support for IBM's MVS ftp server. BLOCKMODE support is also being added to FTA, to allow FTA to transfer Cray Block format files to an IBM MVS ftp daemon. The quote command provides the ability to send any unsupported or vendor specific ftp commands through FTA and onto the remote ftp daemon. FTA 5.0 will also include performance enhancements, which will keep FTA current with improvements in CRI's ftp and that of the industry. More platforms will also be supported in release FTA 5.0.

UNICOS 8.0.3 contains the hooks for the OSF/Distributed File System. DFS will be a separately released product from UNICOS and is planned for 2H94; please see Brian Gaffey's paper in these proceedings for more information on DFS.

The IPI-3/HiPPI protocol will be supported for the EL in UNICOS 8.0.3, allowing the EL to connect to network disks such as the Cray ND12/ND14 disk arrays. The EL will be able to sustain speeds up to 50 MB/sec to these disks.

IBM's automated tape loader, the 3495/3494, will supported in UNICOS 8.0.3.

3.2.3 *UNICOS 8.0.4 Revision Release*

UNICOS 8.0.4 is planned for a second quarter 1995 release. In addition to fixes, this release will contain support for one configuration, the CRAY J916, of CRI's follow-on EL product, the CRAY J90 series. It is YMP compatible, and has a 10 ns clock rate.

3.3 *UNICOS 8.x Restricted Feature Release Level*

3.3.1 *UNICOS 8.1 Restricted Feature Release*

CRI does not plan to deliver on any hardware or software commitments in UNICOS 8.1, but rather is using this release to demonstrate that they have a process which allows integration and delivery of one third of all UNICOS 9.0 features into a Restricted Feature release while still providing a solid, stable release. Its planned completion is third quarter of 1994.

Assuming UNICOS 8.1 is successful, UNICOS 8.2 is planned for third quarter 1994 release. Among the commitments to be delivered via this release are Escon channel support and initial Cray Triton system support.

The Escon channel will allow for a greater distance between Cray and tape unit (up to 3 kilometers, theoretically), and will have a higher I/O bandwidth than currently possible with the block mux channel. Most newer tape products, such as the STK Redwood and IBM Ntp products discussed below, will probably require this channel. The Escon channel requires an FCA-2 channel adaptor.

Initial support for Triton, the follow-on to the C90 product on the parallel vector side, will be included in UNICOS 8.2. Full support for basic Triton will be provided in the UNICOS 8.3 release, planned for first quarter 1995 deliver.

3.4 *UNICOS 9.0 Major Release Level*

CRI's overall emphasis in the UNICOS 9.0 timeframe is Open Supercomputing. Computing environments are becoming more heterogenous; in other words, including a variety of unlike hardware and software components, each of which efficiently performs a different computing task. CRI is defining and developing Open Supercomputing as an environment in which Cray Research tools can be easily integrated in a variety of combinations, and can effectively cooperate to solve problems.

3.4.1 *UNICOS 9.0 Major Release*

UNICOS 9.0 is planned for third quarter 1995 release. In addition to this overall emphasis, five major themes have been chosen for UNICOS 9.0: New Platforms, Additional Standards, Heterogeneous Computing, Reliability, Availability and Serviceability, and Additional Peripherals. Content for 9.0 is still firming up, but this paper will highlight some of the major features currently planned for UNICOS 9.0.

3.4.1.1 *New Platforms*

Support for basic CRI Triton systems will be part of UNICOS 9.0. CRI IEEE Triton system support will follow in UNICOS 10.0, and will also be added as a part of one of the UNICOS 9.x releases.

IOS-F is our follow-on product to the IOS-E, and we will be using it to support new peripheral and network technologies, such as ANSI Fiber Channel and ATM.

3.4.1.2 *Additional Standards*

In the area of standards, 3 components to the ATM software will be included in UNICOS 9.0: a signaling protocol, based on UNI 3.0, an ARP server, and a convergence layer.

For the past five years, CRI has focused its operating systems standards participation on POSIX. In August of last year we certified compliance with the 1988 POSIX 1003.1 for UNICOS. CRI has also committed to POSIX 1003.2. A survey of customers and potential customers revealed that, for the future, the requirement is for X/Open branding for UNICOS. Therefore, in the 9.0 timeframe, CRI plans to obtain X/Open branding for XPG4 compliance.

UNICOS 9.0 will include ONC+, the follow-on to ONC (Open Network Computing) developed by Sun. The components of ONC which will be upgraded from their ONC versions will include NFS V3 (in the transparent file access area), NIS+ (in directory services), AUTH_KERB (in RPC authentication), and LockManager V3 (for file locking).

3.4.1.3 *Heterogeneous Computing*

CRI knows that heterogeneous computing is important to its customers. CRI will continue to measure and improve cross-platform computing performance in the 9.0 timeframe.

The Shared File System feature provides the ability to share file systems across multiple Cray systems. The attributes of the SFS include high-speed data access, full UNIX file semantics, and shared media controlled by a semaphore arbitration service.

The initial support for SFS for file systems spanning multiple SSDs is included in UNICOS 8.0. UNICOS 9.0 will complete this feature, and will include the support of network disks devices such as the CRI ND-12/ND-14 disks across high-speed HiPPI networks.

3.4.1.4 *Reliability, Availability, Serviceability*

Support for UNICOS under UNICOS will help make upgrading from UNICOS 8.0 to UNICOS 9.0 easier.

A dynamic kernel memory allocator will be available in UNICOS 9.0. A dynamic kernel memory allocator allows for centralized memory allocation management. Before this feature, if a kernel subsystem needed temporary or static space, the space was allocated from buffer cache or m-bufs. In most cases this imposed

problems on systems that relied on space in the buffer cache or m-buf being available. This feature will also makes maintenance and porting of UNICOS features to the new microkernel technology easier.

The checkpoint interface will be re-engineered for 9.0. These changes should help to alleviate some of the problems inherent in the previous design. The new design should be more resilient to changes in kernel data structures. The new version of checkpoint will recognize both old and new format checkpoint files. Additional work is also being done to allow checkpoint files to be moved or copied to secondary storage.

Checkpoint/restart enhancements will also include the ability to checkpoint/restart a tape job at the job level. This feature should have great benefit to those of you with long running jobs with lots of tape mounts.

3.4.1.5 *Additional Peripherals*

The Escon channel will be supported in UNICOS 9.0.

The Redwood tape device is the STK helical scan tape product. It will be supported in UNICOS 9.0, assuming availability of the hardware in time.

IBM's new tape product is also planned for UNICOS 9.0. It is the follow-on to the 3490E, and. uses a new cartridge media, which has 144 tracks of data, and eight times the storage space of a 3490E.

3.4.1.6 *Other Significant Features*

Another significant feature is the optimization of the tape daemon. Currently, the tape daemon forks a child process many times during the life of a tape request (from initial request until it is released). This project will fork one child process per tape request, and that child will handle all of the other tasks without additional fork(2) calls. This will cut down the amount of pipe communication and forks and execs, improving performance.

4 Summary

• UNICOS 7.0

With the release of UNICOS 8.0, UNICOS 7.0 becomes the previous Major release. Updates will continue for UNICOS 7.0 throughout its support life. CRI plans one more Revision to UNICOS 7.0. The emphasis for UNICOS 7.0. continues to be reliability, and 7.0 will be supported through first quarter 1995.

• UNICOS 7.C

Support for UNICOS 7.C will be strictly in the form of critical and urgent fixes. This limited support will end in third quarter.

• UNICOS 8.0

UNICOS 8.0 has just been released. It will be supported with regular Updates, and fairly frequent Revisions. These Revisions will contain some high demand features in addition to reliability fixes.

- UNICOS 8.x

UNICOS 8.x is the next Restricted Feature release leg. Its support will be limited, in a manner very similar to that of UNICOS 7.C.

- UNICOS 9.0

UNICOS 9.0 is our next planned Major release. It is being planned for a third quarter 1995 release.

UNICOS 8.0 EXPERIENCES

Hubert Busch

Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Germany

Introduction

The paper presents our experiences with the new version 8.0 of UNICOS. ZIB was / is one of the UNICOS 8.0 beta-sides, the speciality of ZIB is an old CRAY X-MP 216. I will tell you something about the organization of such a test, about special problems at our side and about some of the new features of UNICOS 8.0.

Situation at ZIB

Today ZIB operates two CRAY computers, a CRAY X-MP and a CRAY Y-MP. In this configuration the old CRAY X-MP operates as a compute server and the newer CRAY Y-MP as a compute server as well as a file server. This gave us the possibility to work with UNICOS 8.0 on the CRAY X-MP while using UNICOS 7.0 on the more powerful CRAY Y-MP. We wanted to test the applications MPGS, CVT, AIT and UniChem and test the system features TCP/IP with *gated*, *named*, *snmpd*, NFS-client, FSS (fair-share scheduling), NQS with FSS, file system quotas and URM (Unified Resource Manager).

Timetable

We started discussion with the German CRAY representatives in June 1993. First we had to sign up a „Field Test Agreement“ with questions like „motivations for field testing“, „test plan overview“, „entrance criteria“, „ZIB field test team“, their duties, „customer representatives duties“, etc.

We needed 4 hours dedicated system time on 2 days for the system people starting on October 6th and used 16 hours on 3 days to generate our own supplements and to test the system. On October 18th we started the „official“ betatest with all of our users.

At November 1st we agreed to finish the field test and to go on with the 8.0 system in full production. The exit criteria of the field test agreement had been met with exception of the availability of the applications UniChem, AIT and CVT. Fortunately, these applications were used for production on the YMP only. We agreed to test these applications and NQS with FSS later. The overall feeling of the field test was: It went better than everybody on site expected and the impact on the users was minimal.

We started with URM on the XMP in January 1994 (see special part of this report) and with UNICOS 8.0 on the YMP, including the former missing applications in February 1994.

Documentation

One of the problems for a customer and his users is the availability of the documentation. From the beginning we had only two draft copies of the important manuals, one for the system people, the other for the application people. There is no paper documentation for the real users. From the beginning we could only use the man-pages in the 8.0-version, which was a great help. The

docview command worked, but it showed the 7.0 documentation. I propose to CRAY to put also the draft documentation of the new software in *docview* format to the betatapes.

Major issues during test period

There were two major issues just before the test period: No routing through a Sun having a FEI3 installed and we had one crash each day because of problems in the TCP code in the kernel. Two days after the occurrence we had a workaround with the help of the people in Eagen.

The problem with the network did reoccur after approximately two hours of UNICOS 8.0. After a boot the channel to the NSC box hangs. Sometimes ping works, but nothing else, sometimes all protocols work, but not so all nets. As a workaround, we configured the interface down (including *netconf -s off*) and up again. *gated* was causing the problems. The *gated* on the XMP scrambles the *hyroute* table entry for the interface from which it receives the last RIP update that changes the kernel routing table. A kernel mod fixed this problem.

We had one crash each day the first days. The problem seemed to be somewhere in the TCP code in the kernel. Watching the quota table usage we usually saw a max usage of approximate 50 out of 250. The last thing we saw on the console screen just before the hang is always an automatic NQS checkpoint. At ZIB the NQS checkpoints go to */tmp/nqschkp*. And */tmp* has *fsquota* on it. We increased the *inode* table size to 1000 and watched the usage by a cron job. We also increased the quota table as a workaround. A fix will be in 8.0.2.

Minor issues

Several problems were caused by the local setting of the *segldr* default to *PRESET=undef*. Executables were built which did not run (e.g. *fpp*, *segldr*, *nupdate*). Several problems when using *mailx*. At the beginning *lpr* did not work.

URM (Unified Resource Manager)

In the UNICOS 8.0 Release Preview CRAY tells us: „URM improves scheduling and performance by centralizing resource allocation with a formal method of communication. This tool is useful in allocating nonbatch usage of Cray systems, with or without NQS.... NQS in UNICOS 8.0 contains many features resulting from customer requests and requirements like... URM interface.... System administrators must decide whether to use URM with NQS to control batch usage of the system.“

We started using URM at the beginning of 1994. We had to modify the user exit according to our needs. We use *urm* to schedule (recommend) all the NQS jobs. On January 17th we activated the new configuration. Although we felt pretty sure about the concept, all the numbers we had to configure gave us some sleepless nights. We switched the NQS scheduling to *urm*

unlimited. We changed our policy in a way that the NQS queues got a totally different meaning. Before URM there were queues with different CPU time and memory limits. Now with URM (and FSS), there are three queues, having basically identical limits, but different interqueue priority and nice values. The queue *fast* is not nice but expensive compared to the queue *normal*, *slow* is quite nice but cheap. „Cheap“ and „expensive“ are in terms of share usage. The price of (not) being nice is set in the array *NiceTicks[nice]* in *limits.c*. The increase with nice is not big enough for the ZIB, so our CRAY site representative had to change the way this table is set up. We want to state that these values should be configured for the site.

At ZIB the users are divided in groups with specific amounts of shares each. Running processes are handled by the FSS accordingly. URM will be used to start only batch jobs that will be given resources by the FSS. URM is not yet used to limit the access otherwise (login, ftp,...), but will think about it.

A job submitted to any queue gets the queue defaults for limits missing in the *qsub* statement. It's values are passed on to URM. URM orders all the requests by using a set of weighting factors and scans for jobs to recommend for running. The result is passed through *site_rank.c*. A template has been provided by Bill Schiefelbein from Cray Eagen which has only slightly been modified to fit local needs.

There will be 5 NQS queues (*ia* stands for interactive users):

Table 1:

	system	ia	fast	bigfile	normal	slow
nice increment	-5	0	+2	+3	+6	+10
relative share usage	+79%	+43%	+29%	+21%	0	-29%
interqueue priority	60	-	45	31	30	15
run limit	9	-	4	1	3	3

A job submitted to NQS will fall in *normal* by default. Jobs submitted to *system* will immediately start to run, but it's access is restricted. For jobs in *fast* the share usage is approximately 29% faster than in *normal*. In *slow*, the chance to run is very low, but the share usage will accumulate approximately 29% slower than in *normal*. An interactive session has a nice increment of 0 and therefore costs approximately 43% more shares compared to a job running in the queue *normal*. All NQS queues (besides *system*) are cheaper and slower than interactive sessions. There are two main issues:

- a) How long does it take for a job to get started?
- b) How fast will it run?

a) is addressed by setting the URM weighting factors, taking into account the share usage (70%), the interqueue priority (20%), and the age (wait time) (10%). All other factors are set to zero.

b) is influenced by the shares, the usage and by the nice value.

The share usage accumulates faster for less nice people and slower for nicer people. So using the NQS queue *fast* means „start me soon and run me quick, and I'll pay more than usual for it“. On the contrary, in *slow*, a job will start much later and run much slower, but at significant less costs.

There are some potential problems:

- A user can put a job in the *fast* queue to get scheduled quickly, but use the *nice* command to lower the cost. The job will run slower, but it runs. To avoid this, the *nice* command has been disabled for batch sessions (local mod at ZIB), and the *nicem* command will be disallowed for ordinary users.

- The *csch* has a built in *nice*.

- The *qsub* command allows to specify a higher (nicer) nice value as the queue default. Due to a minor bug in the NQS-URM interface, the nice value is currently not passed to URM. If this is solved, the *qsub-nice* value can be used to influence the job ranking within URM (an user exit is provided in *site_rank.c*).

For a period of 4 weeks we did not see the correct *rank* in the URM tables according to the FSS values of usage and shares. We had a lot of trouble with our users, because the URM-NQS-scheduling did not work as committed. Our CRAY local representative learned from Eagen, that in this beta release URM used the shares values only and did not look at the usage values. This is solved now and the job ranking works as intended.

Our users need a NQS class like *express* with low limits for CPU-time for example for compilation and other development jobs. The reason is that the class *fast* is not always fast, only expensive. The main memory of 16 MW of the XMP often is a bottleneck for all jobs. Most of our users still work with batch jobs instead interactively for development.

Direct FDDI-connection

We will get a FCA-1 (FDDI channel adapter) for our Y-MP within the next weeks. There we will have some new questions.

Conclusion

Being a CRAY-betatest-side was (or even is) a great experience. We had a lot of work, but we did not regret it. Thanks to the people of CRAY, especially our local CRAY representative Alois Reimer.

UNICOS 8.0 - Field Test Experiences

Douglas A. Spragg

Exxon Upstream Technical Computing Company

Introduction

The purpose of my talk is to share some of the experiences that we have had running on UNICOS 8.0. As one of the five or so Field Test sites chosen by Cray to Beta test the UNICOS 8.0 product, we have been running it since October of 1993. The main reason that we were chosen as a Beta site was because we are a heavy user of tapes both to support our seismic users' processing needs, and to support our use of Cray's Data Migration Facility (DMF). Since we have been up on UNICOS 8.0, we have also installed a 128 processor T3D. Thus the experiences that I will be sharing will inevitably include observations about running the MPP. When I get into reliability (MTTI) statistics it is often difficult to separate out problems that are generic to UNICOS 8.0 versus those that we have encountered because we are running the MPP.

Environment

Let me start off by telling you a little about our Company, and our environment. EUTeC is actually a distinct Exxon Division with about 300 employees set up in late 1990 to act as a service provider for Exxon's "upstream" (exploration, research, and production) users. At the time the organization was formed we consolidated two computer centers that had been in other Exxon organizations. In doing this, we retired an old 4 Mwd Cray X-MP, and an antique Cray 1-S, and replaced them with a model D Y-MP. Although we started off with only two processors, we had enough foresight to buy an 8 CPU frame so that we

could gracefully grow our way into the full blown configuration. In 1993, we changed out our model D for a model E to support the installation of the T3D which we did in November. Our current Cray configuration is:

Hardware

- Y-MP 8-E
 - 4 CPUs
 - 128 mwds main memory
 - 128 mwds SSD
- T3D
 - 128 PE
 - 8 mwds / PE
- Approximately 250 Gbytes of disk (DD-41,DD-62,DD-60)
- 3480/3490 Tapes
- E-MASS Data Tower with 3 recorders (ER90s)

Software

- UNICOS 8.0
- MVS Mainframes used for tape front-end servicing
- Heavy use of DMF (Data Migration) to 3490 and ER90
- Base level MLS (Multi-Level Security) - mostly for auditability.
- NFS Client and Server

Our customer base is entirely comprised of Exxon Affiliates, with the applications being a mixture of seismic, reservoir simulation, structural analysis, and chemical modeling. To give a perspective on history, and even though we have only been operating as a company for a little over 3 years, we have run production on Cray software all the way from COS 1.17, to UNICOS 8.0.

General Comments

Recall that we were specifically chosen by Cray to be a Beta test site because of our extensive tape usage. On an average day, we may mount as many as 2000 3480 tapes, and 3480 tape usage is in addition to DMF activity. This probably means that Cray thought that the piece of UNICOS that we would be most likely to break would be tapes. From our point of view, doing the Beta test was a convenient way to get quickly up to speed on the software that we knew the MPP developers were aiming at, and get off the UNICOS 7.C that we were running. We had installed UNICOS 7.C to support our ER90s in the August time frame but were frankly concerned about running on a "special leg" of UNICOS software. Thus, I guess the big question is did we break the UNICOS 8.0 tape software? Somewhat surprisingly, the answer in large part is no. Right after we came up, we had some tape problems relating to a resource count going negative. However, these problems were addressed, and fixed very quickly. Shravan Pargal from Bill Kennedy's tape development group was on our site during the test, as well as Bob Rekieta from Technical Support. One of the very nice things about doing a Field Test is the fact that Cray pays close attention to your problems and really does get them fixed. Later on in the test, we had other tape issues, but these were due to codes in our shop that had been compiled and linked under very old versions of UNICOS (would you believe 5.1?), and wreaking havoc by calling for tape positioning information (GETTP). It would appear that some data structure that GETTP needed had changed, and the tape daemon fell into a logic hole. We also had some sporadic library incompatibility problems that caused programs to run out of memory. These were mostly addressed by having our

users recompile these programs with the most recent version of cf77 (version 6.0). One other library issue that bit us was the assign command. Many of our applications actually read the file pointed to by the FILENV environment variable. This file has traditionally contained the relevant information about each file encoded with colon delimiters. In the UNICOS 8.0 version, these colon delimited fields are no longer present. To accommodate our users' immediate needs, we are still using the UNICOS 7.0 version of the assign command. A word to the wise here is that when you start changing systems a lot, you need to remind your applications developers to keep their application executables at or very near the version of the system you are running. This may be more a problem for us than is the case at your site because we exercise little control over the applications that are run on our system.

The UNICOS 8.0 Field Test completed with no serious hitch. In fact, we had no system interrupts at all until late November. It was only when we brought in the T3D that we started to have real problems. The fact is that since November 18, we have had 16 system halts on the Y-MP. In breaking this set of data down, we feel that 4 are UNICOS 8.0 problems, 6 are directly related to the fact that we are running an MPP, 3 are site procedure problems, and 3 are of unknown origin.

System Halts

Seeing as how we have both UNICOS 8.0 software and MPP software in the mix, it might be useful to draw a time line where we put in dates that we installed various versions of both software and the interrupts experienced. I will use a code to signify how we have classified the various outages. The table is on the following page:

Key:	
8 - UNICOS 8.0	S - site
M - MPP	U - unknown

Event	Date	Key
Start of UNICOS 8.0 Production	October 16	
System panic - bad network command sequence	November 18	8
System panic - /dev/slog full on dump/restore	November 22	S
System panic - /dev/slog full on dump/restore	November 22	S
Install of MAX 1.0.0.2 MPP software	November 26	
System panic - Assert in YPE driver	November 27	M
System hung (no panic) - kernel/mppexec problem	December 17	8
Install of MAX 1.0.0.3 MPP software	December 19	
System panic - Assert in nc1vops.c	December 29	8
Processes hung in sbrk call (nschedv X=1)	January 7	U
Install of MAX 1.0.0.5 MPP software	January 16	
System hung (no panic)	January 19	M
System panic - recursive lock	January 20	M
System panic - recursive lock	January 20	M
Re-install of MAX 1.0.0.3 MPP software	January 20	
System panic - nallookup failed - NFS config corrupt	January 25	M
System hung (no panic)	January 28	8
System panic - may have been result of site error	February 1	S
Install of MAX 1.0.0.6 MPP software	February 6	
System hung (no panic)	February 8	U
System panic - Assert in YPE driver	March 5	M
System hung (no panic)	March 9	U

As you can see, none of these outages are in any way related to tapes. The problem that we encountered on December 29 is related to kernel resident DMF code. In general, it would appear that in the main, we have stumbled over multi-threaded kernel problems that may be aggravated by the heavy load of system calls that the MPP is throwing back over the fence for UNICOS to handle. For the most part, the problems reported have been fixed in a timely manner, and installed as soon as practical. In our case, scheduling maintenance time can be difficult because of our large production workload. In summary, only 4 of our system hits were directly attributable to UNICOS 8.0.

Performance

In advance of someone asking, let me discuss performance. I would love to be able to get up here and give you all some real pearls about things we discovered relating to UNICOS 8.0

multi-threading performance. The sad fact is that I have precious little information to share. One of the main reasons for my lack of data, is simply that our environment has changed so drastically. In 1993, we added CPUs, disk, converted from the model D to the model E, added the E-MASS, and added the T3D. In that same time frame our work mix changed just as drastically. In a general sense, the recent upgrade that we made from 64 mwds of main memory to 128 mwds has made a tremendous difference in our throughput. We are able to regularly sustain 100% CPU utilization during our heavy workload periods whereas before we were maxed out at about 70%. We still have more system time than I would like to see - ranging as high as 20-30%. This is something that we have seen on previous UNICOS releases, however, and believe that it has more to do with our job mix than anything else. It also appears that the MPP aggravates problems with system CPU usage. During the middle of

February, we had Steve Luzmoor from the West Coast Technical Support Group out to our site, and he found several issues (none of them catastrophic) which we intend to follow up on. In one case, the mppexecs that we were running were being locked, migrating towards the lower portion of memory, and in turn booting out users' telnet sessions. This in turn was creating some very inconsistent interactive response conditions. Our site was Steve's first experience with mppexec, and I think he found it a tricky piece of software to deal with.

Conclusions

In the main, I consider our UNICOS 8.0 Field Test a success. The main reason for saying that is that we would have been in somewhat of a sticky position doing a conversion from UNICOS 7.C to 8.0 in our current environment. Our production workload has increased dramatically over last fall, and at the same time workload on the MPP is ever-increasing. It would be far more difficult to schedule the required system maintenance now than it was last fall. In addition, Cray is now **STRONGLY** advising all MPP sites to get to UNICOS 8.0 as quickly as possible. On the other hand, I am a little disappointed in the number of system problems we have had. I am particularly concerned about the fact that so many of our system outages have been due to the fact that we are running the MPP. I had hoped for a better "fire wall" if you will between the Y-MP and the MPP.

In addition, we still have at least one unresolved problem. Since we have come up on UNICOS 8.0, we have been unable to get the NFS Automounter to work. It just hangs when we try to start it. Because Eagan has been unable to replicate the problem, it has been a difficult issue to track down.

One other problem we encountered was a lack of timely updates. We installed UNICOS 8.0.1.1 in October, and went to UNICOS 8.0.1.2 when we installed the T3D in November. After that, there was no further update available until March. It would have been nice to have at least one more intermediate update.

Before I conclude, let me share with you some other thoughts that are the outcome of a

Process Improvement Team that Exxon and Cray worked on together to improve our site's overall MTTI during 1993. Our basic conclusions are relevant to more than UNICOS 8.0, and so they may belong in another talk, but they are simple and straightforward so let me state them quickly.

- 1) There is a continuing frustration in the use of the install tool to build a system. In particular, it was several months after we got the MPP before we were able to straightforwardly build a parameter file acceptable to UNICOS using the install tool. Also, because the MPP CSL parsing code has been at times either bad or at least not current, we have actually had situations where expected system configuration parameters for other sub-systems have been affected (see January 25 system halt). Our frustration is exacerbated by the fact that the people in Eagan do not appear to use the install tool when they build systems.
- 2) There are too many instances of software provided at one generation of Cray hardware not being adequately carried on to the next. One example we identified was Cray's failure to carry over some tape resiliency mods from the model D to the model E environment. Another example was our discovery last summer that we had significant problems with the operation of DD-41 disk diagnostics/utilities in the model E.
- 3) The process that Cray uses to manage SPRs still appears to hide a lot of potentially useful information from the customer. Unless the customer is extremely diligent in following SPR progress, which is the case since we have been running on the T3D, the information flow on SPRs appears sporadic and poorly coordinated between Software Development and Customer Service.
- 4) We feel that in the main, Software Development is unnecessarily remote from what real Cray customers do with Cray products. There exists a need in our minds for Software to spend more time at

the customer sites. In our case, we believe that having Shравan Pargal here from Tape Development during our Field Test was of mutual benefit.

Summary

Our site has been running UNICOS 8.0 as our production system since October of 1993. While we have encountered several system outages, we believe that (excluding the MPP) the number is in the range of what we would expect for Beta software. In addition, we have encountered no serious problems with any major UNICOS sub-system, and in particular the tape daemon. It appears to us that UNICOS 8.0 will in fact live up to Cray's expectations for reliability. Our experience is that Cray does an excellent job in following

up with problems that are encountered in running Beta software. In fact, I think we got really superb service - especially from tape development in solving the issues that we did have.

Acknowledgments

I would like to thank all of my staff at EUTeC for helping to make the conversion to UNICOS 8.0 and the installation of the T3D a success. In addition, I would like to extend special appreciation for the efforts of our two Cray site analysts (Bryan White and Ed Liu). Their prompt attention to our problems and their excellent interface work with Cray in Eagan were an enormous help in getting us to timely solutions to our problems.

SDSC HOST SITE PRESENTATION

Daniel D. Drobnis
Michael Fagan
San Diego Supercomputer Center
San Diego, California USA

Introduction

The San Diego Supercomputer Center is a National Laboratory for Computational Science and Engineering. The Center was established in 1985, and is currently working to advance computational science research and enhance U.S. economic competitiveness. This is accomplished through three basic activities: research and development, education/training/outreach, and service. In particular, the Center

- Participates in computational science research
- Supports others doing science
- Educates and trains computational scientists
- Builds tools to advance computational science
- Enables technology transfer to U.S. industry

The San Diego center is one of four such centers supported primarily by the National Science Foundation (NSF). The others are associated with the University of Illinois (The National Center for Supercomputer Applications, with Carnegie-Mellon and the University of Pittsburgh (the Pittsburgh Supercomputing Center), and with Cornell University (Cornell Theory Center).

The San Diego Supercomputer Center is operated by General Atomics, a privately held Research and Development company, in association with a group of 25 user institutions. This user group, or consortium, provides policy guidance to the Center, and includes, in addition to all of the UC campuses, other prominent universities elsewhere in California and across the U. S. from Hawaii to Maryland.

Finances

Most of the computational resources of the Center are distributed free of charge to academic researchers who have submitted proposals to the Center's Allocation Committee, made up of leading U.S. computational science researchers. The best proposals are selected for

a peer-reviewed grant of resources--principally computer time--just as they would receive a grant of money to support their research. SDSC also maintains an active industrial partners program.

The principal financial support for the Center and its communication facilities comes from the federal government through NSF, and additionally through other research-oriented agencies such as the Defense Department Advanced Research Projects Agency (ARPA) and the National Institutes of Health (NIH). Support comes in addition from our consortium members, the State of California (both directly and through the Regents of the University of California), UCSD, and industrial partners.

There are currently over 500 peer-reviewed projects, with over 3000 users. There are also over 60 industrial partners, who contribute equipment, services, and money to the Center, and receive supercomputer time, visualization assistance, training, and staff support.

Communications

Consortium members, industrial partners, and others access the Center over high speed data communications lines, both direct and through connections to the nationwide interconnected research communications network known as the Internet. The total communications capability of production trunks and network connections to the Center exceeds 60 million bits per second. The major portion of this is the Internet's cross-country trunk, operating at T3 speeds--45 Mbits/second.

In addition, an experimental network (the CASA project) links the Center to the Los Alamos National Laboratory in New Mexico, and to Caltech and the Jet Propulsion Laboratory in Pasadena. It utilizes optical fibers for communication, which are already in place, and operates at over 800 million bits per second.

The Center is designed to provide complete service to people hundreds or thousands of miles away. One of the principle challenges in its operation is to provide the same quality of support and services to users hundreds or

thousands of miles away as is provided to those within the building or in San Diego.

Research and Development

Most applications development and computational research is done by the Center's academic users. However, an active research and development program is also carried on by the Center staff; areas currently active at the Center include:

- **Networking**
 - Gigabit networks (CASA)
 - Network performance monitoring
 - Analytical modeling (NREN)
- **Operating Systems**
 - Parallel processor scheduling
 - Parallel architecture resource management
- **Visualization**
 - Volume Visualization tools (NetV)
 - Interactive Interfaces (Sequoia 2000)
 - Image manipulation (ImageTools)
- **Applications Development**
 - Global Climate Modeling
 - Molecular modeling on parallel architectures
- **Primary and Secondary Education**
 - K-12 outreach presentations
 - Secondary school science teacher training

Facilities

The Center is housed in a building designed and constructed specifically to house it. The building has about 23,000 square feet of computer room and mechanical space, and about 35,000 square feet of office and meeting facilities. It cost about \$8 Million to construct and furnish in 1985, and houses about 100 people.

The Center's workhorse supercomputer is currently an 8-processor Cray C90, installed in November, 1993, with a peak computational speed of 7.8 GFLOPs for 64-bit operands. It has 128 Mwords (1 Gbyte) of main memory, and 1024 Gwords (8 Gbytes) of SSD. There are 188 Gbytes of directly attached disk: 13 DA-62 arrays, 4 DD-60 disks, and one string of DD-42s.

SDSC has had two previous Cray Research computers. In 1986, the Center opened with a Cray X-MP, and a successor Cray Y-MP served during the period 1990 through 1993.

The Center also operates an Intel Paragon massively parallel computer. It presently contains 400 nodes, each containing two i860-XP processing elements--one for computation and one for communication. It is rated at a combined peak speed of 20 GFLOPs for double precision (64-bit) operands and 40 GFLOPs for single precision.

File Management

The high performance computers utilize a DataTree file management system, which provides permanent common file storage for all user files at the Center. The system runs on an IBM-MVS platform, and manages all data in a hierarchy, with smaller and frequently accessed files on 3380 disc, and larger and less-used files on 3480-format cartridge magnetic tapes. A Storage Technologies tape robot handles the most frequently accessed tape cartridges automatically. With disk and robot file caching, about 95% of all file requests are answered without human intervention.

DataTree currently contains over 5 Terabytes of information in over 1.5 million files. Files are added at the rate of about 25,000 per month. Later in 1994, the file management system will be migrated to UniTree, which will provide improved performance and features, and will operate on an IBM RS-6000 Model 980. It will also control a Storage Tek cartridge tape robot when it goes into production.

Supporting the supercomputers, a couple of dozen more computers and communication routers provide service to Center users, and almost 20 high performance workstations in our Scientific Visualization Center translate the results of computations into graphic images. Workstations are served by an Auspex NFS file server, which allows them to remote mount their application file systems and user home directories.

All of these computers are interconnected by a local high speed communications Ethernet system at 10 million bits/second. An additional system links the fastest computers and uses FDDI optical fibers at 100 million bits/second. Altogether about 100 workstations and 120 personal computers are installed in the SDSC building.

The CASA Gigabit network testbed, which operates at 800 million bits per second uses HiPPI (High Performance Peripheral Interchange) channels and a 32-channel crossbar switch.

Control and Monitoring

The SDSC Control Room is designed to allow operation of the entire Center, including building facilities and security, from one place. If necessary, the entire Center can be operated and monitored by one person, although we normally have two operators on off-shifts, and three during the day on weekdays. A useful tool for centralizing status monitoring is a PC-based status monitoring and summary system, which watches computer and router console output message streams.¹ This tool, complete with

documentation and usage examples, is available from the Center's anonymous ftp server at ftp.sdsc.edu.

Visualization Lab

An Advanced Scientific Visualization Laboratory is located directly off the computer floor. Its workstations and CPUs can thus be immediately adjacent on the raised computer room floor, keeping the VisLab itself quiet and comfortable. The major elements of the VisLab are a variety of graphics workstations, principally from Silicon Graphics and Sun. These are used both to generate production visual images by researchers and students nearby, and to provide a base for the development of tools which can be used by remote researchers on workstations closer to their homes. The set of tools developed for manipulating visual images (SDSC ImageTools) have proved so popular that SDSC has had over 3000 requests for them through anonymous ftp.

Video Post Production

After researchers have computed their data on the supercomputer, and used the facilities in the VisLab for converting the data from numeric form to computer graphics images, they often record their images onto video disk. Once recorded, the raw footage is edited onto video tape for scientific study, presentations at colloquia, submissions to judged competitions such as SIGGRAPH, training and demonstrations, and productions of broadcast-quality "programs."

Training Facilities

A specially equipped training facility is available for Workshops and Training Seminars for users of the Supercomputer Center, as well as for training on personal computers. Both the Center and the University's Adult Extension make extensive use of this area. It is equipped with 20 Apple Macintosh IIcx computers, with large high-resolution color monitors capable of full X-Windows resolution, 9 MB of RAM, and 80 MB hard discs. The material on the Instructor's screen is projected onto the screen on the front wall, so that students may see what the instructor is demonstrating, and follow along on their own screens.

The Center's auditorium is equipped to support a wide variety of meeting needs for the Center, the University, Industrial Partners, and local computer user groups. It can be set up with tablet-arm chairs, for large lectures or group meetings of 100 or more people, or with tables and chairs for seminars and conferences. In addition to slide, motion picture, and overhead projection facilities, the video and sound systems allow viewing of video tapes, cable and broadcast TV, terminal and workstation screens, and video teleconferences. The auditorium is equipped with the

same infrared synchronizers to permit viewing of high-resolution 3-D workstation images as are used in the VisLab. The Center is connected to both UCSD's broadband TV distribution system, and to San Diego State University's Profnet instructional microwave system.

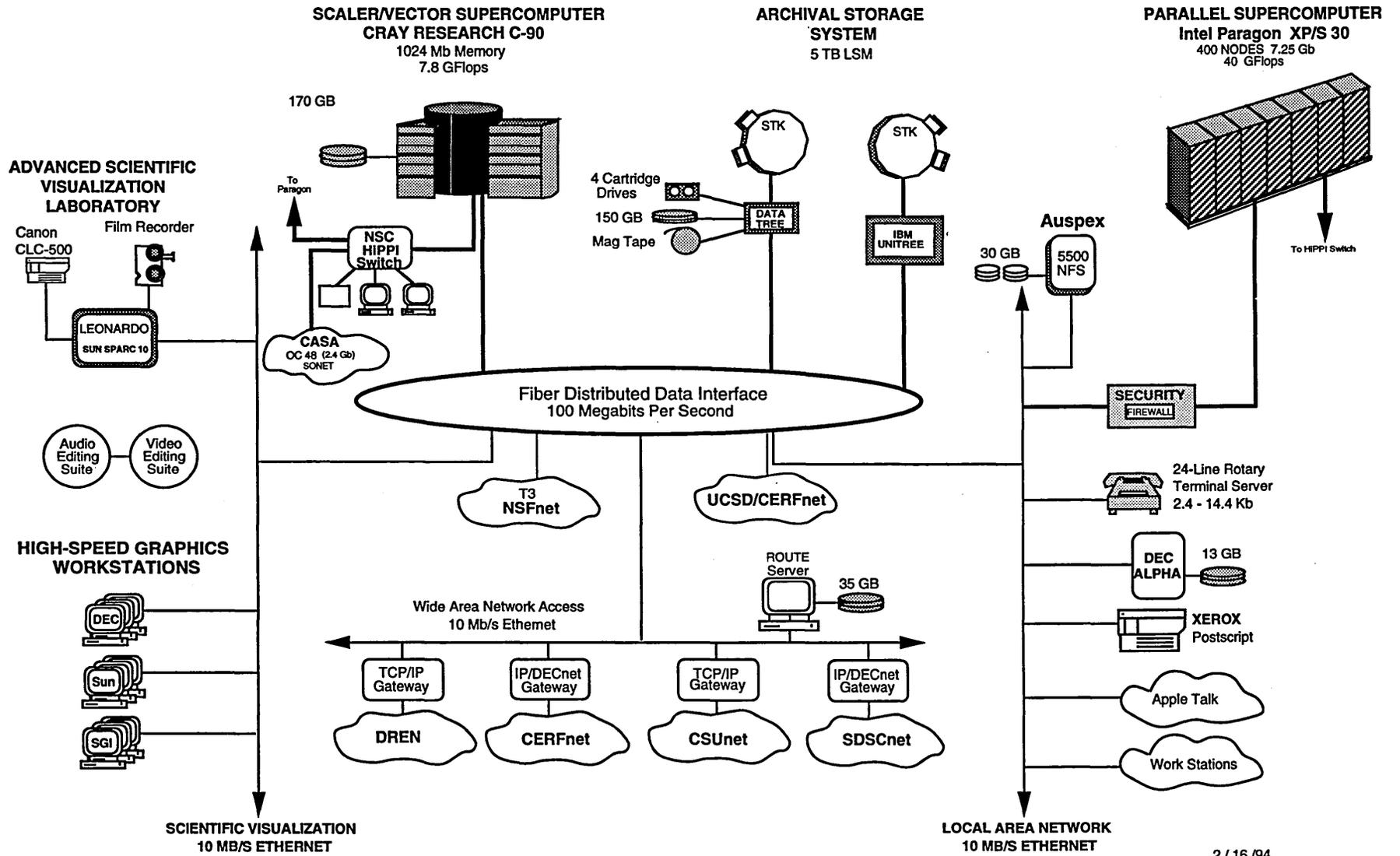
Work on this project was performed under NSF grant number ASC-9019070.

Dan Drobnis is the manager of the Engineering and Operations department at SDSC. Mike Fagan is a member of the Engineering, Networking, and Security group at SDSC.

Reference

1. Dombrowski, H.F. and Drobnis, D. D., "PC-Based Status Monitor Now Available", *Proceedings of the Spring 1993 Cray Users Group Meeting*, Montreux, Switzerland 1993.

SDSC SYSTEMS CONFIGURATION



Tools for the Cray Research OWS-E Operator Under UNICOS 8.0

**Leon F. Vann
Project Leader
Computer-based Training
Software Information Services**

Cray Research, Inc., Eagan, Minnesota, U.S.A.

Abstract

A full complement of operator tasks and functions can now be performed easily through the operator interface (*opi*) utility. With the release of UNICOS 7.0.5, *opi* is now a part of the standard UNICOS release package. It is being enhanced with optional choices such as automatic dumps and reboots in the UNICOS 8.0 release. Training for the OWS-E operator is being revised to include functioning through *opi* as well as through standard commands.

Overview of Progress and Process

Significant progress has been made to make operating Cray Research systems easier for the operator: the operator interface (*opi*) feature that is part of the OWS-E software, the documentation set, and computer-based training. The following information further describes these products and provides an overview of the processes that are in place to enable continued improvement.

- Cray Research support personnel (on-site and in support centers)
- Questionnaires
- Contact with customer personnel attending training in Eagan and at customer sites
- Information exchanged through the Operator Environment Group (OEG)
- Reader comment forms in documentation
- CBT questionnaires
- Informal and formal communication exchanges

Identified Customer Concerns

Cray Research, Inc. (CRI) continues to strive to identify customer requirements for operator information in order to improve our products, training, and documentation.

A variety of vehicles have been and are being used to solicit customer needs.

Major sources of customer input are as follows:

- Cray Users Group (CUG)
- SIS Customer Advisory Board (CAB)

We encourage the use of all of the above as a means of communicating and validating needs and solutions.

A summary of identified customer concerns follows:

- Cost of training in terms of both time and money and being able to obtain the training when needed
- High turnover rate of operator personnel
- Knowing where to find and gain access to documentation
- Being able to provide training to operations personnel that fits their level of expertise and current environment

- Need for operators and operations personnel to have a customizable, easy-to-use interface to Cray Research systems
- Need for online training that is available when needed (CBT)
- Capability to customize documentation, training, and the system interface
- Lights-dim operations environment
- Lights-out operations environment

Current and Future Activities

Operator Environment Group (OEG)

This group is made up of representatives from the following software groups: development, testing, publications, training, release coordination, and Cray Corporate Computing and Networking (CCN), as well as an email information exchange for all of the above groups and customers through the email address oeg@cray.com. The OEG provides a forum for all operator-related plans and projects.

Cray Research Operator Documentation

The *CRI Software Documentation Ready Reference*, publication SQ-2122, lists all CRI software publications. A complete set of documentation is available for the OWS-E and includes the following publications:

OWS-E/IOS-E Reference Manual (SR-3077)

OWS-E/IOS-E Operator's Guide (SG-3078)

OWS-E/IOS-E Administrator's Guide (SG-3079)

OWS-E/IOS-E Ready Reference (SQ-3080)

The UNICOS 8.0.3 release will enable the OWS-E documentation, except for the *OWS-E/IOS-E Ready Reference*, to be accessed online using CrayDoc, which provides hyperlinked online information.

Cray Research Operator Training

Curriculum Planning Team

The SIS Curriculum Planning Team (CPT) includes instructor and management representatives from each

of the identified audiences. These audiences include operators, system administrators, end users, applications programmers, and systems analysts. The short-term goal of the CPT is to review and revise the current curriculum so that changes can be made and reflected in the next issue of the CRI Software Training Catalog. The group's long-term goal is to identify a process by which the curriculum can be reviewed and modified at regular intervals.

The CRI Software Training Catalog contains the following for each intended audience:

- Audience definition
- Recommended sequence (flowchart format)
- Skills map (indicates the skills covered by each course and the level of course content, that is, basic, intermediate, or advanced).
- Course description, which includes the objectives and the outline
- Schedules for classes offered at CRI Headquarters, Regional, and Country training sites

Training Plan

It is strongly recommended that customers work closely with the Cray Training Coordinator and account representative to establish a training plan for each identified audience. This plan can then be implemented by using the training available in various forms from Cray Research; courses can also be customized to meet customer needs.

Operator Training Classes

Two types of Cray Operator Training (COT) courses are available: computer-based training and lecture/lab. The current computer-based training course (COT-CBT) runs on the OWS-E, supports UNICOS 7.0, and uses simulated exercises, while the lecture/lab course (COT) has hands-on exercises and includes using windowing tools such as *opi*. The COT course also can be customized to fit the customer's hardware configuration and specific operations procedures.

Both Cray Operator Training courses address the following topics and skills:

- Basic UNIX usage
- OWS, IOS, and CRI system startup and shutdown
- Response to requests; tape and file system
- System dumps and backup procedures

- Response to errors and general troubleshooting procedures
- Monitoring system activities

Future Computer-based Training (CBT) Plans

Future CBT courses will be developed and delivered using a new feature-rich system called CrayTutor. Courses developed using CrayTutor will run on UNIX workstations, like the OWS-E or equivalent Sun SPARCstations and can use color, graphics, animation, and sound.

Future courses to be developed using CrayTutor are the following:

- CRAY EL System Administration Training Phase I: basic operator/administrator tasks
- Cray Operator Training for UNICOS 8.0
- UNICOS 8.0 Basic User Training

Other courses and/or specific topics will be considered and prioritized according to customer needs.

Start-up Services

A series of customer-integrated solutions, known as Quick Starts, have been defined in the following areas:

- Operator's quick start
- System administrator's quick start
- Application conversion and optimization quick start
- Network administration quick start

The Quick Start service is primarily aimed at new customers, but is useful in transitions to new hardware and software. The services provide a CRI specialist to work with the customer staff in planning and making technical decisions. Procedures are recommended and defined to fit the customer environment. The overall objective of the Quick Start is to shorten the time required to bring a new system into full production.

The operator's quick start involves an experienced operator working with the customer's operations staff to discuss and determine operational procedures. This specialist also can either provide or customize scripts for identified procedures. Hands-on time is spent with lead operators performing standard and customized procedures.

The operator's quick start could and should be complemented by formal training. Before a Quick Start, lead operators should be trained in the basics in order to take full advantage of planning and customization possibilities. All operators can then be trained using the computer-based training (COT-CBT), followed by the lecture/lab course (COT) for hands-on experience using site procedures.

Operator Interface (opi)

The operator interface (opi) utility is part of the OWS-E software that allows operators to perform a variety of functions in a point-and-click manner. These functions include the following:

- Monitoring system status and activity on a number of Cray Research systems from a single OWS-E
- Halting, booting, and dumping the IOS-E and the mainframe on a selected Cray Research system
- Opening connections to Cray Research systems, including the UNICOS console
- Opening connections to local and remote OWS-Es
- New opi features available with the UNICOS 8.0 release, including autodump and autoreboot

Online help is available to provide information about tasks and functions. opi provides two windows that are used for standard output and error information.

One of the important features of opi is the flexibility to customize functions and help facilities to fit the customer's operations environment.

opi is part of OWS-E release 7.0.5 and subsequent releases. Future releases will also support Solaris 2.0.

Integrated Performance Support Systems (IPSS)

The goal of a fully implemented IPSS is to provide customers with the ability to use CRI tools and products in an expert fashion. With a standard graphical user interface (GUI) ensuring that CRI products are consistent with each other, and with online information tools that provide a seamless user environment, customers will be better able to do their jobs without in-depth technical knowledge of UNICOS.

The IPSS system includes access to online information in SGML source by way of CrayDoc. CrayDoc runs on a UNIX workstation and is scheduled to be delivered with future products this year, including UNICOS, CRAY T3D, UniChem, and the programming environment. Computer-based training for selected topics will be provided by CrayTutor, which also runs on UNIX workstations.

Customization to Fit Your Needs

The ultimate goal of an Integrated Performance Support System is to provide the capabilities to allow customers access to information more easily, to learn how to use CRI products more quickly, to get expert advice related to the tasks at hand, and to customize system components to fit the customer procedures and environment.

CRAY RESEARCH PRODUCT RESILIENCY

Presented by Gary Shorrel

Cray Research, Inc., Chippewa Falls, WI, U.S.A.

ABSTRACT

Cray Research, Inc. has placed continued emphasis on product reliability throughout the history of the company. Significant advances in reliability or product mean time to interrupt (MTTI) have been achieved with each new generation of hardware and software. In 1992, a project was begun to address users needs for high system availability, in addition to high system MTTI. The resiliency project has identified numerous opportunities for resiliency implementation in CRI hardware and software products. This paper will review the history and current status of the resiliency project and discuss specific resiliency feature implementations in CRI products.

RESILIENCY DEFINITION

Webster defines resiliency as “an ability to recover from or adjust easily to misfortune or change.” The CRI Resiliency Project Team has defined resiliency as “the ability of hardware and software to cope with failures so that the end user realizes 100% availability.” 100% availability is a lofty goal, but CRI clearly understands that end user availability is of utmost importance to our customers.

RESILIENCY AND RELIABILITY

Inherent reliability of CRI hardware and software has increased from hundreds of hours to thousands of hours over the past ten years, while at the same time hardware and software products became increasingly complex. Because of this increased complexity in hardware and software, it has become much more difficult to achieve increased end user perceived reliability with traditional methods of hardening the hardware and testing software.

Traditionally, Cray Research has subjected its hardware components and software systems to rigorous reliability tests in order to assure the highest possible reliability of our products. Today this approach has become increasingly difficult, so in addition to traditional reliability techniques, it has been determined that resiliency must be incorporated into our products in order to achieve our goal of high end user availability.

HISTORY OF RESILIENCY PROJECT

Realizing the need to advance resiliency within Cray Research, management initiated a project in 1992, which became known as the Cray Resiliency Project. The Resiliency Project was designed to take a “bottoms up” approach by asking individuals who design, test, and maintain Cray products on a daily basis, what could be done to improve the resiliency of our products.

In forming the System Resiliency Project Team, approximately 20 individuals were recruited from various areas of the company including Engineering, Software Division, Customer Service, System Test, and Manufacturing. The main objective of the team was to look at Cray Research’s products of today and the future and determine what can be done to make these products more resilient. The team focused on four areas of biggest impact. These being:

- Disk subsystems
- Mainframe
- I/O Subsystems
- Electro-mechanical

Four product sub-groups were formed based on these four areas of opportunity, each with a cross section of individuals with various backgrounds, job functions, and expertise. Each group researched resiliency issues and opportunities for current products, products in development, and future products. Primary focus was placed on the Cray YMP system, IOS-E, DD6x series of disks, and later generation products.

During the summer of 1992, the groups met weekly to develop a resiliency requirements document. In the fall of 1992, the group worked with Software Division to prioritize the issues and opportunities for resiliency of Cray Research's products. A system resiliency requirements document was produced in late 1992.

PROJECT GOALS

Goals established for the project were:

- Identify existing resiliency capabilities in current hardware products, assess the effectiveness of our use of resiliency features, and determine how unused features could be used.
- Identify resiliency features in new hardware being developed. Determine what plans are in place to utilize these features and recommend additional features where appropriate.
- Recommend additional areas of hardware and software design for resiliency for future products.

PROJECT AXIOMS

As the system resiliency project moved forward, the project team developed three axioms. These axioms have formed the foundation for our ongoing efforts.

- Products must be designed with emphasis on resiliency. Resiliency starts with the conceptual design of the system. Resiliency is not an add on option.
- Resiliency must be verifiable under realistic operating conditions. Cray Research must be able to verify that resiliency features designed into the product are actually providing benefit to the customer.
- Availability improvement must be seen and experienced by our customer. This is the overall end result of the resiliency project.

SPECIFIC RESILIENCY IMPLEMENTATIONS

The following are examples of resiliency features on Cray Research mainframes:

On older products (CRAY X-MP, CRAY 2, Y-MP systems):

- SECEDED on memory and on some of the channels.
- Some graceful degradation on multi-CPU systems.

Capability was fairly minimal and not much was designed into the products for resiliency.

On future products, some examples are:

- Spare P.E. nodes on the MPP systems.
- Partitioned memories that will allow for degrading memory systems.
- Spare memory chip implementation.

Disks have historically been one of the more difficult issues at Cray Research from a reliability perspective. Certainly the disks CRI is shipping today are orders of magnitude better than what has been shipped in the past. The DD6X series (and we believe the next generation will be even better) are almost not a factor when reliability analysis is done. But, nevertheless, there was some opportunity for resiliency implementation in the disk area.

The following are examples of resiliency features on Cray Research disk subsystems:

- On existing disk products (DD4X and DD6X disks):
 - Mirroring of critical files for IOS Model E disks with UNICOS 7.0 and 7.C.
 - Alternate path access to IOS Model E disks available with operator intervention with UNICOS 7.0 and automatic alternate path with UNICOS 8.0.
 - RAID technology with parity protection.

- On future disk products:
 - RAID disk with improved concurrent maintainability.
 - Power, cooling, and environmental monitoring built into disk products allowing for early warning of abnormal conditions.
 - Redundant power supply systems so that, in the event of a power supply failure, a service person will be able to concurrently remove and replace the power supply without an interrupt.

The following are examples of resiliency features on Cray Research I/O subsystems and SSDs:

- On existing products
 - Automatic system clear software on power up available in IOS-E 7.0.5.
 - On some of the air-cooled I/O subsystems, we have redundant (N+1) power supply technology implemented.
- The next generation I/O products will be the most concurrently maintainable product in the history of Cray Research.
 - Concurrent diagnosis and repair of hardware problems for most subassemblies.
 - Implement a spare memory chip on SSD products, allowing for spare memory chip mapping.
 - Redundant cooling systems, power supplies and cabinet designs that support concurrent maintenance.

As stated previously, resiliency must be designed into the product, not added on. The awareness of resiliency requirements within Cray Research has increased dramatically since initiation of the resiliency project. We believe that resiliency must become “a state of mind” in the development process for Cray Research hardware and software products. The importance of implementing resiliency into Cray Research products is recognized at the highest levels of the company.

CRI will continue to make product resiliency and reliability a high priority for hardware and software development projects. Continued emphasis on resiliency within Cray Research will result in higher end user availability and improved customer satisfaction with Cray Research products.

CURRENT RESILIENCY STATUS

The resiliency project team has completed and prioritized project resiliency requirements. The resiliency requirements that are defined in the document are an integral part of hardware and software product plans. Product development teams have updated the requirements to include status of their projects and how resiliency requirements will be addressed.

Performance Evaluation

UNICOS 7.C versus 8.0 Test Results

C. L. Berkey

Cray Research Inc.
Eagan, Minnesota

This paper discusses some of the testing that was done to compare the UNICOS 8.0 Multi-threaded system with UNICOS 7.C. The tests and the testing methodology are described, the results are presented, and some conclusions are suggested.

1. Introduction.

The purpose of the multi-thread feature in UNICOS 8.0 was to reduce the Kernel overhead by reducing the granularity of the locked regions of code allowing multiple processors in the kernel simultaneously. A detailed discussion of the multi-threading code can be found in reference 1.

To measure the success of this feature a set of "multi-thread tests" were developed to show the effect of reduced overhead as the number of processors is increased. The multi-thread tests were then run on UNICOS 7.C and UNICOS 8.0 systems and the results compared. There are three parts to these tests:

1. *C I/O test* run with a varying number of concurrent copies,
2. *Fortran I/O test* run with a varying number of concurrent copies, and
3. A *Fortran workload* made up of the unoptimized and optimized versions of the Perfect benchmarksTM, the *Fortran I/O test*, a matrix multiply using 4 processors, and Nuet and Pueblo from the LANL BenchmarksTM each using two processors.

These tests were used to measure the difference in elapsed time, user time, system time, file transfer rate,

and physical I/O request rate between the two systems.

2. Configuration.

All the runs were done on a CRAY YMP-C90 with 16 processors, a 256 MW central memory, and a 1 GW SSD with 4 VHISP's. For the disk tests a file system with 33 partitions (0-32) of DD60's was used.

UNICOS versions 7.C.4ab and the 8.0.2ab were used for these measurements. The */tmp*, */root*, and */usr* file systems were cached to SSD. The file system */usr/tmp* which had the 33 partitions was not cached in SSD.

3. Test Description.

- *C I/O test*

The primary purpose of the *C I/O test* is to determine the maximum physical I/O request rate. Therefore a fragmented file with record sizes that are not well formed was chosen. The fragmented file causes extra reads for file extends and the SSD residency causes read before write and no blocks are kept in the system buffers.

The *C I/O test* writes a 40 Mbyte file in an SSD file system and then reads it twice. The I/O is synchronous and sequential. The following *write* and *read* calls are used, in a C program, to transfer data in record sizes of 1024 bytes.

```
rc=write(fd,a.barray,bytes_to_write);
```

```
rc=read(fd,blk_in,bytes_expected);
```

An NQS queue consisting of sixteen (16) identical jobs, each executing the *C I/O test* using a separate file in SSD, was created. Five runs were made with the number of concurrent processes of 1, 2, 4, 8 and 16 each. The job accounting data was captured for each job and the “sar” data was captured for each run.

- Fortran I/O test

The *Fortran I/O test* was designed to model the I/O requirements of a large application with a modest effort made to optimize the I/O bandwidth. The record size was selected so the available I/O bandwidth could be demonstrated while the physical I/O request rate could be correlated to the *C I/O test*.

The *Fortran I/O test* writes and reads a 163 Mbyte file several times. The reads account for two thirds (66.6%) of the I/O requests to the test file. The record size was a constant 65536 bytes for all the I/O requests. Synchronous and sequential I/O was performed using the following Fortran statements:

```
write(i) ia
```

```
read(i) ia
```

An NQS queue consisting of sixteen (16) identical jobs, each performing I/O to a separate file on SSD, was created. When using SSD the number of concurrent jobs was 1, 2, 4, 8, and 16, while for Disk test it was 1, 2, 4, 8, 16, and 32. Also for disk the number of jobs queued is equal to the number of jobs allowed to run concurrently. Up to 32 jobs are executed each using a separate DD60 partition. The job accounting data was captured for each job and the “sar” data was captured for each run.

- Fortran workload:

The workload was intended to model a site and also demonstrate the sustained I/O and memory bandwidth of the system. The autotasked matrix multiply (MxM) test using four processors generates a sustained memory demand of 24 words per clock period. The Perfect and LANL codes represent applications.

There are five sets of programs used in the *Fortran workload* to create a workload of 130 jobs:

- 8 copies of the Perfect Benchmark jobs. This created 104 Perfect NQS jobs with each job executing the baseline and the optimized version of the benchmark.
- 12 copies of the Fortran I/O test.
- 8 copies of the Fortran autotasking test (MxM 4000x4000) using 4 processors.
- 2 copies of the LANL Benchmark Neut, problem size 128 using 2 processors.
- 4 copies of the LANL Benchmark Pueblo, problem size 128 using 2 processors.

Five NQS queues, one for each program set, were defined. The number of active jobs allowed for each NQS queue was set as follows:

Fortran I/O	(4)
Fortran autotasking	(1)
LANL NEUT	(1)
LANL PUEBLO	(1)
PERFECT	(15)

The queues are started in the above order with a five second wait between each queue start. All jobs in the workload were equal in priority. The job accounting data was captured for each job and the “sar” data was captured, with a sample every 4 seconds using the “sadc” command for the entire run.

4. Results

The results are presented on the following pages in the form of graphs and tables. Each graph shows a bar graph for UNICOS 7.C and UNICOS 8.0. For each graph the x-axis is the number of concurrent processes and the y-axis is seconds or physical requests per second. Preceding each graph is a table showing the raw numbers and the percent difference between UNICOS 7.C and UNICOS 8.0. For the *Fortran workload* all the results are shown on one graph.

5. Conclusions

The main observation is the large reduction in system time for UNICOS 8.0 compared to 7.C as the number of concurrent processes is increased. The graphs of system time, Figures 3, 7, and 11, show the system time for UNICOS 7.C grows more than linearly as the number of concurrent processes increases. For UNICOS 8.0 system time increases very little between 8 and 16 concurrent processes. This smaller system time translates into smaller elapsed time and higher transfer rates. However, for the *Fortran workload*, the 43.5% decrease in system time leads to only a 3.9% improvement in elapsed time. The reduction in system time means there is more CPU time available to user code but this workload is I/O bound, as can be seen by the high sustained Mbyte rate, and does not have enough cpu work to use the additional cycles available.

The system time decreased on UNICOS 8.0 for all test cases with more than 2 concurrent processes. The higher system time, on UNICOS 8.0 for 1 and 2 concurrent processes, is due to the additional locks introduced for multi-threading. When only one or two concurrent processes are active the benefits of multi-threading maybe masked by increased path length. It should be noted that for a one CPU system these locks are not necessary and in fact for a one CPU system UNICOS 8.0 will bypass the locks and therefore system time should be close to that for UNICOS 7.C.

The increase in user time for the *typical Fortran I/O* on UNICOS 8.0 was isolated to some degradation in the Fortran I/O library. For the C I/O test there was an improvement in the user time. The differences in user time are not related to multi-threading but rather to changes to compilers and libraries.

No attempt was made to measure the effects of other changes to UNICOS 8.0 such as VNODES, MLS, etc. The assumption is that for these tests the effect of other changes is small compared to the multi-thread improvements.

6. Future

These tests will be run periodically to track performance in future UNICOS releases. More analysis and data reduction of the sar and accounting data should be done. There are a number of additional tests that could be done, for example: a disk version of the *C I/O test*, some

experiments with limiting the number of active CPU's to the number of active processes, and varying the I/O record sizes to get some additional data points.

7. Acknowledgements

I would like to thank Dick Sandness for providing the initial set of tests and procedures and helping to explain some of the results. Thanks also to Jeff Pomeroy, Dave DeHerder, and Neil Williams for helping to analyze the results and review this paper.

8. References

- 1) Neil Williams, *Performance Analysis of the UNICOS 8.0 Multi-threaded Kernel*, 1993 Fall Proceedings, Cray User Group Inc.

C I/O Test Results

Table 1:

Concurrent Processes	Elapsed Seconds			User Seconds			System Seconds		
	7.C	8.0	%diff	7.C	8.0	%diff	7.C	8.0	%diff
1	816	816	0.0	540.8	486.4	-10.1	260.0	314.4	+20.9
2	481	486	+1.0	540.8	486.8	-10.0	373.2	458.4	+22.8
4	361	302	-16.3	540.9	487.1	-9.9	812.2	606.5	-25.3
8	384	236	-38.5	541.0	487.4	-9.9	2135.5	908.0	-57.5
16	460	239	-48.0	541.3	487.4	-10.0	4757.1	941.7	-80.2
Concurrent copies	Total CPU Seconds			Mbytes/ Second			Phys I/O req/Second		
	7.C	8.0	%diff	7.C	8.0	%diff	7.C	8.0	%diff
1	800.8	800.8	0.0	2.2	2.2	0.0	2875.5	2875.5	0.0
2	913.9	945.2	+3.4	3.8	3.8	0.0	8263.7	8332.2	+0.8
4	1353.1	1093.6	-19.2	5.1	6.1	+19.6	11218.1	13407.8	+19.5
8	2676.5	1395.3	-47.9	4.8	7.8	+62.5	10546.0	17150.5	+62.6
16	5298.4	1429.1	-73.0	4.0	7.7	+92.5	8803.6	16941.7	+92.4

C I/O Results

Elapsed Time

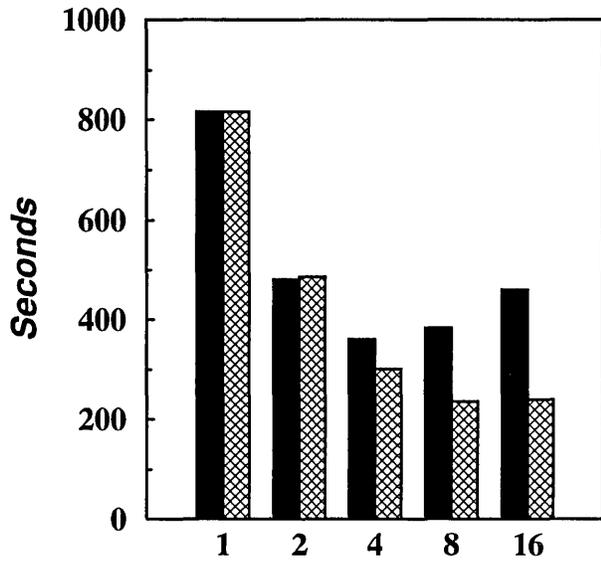


Fig 1. Concurrent Processes

User Time

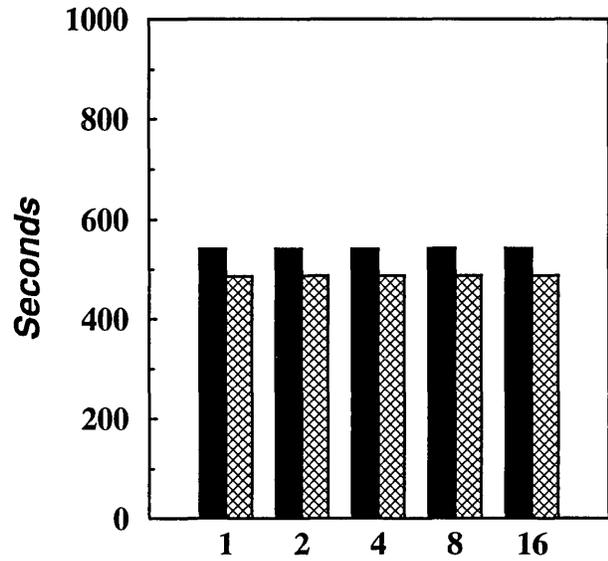


Fig 2. Concurrent Processes

System Time

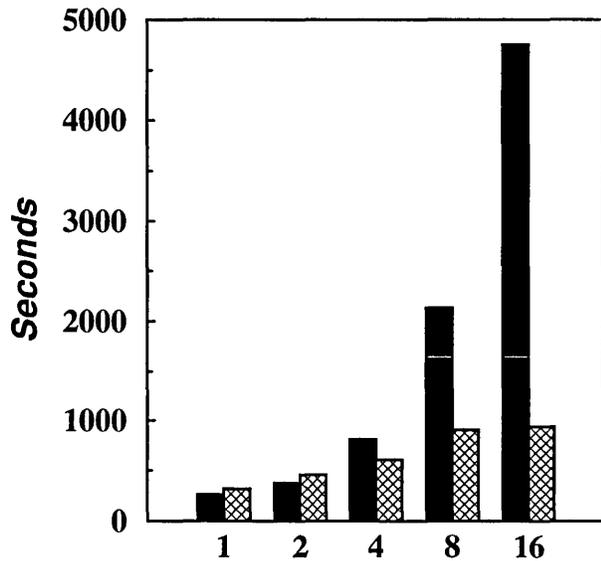


Fig 3. Concurrent Processes

Physical I/O Request Rate

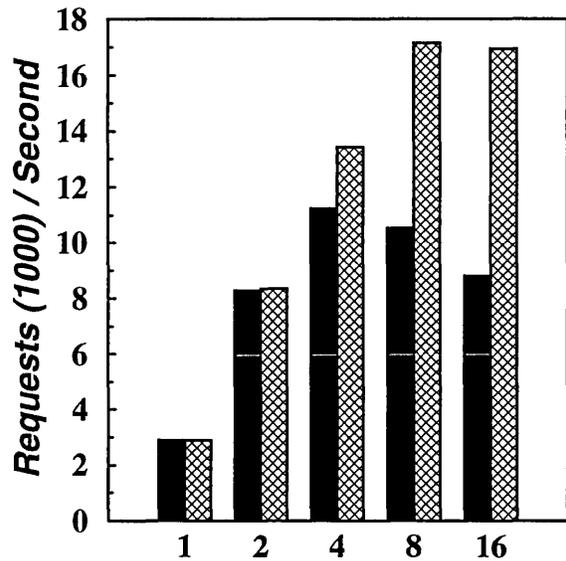


Fig 4. Concurrent Processes



Fortran I/O test results (SSD)

Table 2:

Concurrent Processes	Elapsed Seconds			User Seconds			System Seconds		
	7.C	8.0	%diff	7.C	8.0	%diff	7.C	8.0	%diff
1	239	330	+38.1	36.7	50.9	+38.7	183.0	258.7	+41.5
2	181	167	-7.7	36.8	50.9	+38.3	300.2	262.4	-12.6
4	175	140	-20.0	36.8	51.0	+38.6	608.5	397.5	-34.7
8	172	125	-27.3	36.8	51.0	+38.6	1286.6	503.8	-60.8
16	180	124	-31.1	37.2	51.0	+37.1	2739.8	505.9	-81.5
Concurrent Processes	Total CPU Seconds			Mbytes/ Second			Phys I/O req/Second		
	7.C	8.0	%diff	7.C	8.0	%diff	7.C	8.0	%diff
1	219.8	309.6	+40.9	502.1	363.6	-27.6	8045.5	5822.6	-27.6
2	336.9	313.5	-7.0	663.0	714.3	+8.4	10622.6	11505.5	+8.3
4	645.3	448.7	-30.5	685.7	857.2	+25.0	10987.7	13724.4	+24.9
8	1323.4	554.7	-58.1	697.7	960.0	+37.6	11178.5	15371.4	+37.5
16	2777.0	554.7	-79.9	666.7	960.0	+45.2	10682.6	15495.4	+45.1

Fortran I/O SSD Results

Elapsed Time

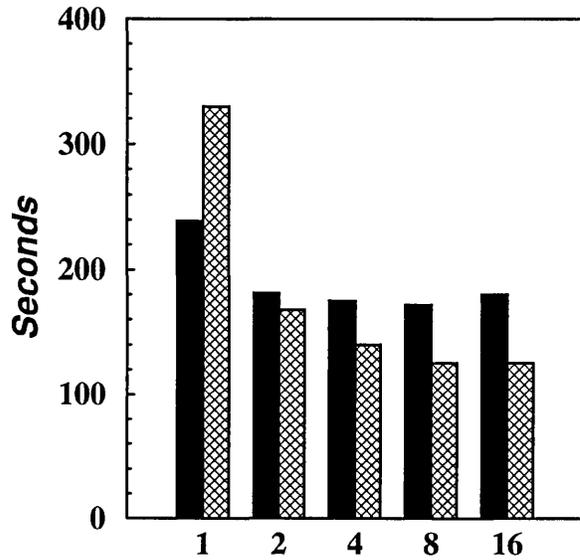


Fig 5. Concurrent Processes

User Time

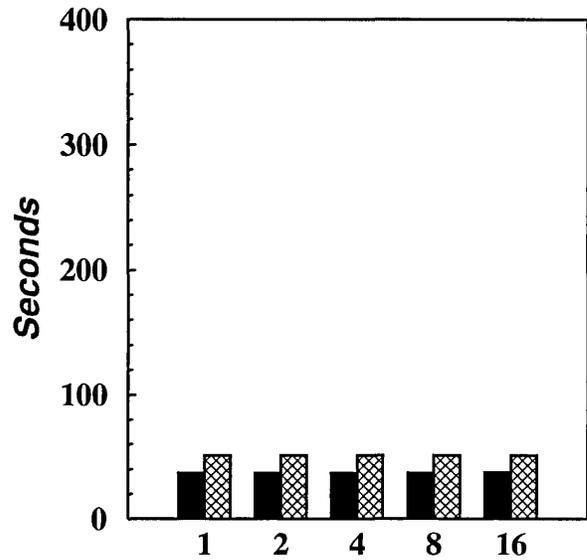


Fig 6. Concurrent Processes

System Time

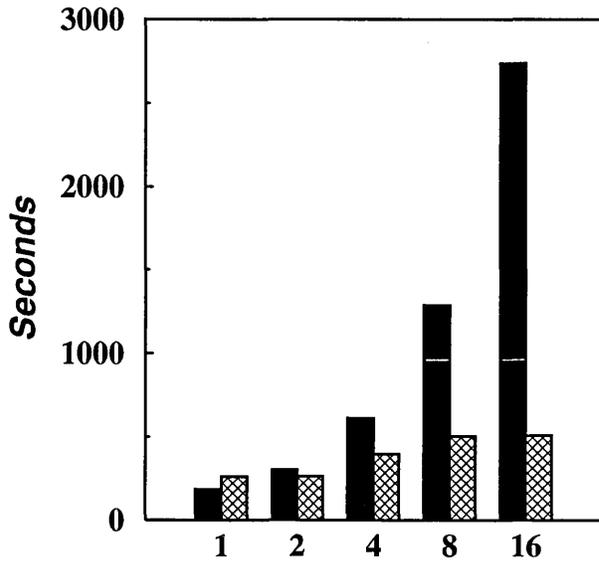


Fig 7. Concurrent Processes

Physical I/O Request Rate

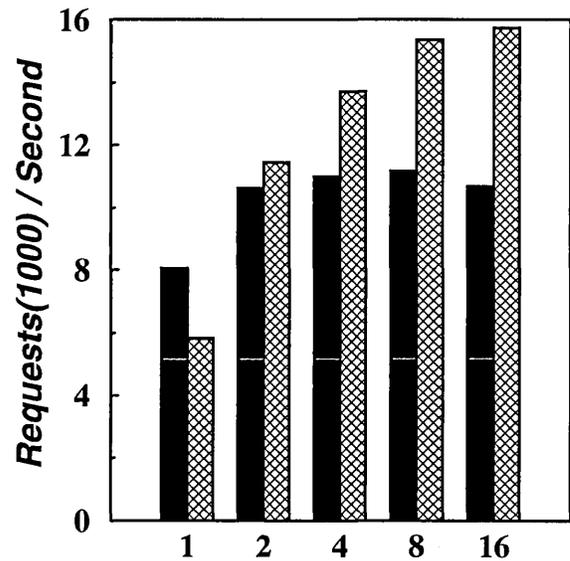


Fig 8. Concurrent Processes


UNICOS 7.C

UNICOS 8.0

Fortran I/O test results (Disk)

Table 3:

Concurrent Processes	Elapsed Seconds			User Seconds			System Seconds		
	7.C	8.0	%diff	7.C	8.0	%diff	7.C	8.0	%diff
1	164	161	-1.8	1.1	1.5	+36.4	7.2	10.4	+44.4
2	168	165	-1.8	2.2	3.0	+36.4	15.1	20.8	+37.7
4	168	164	-2.4	4.4	5.9	+34.1	33.2	41.7	+25.6
8	169	165	-2.4	8.9	11.8	+32.6	89.3	85.3	-4.5
16	219	181	-17.4	18.0	23.7	+31.7	1218.7	176.1	-85.6
32	407	194	-52.3	36.2	47.4	+30.9	4471.4	411.3	-90.8
Concurrent Processes	Total CPU Seconds			Mbytes/ Second			Phys I/O req/Second		
	7.C	8.0	%diff	7.C	8.0	%diff	7.C	8.0	%diff
1	8.3	11.8	+42.2	18.3	18.6	+1.6	295.2	300.2	+1.7
2	17.4	23.7	+36.2	35.7	36.4	+2.0	576.1	585.7	-1.7
4	37.6	47.7	+26.9	71.4	73.2	+2.5	1152.3	1178.6	-2.3
8	98.2	97.1	-1.1	142.0	145.5	+2.5	2291.6	2343.0	+2.2
16	1236.7	199.8	-83.8	219.2	265.2	+21.0	3536.7	4271.8	+20.8
32	4507.6	458.7	-89.8	235.9	494.9	+109.8	3807.6	7971.1	+109.3

Fortran I/O Disk Results

Elapsed Time

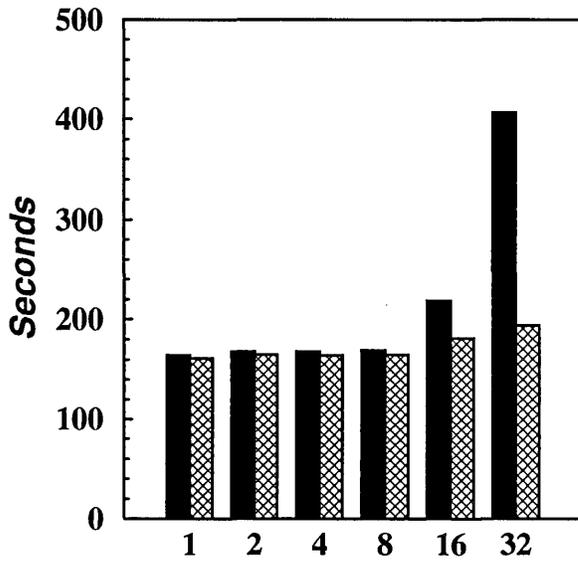


Fig 9. Concurrent Processes

User Time

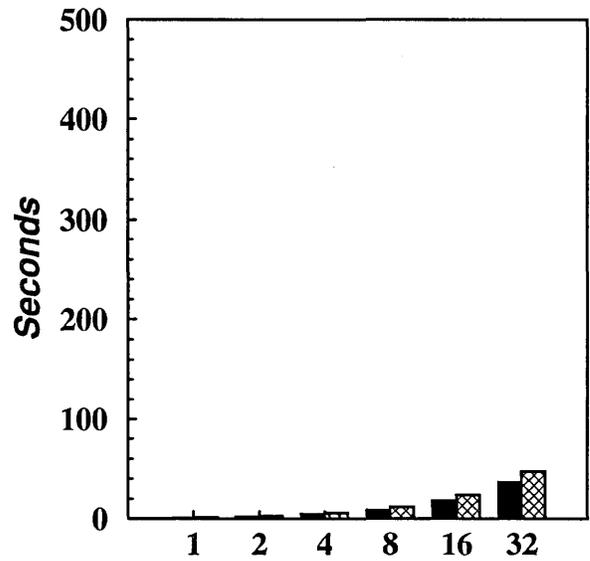


Fig 10. Concurrent Processes

System Time

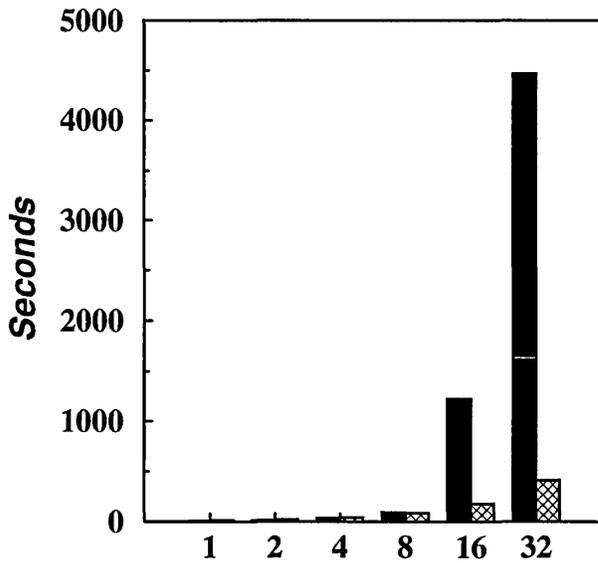


Fig 11. Concurrent Processes

Physical I/O Request Rate

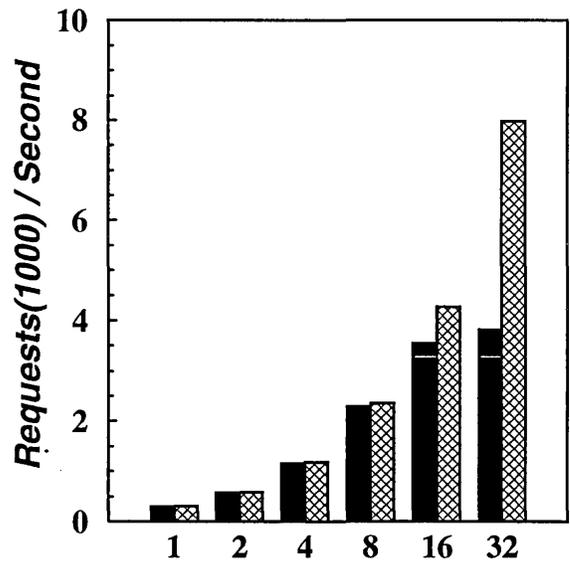


Fig 12. Concurrent Processes


UNICOS 7.C

UNICOS 8.0

Fortran Workload results

Table 4:

Processors	Elapsed Seconds			User Seconds			System Seconds		
	7.C	8.0	%diff	7.C	8.0	%diff	7.C	8.0	%diff
16	674.0	648.0	-3.9	6503.5	6616.5	+1.7	2320.5	1312.2	-43.5
Processors	Total CPU Seconds			Mbytes/ Second			Phys I/O req/Second		
	7.C	8.0	%diff	7.C	8.0	%diff	7.C	8.0	%diff
16	8824.1	7928.7	-10.1	545.2	568.2	+4.2	8627.9	8936.3	+3.6

Fortran Workload

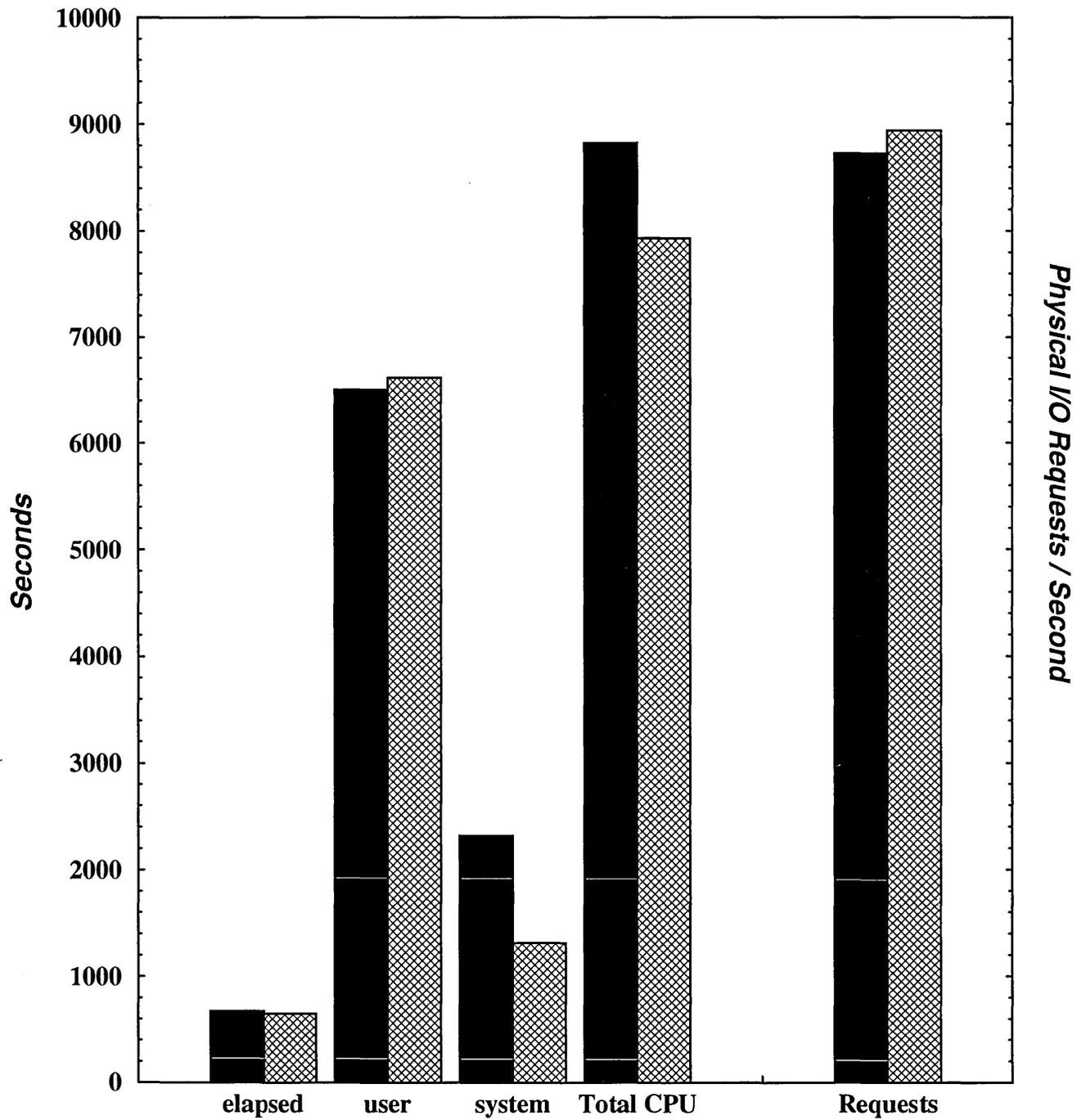


Fig 13.

■ UNICOS 7.C
▨ UNICOS 8.0

Workload Characterization of CRAY Supercomputer Systems running UNICOS for the Optimal Design of NQS Configuration in a site.

Young W. Lee, Yeong Wook Cho
KIST(Korea Institute of Science and Technology) /
SERI(Systems Engineering Research Institute)
Yoo-Sung P.O. Box #1, Yoo-Sung, Dae-Jeon, South Korea
ywlee@garam.kreonet.re.kr
ywcho@kumdori.seri.re.kr

Alex Wight
The Computer Science Department of Edinburgh University
Room 2422, JCMB of Kings Buildings, Mayfield Road,
Edinburgh, EH9 3JZ, United Kingdom
asw@dcs.ed.ac.uk

This paper discusses a workload characterization study of CRAY supercomputer systems running UNICOS operating system, of which the result was used to optimally configure NQS queues in a site in terms of system performance. A NQS configuration advisor package has been developed and implemented at KIST/SERI National Supercomputer Center as a CRAY supported work. This paper discusses a part of the work. UNICOS supplied workload records were used to generate two-dimensional workload distribution charts as advising information for optimal NQS configuration. A clustering technique using K-means non-hierarchical clustering algorithm is also applied to analyze the workload in terms of NQS queues.

1. INTRODUCTION

The NQS(Network Queueing System) is the batch processing subsystem of UNICOS.

Requests in batch queues are scheduled for initiation based on criteria configured into NQS by an system administrator. Each site should design its own NQS configuration based on its

workload demand and operation policy. How to configure NQS efficiently is a time consuming and erroneous job but also very important task to system administrators. The efficiency of NQS configuration directly affects the system performance and a coarse or erroneous NQS configuration would cost expensively thinking the price of CRAY systems. Also, a system administrator would require significant amount of system administration experience in order to configure NQS efficiently. With the experience, the person who is in charge of NQS configuration design should refer to the workload of his/her CRAY supercomputer system for certain period of time. The workload collection and analysis requires a lot of effort and time because the collected data is huge in its size. Nevertheless, good workload characterization is essential for the design of efficient NQS configuration. The person who is in charge of NQS configuration design should test the efficiency of the designed NQS configuration before implementing it into the production system. However, we could not find any available tool for these required tasks of designing efficient(optimal) NQS configuration.

In recognition of these hard tasks and situation, we developed a package for it, named "An Advisor for optimal NQS Configuration Design". We installed and have been testing the package in the National Supercomputer Center at KIST/SERI, Dae-Duck Science Town, Korea. The package analyzes collected

workload data and plots work demand on two dimensional chart in the form of NQS configuration chart. A systems administrator who is in charge of NQS configuration design can readily design his NQS configuration based on the chart. The package even generates recommended queue sectors for optimal NQS configuration through K-means non-hierarchical clustering algorithm. These functions are a good bench which systems programmers can rely on to make decision when they design their NQS configuration. The package further offers NQS configuration simulation function which enables a systems programmer to simulate his test version NQS configuration freely before implementing it into his real system so that he can select the best configuration which the site seeks. This simulation function will save a lot of operational cost, time and effort because the site can obtain the best configuration for its work demand and policy without having it implemented into its real system. The site will be free from the risk to run its NQS configuration draft directly at the production system with its valuable customers' jobs!

In this paper, we introduce two functions among the above three functions of our package with real usage experience gained at the National Supercomputer Center of KIST/SERI. Figure 1 and figure 2 show the general diagram for the first function. For further explanation, we shows a CRAY2S 4/128(4 CPUs

& 128 MW main memory) system configuration at the National Supercomputer Center of KIST/SERI which had been operated for last 5 years until November 20, 1993 in figure 3 and a CRAY C90 16/512 (16 CPUs & 512 MW main memory) supercomputer system configuration which was opened December 1, 1993 and will be operated for next 5 years in figure 4. In this paper, we show examples using workload data of November 18, 1993 in a CRAY2S system at the National Supercomputer Center of KIST/SERI (reported on November 19, 1993 by UNICOS accounting facility) and/or the workload data of January 19, 1993 in a CRAY C90 system at the National Supercomputer Center of KIST/SERI.

2. WHAT WORKLOAD DATA SHOULD BE COLLECTED FOR NQS CONFIGURATION DESIGN ?

Workload characteristics can be represented by parameters (variables). In characterizing physical resource demands for process processing, three variables are usually considered significant : CPU time demand (real time, system time, user time), memory space demand (mean, peak, total), and I/O data transferred. Real CPU time demand (system time + user time) and peak memory space demand are used for NQS queue setting and I/O data transferred variable is not used for NQS queue configuration setting. Instead I/O

tape unit usage information is used in NQS configuration setting. Our package uses real CPU time demand, peak memory space demand and some other necessary variables such as frequencies, etc. for the workload characterization of NQS configuration design.

3. HOW TO COLLECT THE REQUIRED WORKLOAD DATA?

How to collect the above workload data for NQS configuration design? Three methods are available to collect workload data. The first method is to use accounting facility provided by CRAY UNICOS system. Performance related UNIX packages can be also used. The third method is to use our own self developed kernel programs to collect workload data. Our package uses the accounting facility provided by standard CRAY UNICOS system, because we considered maintainability, portability, generality and easiness.

Our package first collects workload data which contain variables pertaining to process characteristics and then extract the necessary variables including real CPU time demand and memory space demand. Table 1 shows the format of the collected workload of November 18, 1993, which was processed by `"/bin/acctcom -fhikrt /usr/adm/acct/day/pacct"` system command. The size of the produced data was approximately 50 MBytes on the day of

November 18, 1993 on a CRAY2S super-computer system running UNICOS operating system at KIST/SERI National Supercomputer Center. Table 2 shows the processed data according to process type on March 18, 1993. The results were statistically very similar to each other.

4. CONVENTIONAL WORKLOAD CHARACTERIZATION

On a CRAY2S and a CRAY C90, we collected and analyzed these data for more than 3 months in October, November, December 1993 and January 1994, and found that the number of used process type was less than 500. The top 20 popular system commands were shown in table 3. Figure 5 shows the cumulative frequency distribution of all processes. The top 2 popular system command types (sh and echo) accounted for 46% of the total number of processes in the workload, the top 5 popular system command types (sh, echo, tpstat, grep, wc) for 72% of the total number of the processes in the workload and the top 16 popular system command types for 90% of the total number of processes in the workload. Among top 20 processes, there was no user process type but system command types.

Figure 6 shows the cumulative real CPU time distribution of all process types. The top process consumed 306.62 minutes (7.4%

among 4146.85 minutes), the top 10 processes consumed 49% of the total CPU time and the top 20 processes consumed 69% of the total CPU time. Most of system command types consumed less than 1 second CPU time per each.

Figure 7 shows mean memory size distribution of all process types. The largest one occupied 44% (57 MW) of the global memory space. Each of the top 6 processes occupied more than 10% (13 MW) of the global memory space. The bottom one occupied only 19,240 Word (0.015%) of the global memory space. Most of system command types occupied less than 100 KW of the global memory space.

Figure 8 shows the utilization of the CRAY2S on November 18, 1993 at the National Supercomputer Center of KIST/SERI and figure 9 shows the average daily utilization of the CRAY2S from October 25 to November 18, 1993.

5. TWO DIMENSIONAL WORKLOAD CHART

There can be several ways to analyze the collected workload. As we mentioned in introduction section, it is required to analyze the collected workload in order to design optimal NQS configuration. We found, 2 dimensional workload plotting method, what we are going to explain below, is very helpful for

it. Figure 10 shows the old NQS configuration of the CRAY2S at KIST/SERI on 2 dimensional chart before we applied our package for NQS configuration design. We have 16 general purpose NQS queues and 6 special purpose queues. The interactive jobs have been defined to have 3000 second real cpu time limit and 4 MW maximum memory limit. Excluding the I/O tape unit usage information, we can specify the physical resource requirement of NQS queues well in the 2 dimensional chart. Hence, we become to have an idea that, if the physical resource demands of each process can be characterized in the form of 2 dimensional NQS configuration chart, the systems programmer can readily design the required NQS configuration.

Figure 11 shows the output chart by using plotting function of our package, that is, by issuing "acctanal_1 acctcom.1119 -f" command (the workload data used were collected during March 18, 1993 and reported on March 19, 1993 by UNICOS accounting facility). The x axis denotes memory size(mean size in KW) occupied by each process and y axis denotes cpu time(real time in second) consumed for each process. A denotes 100 and therefore 2A means 200 processes. B denotes 1,000 and therefore 5B means 5,000 processes. C denotes 10,000 and therefore 6C means 60,000 processes. D denotes 100,000 and therefore 4D means 400,000 processes. Comparing the 2 dimensional workload chart with the 2 dimen-

sional NQS configuration chart of figure 10, it could be found that our approach is very useful for NQS configuration design.

Our package provides zooming facility. If it is required to close up a window area, then by zooming the specified window area, users can close up the selected window area in detail. Figure 12 shows a sample chart. Our package can process both full process workload data obtained by "acctcom" procedure (figure 11) and reduced workload data according to command type by "acctcms" procedure (figure 13).

6. CLUSTERING

Clustering techniques has been often used to analyze workload data, especially to find input distribution and related values to drive performance models. Our package provides clustering function to produce recommended NQS queues as well as the above purpose. CRAY UNICOS System administrators will have better chance to design optimal NQS configuration by using the clustering function of our package together with 2 dimensional workload chart plotting function of our package.

Our package uses an adopted K-means non-hierarchical clustering algorithm ([ANDERBERG 73], [HARTIGAN 75], [SPÄTH 82]). The process types were regarded as points in the 2 dimensional chart(space). The essence of the

technique can be explained in 4 steps. First, it assigns the initial values to pre-defined number of clusters(say k), that is, selects the first k data from all data. Second, it takes the initial values as the centroid of each mass and using it, assigns the next data to clusters which have least Euclidean distance in the 2 dimensional chart(space) between each component and the centroid of its cluster. Third, it recalculates the centroid of each cluster and assigns the next data to cluster which have least Euclidean distance in the 2 dimensional chart(space) between each component and the centroid of its cluster. Fourth, it repeats the second step and the third step until the member of each cluster does not change, that is, the members of each cluster becomes stable. The pre-defined number of cluster or the value of k is varied. The goodness of a partition is tested based on SSE (Sum of Squared Error) and the best K is selected.

The clustering function accepts workload data classified to the process type, which is produced by "acctcom" procedure and generates the following attributes.

- Elements of each cluster
- Frequency of each cluster
- Mean(Centroid) of each cluster
- Minimum and maximum values of each cluster
- Standard deviation from Centroid of each

cluster

- Median
- Sum of squared error from median of each cluster

Figure 14 shows a sample output from a CRAY2S data of November 18, 1993 in the National Supercomputer Center of KIST/SERI. We show the recommended NQS queue in 2 dimensional chart using this output in figure 15. By overlapping figure 15 onto figure 13, system administrators can find workload information in each recommended queue. This will enable system administrators to have better insight to design their own optimal NQS configurations.

7. CONCLUSIONS

Up to now, system administrators have relied on their experience and feeling in designing their NQS configuration. This kind of non-scientific approach, so called, art approach to design NQS configuration should be no longer allowed, because the CRAY super-computer systems are very expensive machine and the NQS configuration directly and greatly affects the performance of CRAY systems. In this paper, we showed how our package can effectively analyze workload in terms of NQS queues. We further showed the clustering function of our package suggests recommended NQS queues for optimal NQS configuration using given workload. Using our approach, the system administrators can

effectively configure their NQS queues according to their workload characteristics and their operation policy. We hope other sites share our experience and package for their NQS configuration setup. Anyone who interested in this package will be welcomed by Young W. LEE at the email address of "ywlee@garam.seri.re.kr". We will send the introduced modules and documents to him/her. In this paper, we have not introduced the simulation function of our package which enables system administrators to simulate his test version NQS configuration freely without implementing it into his/her real production system so that he/she can select the best configuration which the site requires. We leave it for another opportunity.

ACKNOWLEDGEMENTS

This research would not have been possible without the help of the following individuals : Mr. Yoon, Sang Hoon at the National Supercomputer Center of KIST/SERI and Mr. Park, Sung Won at CRAY Research Inc., Korea. We deeply appreciate their help.

This work was funded by CRAY Inc. under Cooperative Agreement No. KIST/SERI-D003200.

REFERENCES

- [ANDERBERG 73] Michael R. ANDERBERG, "Cluster Analysis for applications", Academic Press, Inc. 1973
- [HARTIGAN 75] John A. HARTIGAN, "Clustering Algorithms", John Wiley & Sons, Inc., 1975
- [SPÄTH 82] Helmuth. SPÄTH, "Clustering analysis algorithms", John Wiley & Sons, Inc., 1982

FIGURES AND REFERENCES

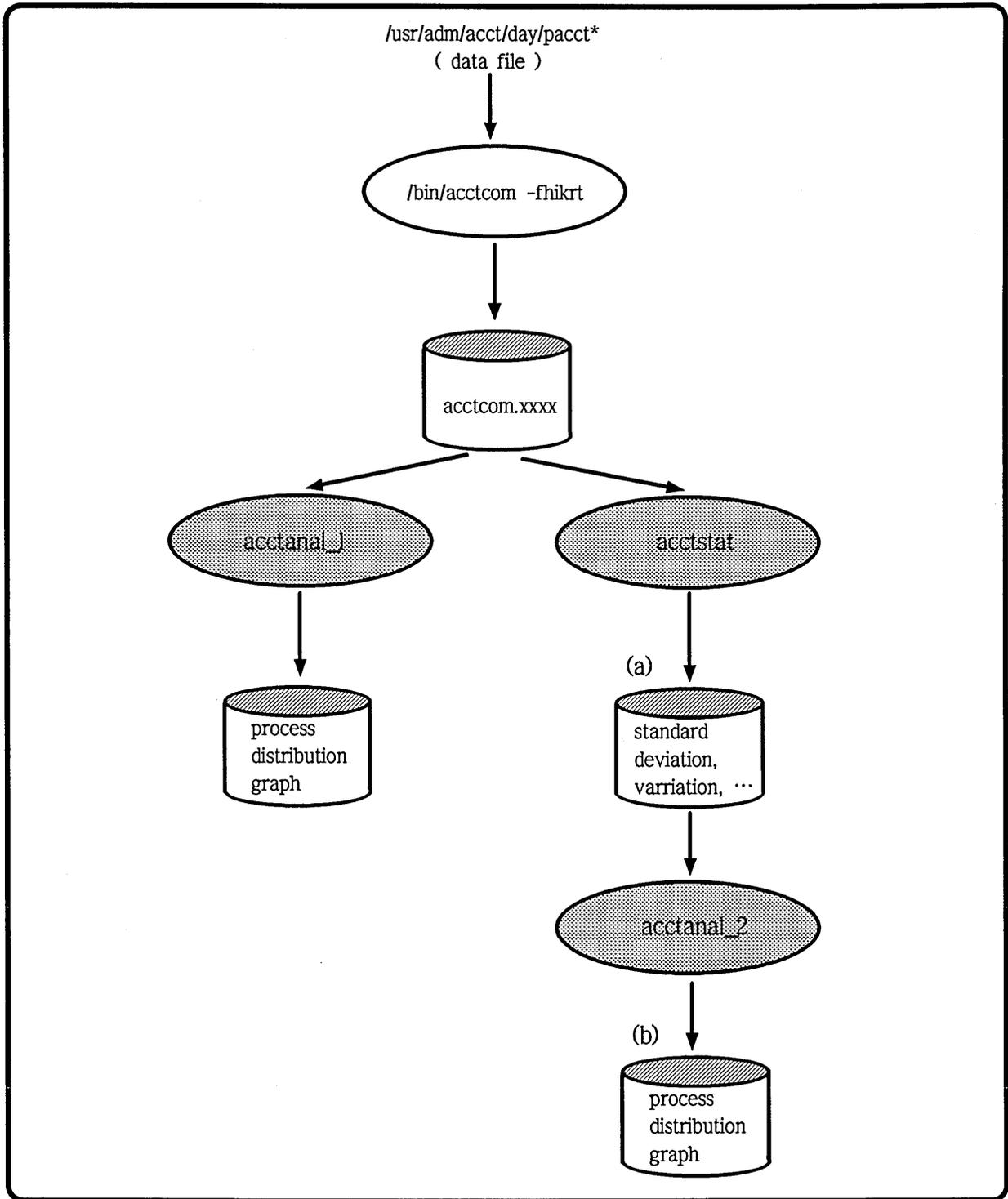


Figure 1

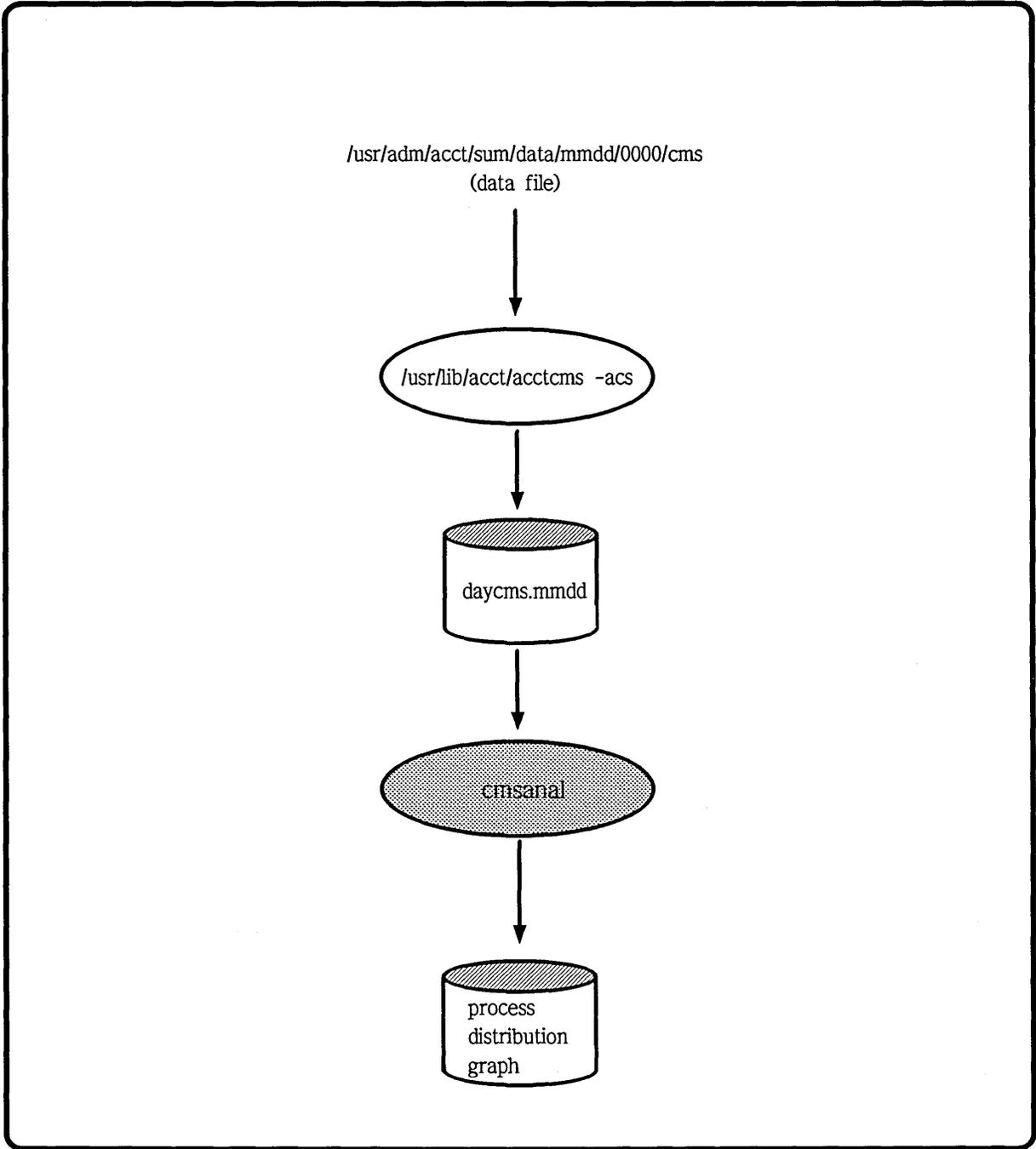


Figure 2

KIST CRAY-2S SYSTEM CONFIGURATION

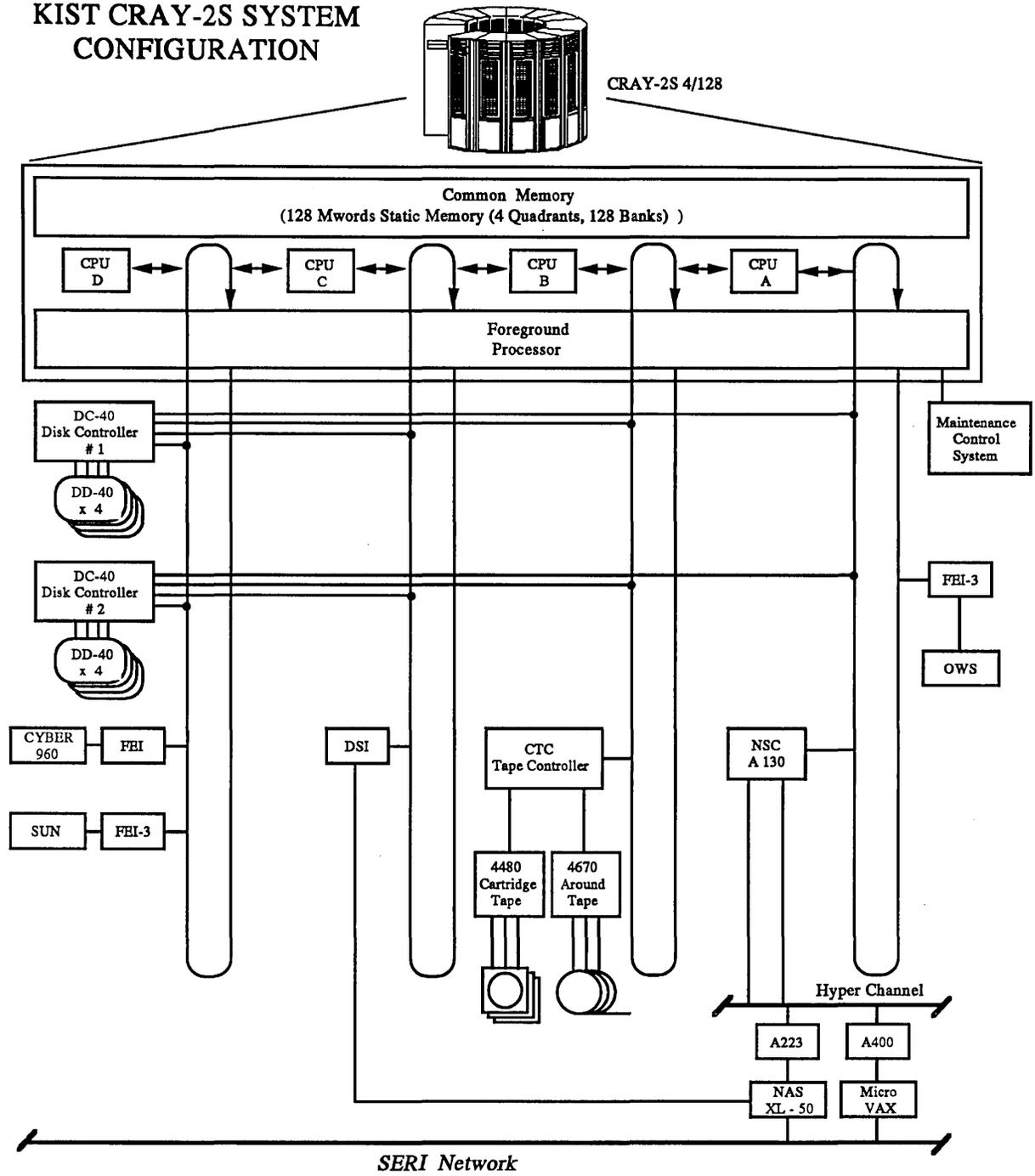
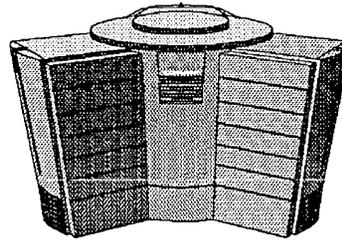


Figure 3

KIST CRAY Y-MP C90 SYSTEM CONFIGURATION



C90 / 16512

Figure 4

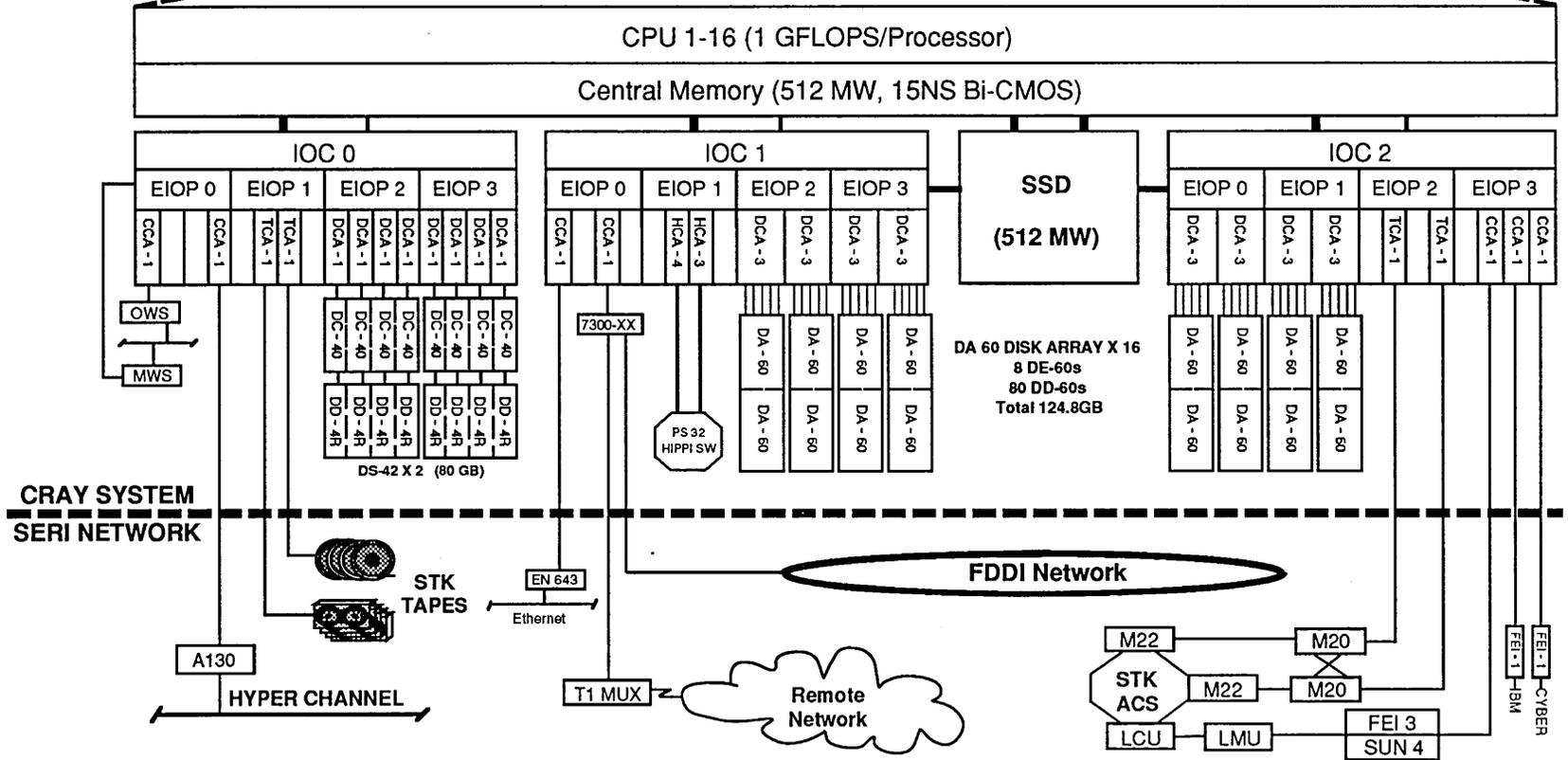


Table 1

ACCOUNTING RECORDS FROM: Fri Nov 17 00:00:03 1993

COMMAND	START	END	REAL	CPU	(SECS)	CHARS	PHYS	CPU	HOG	KCORE		
NAME	USER	TTYNAME	TIME	TIME	(SECS)	TRNSFD	REQS	FACTOR	FACTOR	MIN	F	STAT
#accton	root	?	00:00:09	00:00:09	0.02	0.01	0.00	16392	0	0.27	0.48	0.00 2 0
echo	root	?	00:00:09	00:00:09	0.01	0.01	0.00	0	1	0.32	0.58	0.00 0 0
sh	adm	?	00:00:08	00:00:08	0.84	0.02	0.03	981	20	0.55	0.06	0.04 1 0
echo	root	?	00:00:09	00:00:09	0.01	0.00	0.00	0	1	0.32	0.65	0.00 0 0
echo	root	?	00:00:09	00:00:09	0.02	0.01	0.00	0	1	0.32	0.39	0.00 0 0
sh	root	?	00:00:09	00:00:09	0.00	0.00	0.00	0	0	0.65	0.58	0.00 1 0
tpstat	root	?	00:00:09	00:00:09	0.05	0.01	0.01	2668	7	0.35	0.34	0.01 0 0
grep	root	?	00:00:09	00:00:09	0.06	0.00	0.00	716	2	0.47	0.16	0.01 0 0
wc	root	?	00:00:09	00:00:09	0.24	0.01	0.00	9	4	0.34	0.06	0.01 0 0
sh	root	?	00:00:09	00:00:09	0.00	0.00	0.00	0	0	0.63	0.85	0.00 1 0
echo	root	?	00:00:09	00:00:09	0.01	0.01	0.00	0	1	0.31	0.95	0.00 0 0
echo	root	?	00:00:09	00:00:09	0.01	0.00	0.00	0	1	0.32	0.95	0.00 0 0
echo	root	?	00:00:09	00:00:09	0.01	0.01	0.00	0	1	0.32	0.64	0.00 0 0
sh	root	?	00:00:09	00:00:09	0.00	0.00	0.00	0	0	0.65	0.56	0.00 1 0
sh	operator	ttyp005	00:00:09	00:00:09	0.03	0.01	0.00	0	0	0.32	0.39	0.01 0 0
tpstat	operator	ttyp003	00:00:09	00:00:09	0.06	0.01	0.01	2668	7	0.32	0.32	0.01 0 0
tpstat	root	?	00:00:09	00:00:09	0.06	0.01	0.01	2668	6	0.35	0.33	0.01 0 0
sh	operator	ttyp003	00:00:09	00:00:09	0.16	0.01	0.01	0	0	0.31	0.10	0.01 0 0
grep	root	?	00:00:09	00:00:09	0.06	0.00	0.00	716	2	0.47	0.16	0.01 0 0
wc	root	?	00:00:09	00:00:09	0.17	0.01	0.00	9	4	0.34	0.08	0.01 0 0
sh	root	?	00:00:09	00:00:09	0.00	0.00	0.00	0	0	0.63	0.81	0.00 1 0
echo	root	?	00:00:09	00:00:09	0.01	0.01	0.00	0	1	0.31	0.95	0.00 0 0
echo	root	?	00:00:09	00:00:09	0.01	0.00	0.00	0	1	0.32	0.95	0.00 0 0
echo	root	?	00:00:09	00:00:09	0.01	0.01	0.00	0	1	0.31	0.95	0.00 0 0

Table 2

COMMAND NAME	NUMBER CMDS	TOTAL KCOREMIN	TOTAL COMMAND SUMMARY							
			TOTAL CPU-MIN	TOTAL CONN-MIN	TOTAL REAL-MIN	MEAN SIZE-K	MEAN CPU-MIN	HOG FACTOR	K-CHARS TRNSFD	I/O BUFS RD/WR
TOTALS	85968	101065215.15	10046.40	10046.40	139744.45	10059.84	0.12	0.07	738232753	7704147
a.out	205	9774474.34	1861.22	1861.22	1932.05	5251.66	9.08	0.96	1370040	40882
standard	51	2173513.16	685.42	685.42	1387.67	3171.06	13.44	0.49	273048404	3418696
discover	2	743983.79	647.06	647.06	85.82	1149.79	323.53	7.54	848488	3518
l9999.ex	31	60252425.60	633.04	633.04	664.99	95179.96	20.42	0.95	186603915	486596
opt_b128	3	188866.58	590.21	590.21	605.49	320.00	196.74	0.97	2026045	58326
tnd	1	4206828.66	501.77	501.77	501.82	8384.06	501.77	1.00	22	20
crfree.x	6	400925.58	468.37	468.37	468.47	856.00	78.06	1.00	8970	620
tfq	1	773434.03	389.83	389.83	399.86	1984.03	389.83	0.97	2005920	126760
run3	1	2335266.13	377.14	377.14	382.15	6191.97	377.14	0.99	16356352	99928
run4	1	2165637.12	349.75	349.75	354.93	6192.00	349.75	0.99	16356352	100096
run2	1	2165018.17	349.65	349.65	355.61	6191.90	349.65	0.98	16356352	100352
run1	1	2159484.02	348.76	348.76	354.28	6191.90	348.76	0.98	16356352	100424
relap5.x	11	312446.34	285.08	285.08	285.65	1095.98	25.92	1.00	370197	2174
cyl4c	21	237473.78	243.77	243.77	152.66	974.16	11.61	1.60	13656	1863
p94a_03d	18	666095.51	193.54	193.54	193.55	3441.63	10.75	1.00	69746764	933444
cpc.r	2	127567.64	187.60	187.60	187.70	679.99	93.80	1.00	2323	69
p4_05.e	25	550560.50	143.42	143.42	139.36	3838.77	5.74	1.03	3712951	54814
crf	1	518426.17	134.73	134.73	134.75	3848.00	134.73	1.00	3525	101
opt_fpf	2	40082.47	131.85	131.85	138.71	304.00	65.93	0.95	450927	23018
AMIP_1	10	214100.39	122.31	122.31	124.96	1750.41	12.23	0.98	1533598	16299
p4_04.e	28	445766.33	117.94	117.94	113.09	3779.53	4.21	1.04	1439711	20496

Table 3

*** TOTAL FREQUENCY RATIO ***

* COMMAND *	* FREQUENCY *	* ACCUM_FREQ *	* RATIO *	* ACCUM_RATIO *
sh	95299	95299	(23.83935 %)	(23.83935 %)
echo	88274	183573	(22.08203 %)	(45.92138 %)
tpstat	37566	221139	(9.39726 %)	(55.31863 %)
grep	37088	258227	(9.27768 %)	(64.59632 %)
wc	29167	287394	(7.29622 %)	(71.89253 %)
qstat	18902	306296	(4.72840 %)	(76.62093 %)
rm	10196	316492	(2.55056 %)	(79.17149 %)
cut	9110	325602	(2.27890 %)	(81.45039 %)
assign	5594	331196	(1.39936 %)	(82.84975 %)
cat	5146	336342	(1.28729 %)	(84.13703 %)
expr	4860	341202	(1.21574 %)	(85.35278 %)
ls	4655	345857	(1.16446 %)	(86.51724 %)
cp	3807	349664	(0.95233 %)	(87.46958 %)
line	3280	352944	(0.82050 %)	(88.29008 %)
date	2644	355588	(0.66141 %)	(88.95148 %)
csh	2385	357973	(0.59662 %)	(89.54810 %)
awk	2285	360258	(0.57160 %)	(90.11970 %)
sed	2074	362332	(0.51882 %)	(90.63852 %)
vi	1746	364078	(0.43677 %)	(91.07528 %)
setvar	1615	365693	(0.40400 %)	(91.47928 %)
mv	1598	367291	(0.39974 %)	(91.87903 %)

CRAY-2S FREQUENCY DISTRIBUTION
(Nov. 18, 1993)

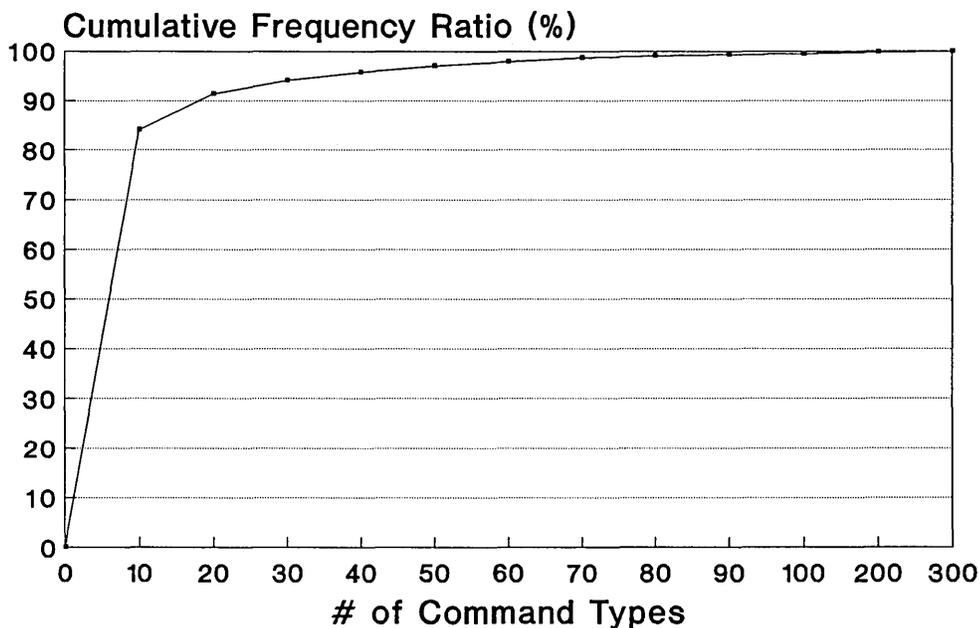


Figure 5

CRAY-2S CPU TIME DISTRIBUTION

(Nov. 18, 1993)

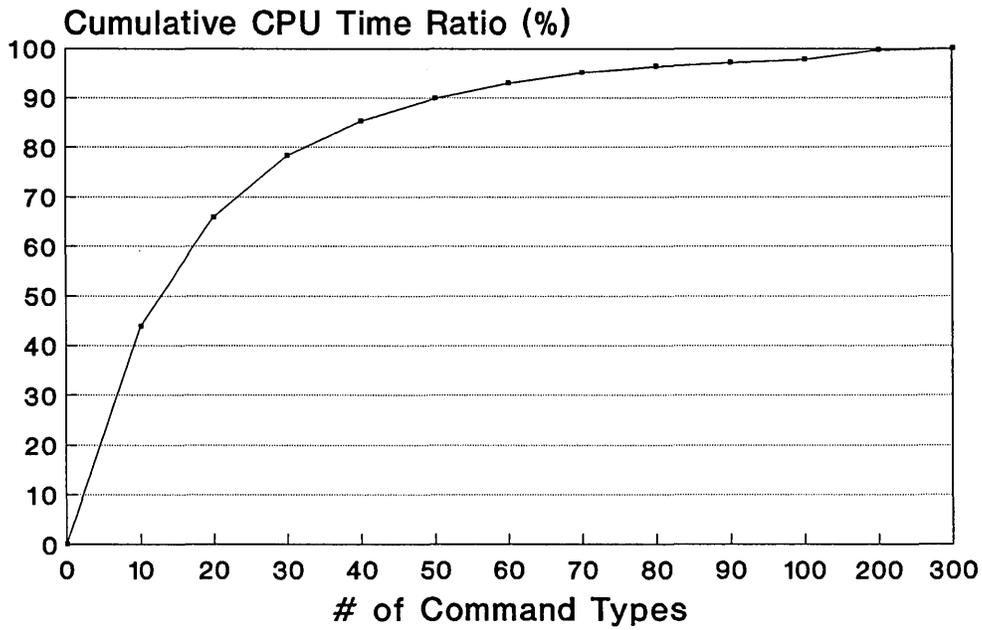


Figure 6

CRAY-2S MEAN MEMORY DISTRIBUTION

(Nov. 18, 1993)

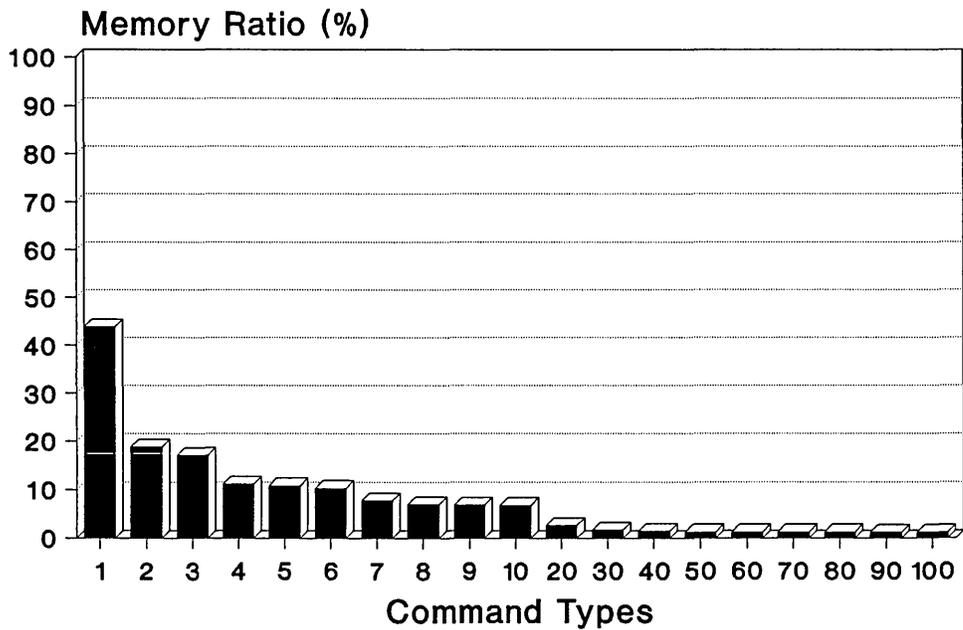


Figure 7

CRAY-2S SYSTEM UTILIZATION

(Nov. 18, 1993)

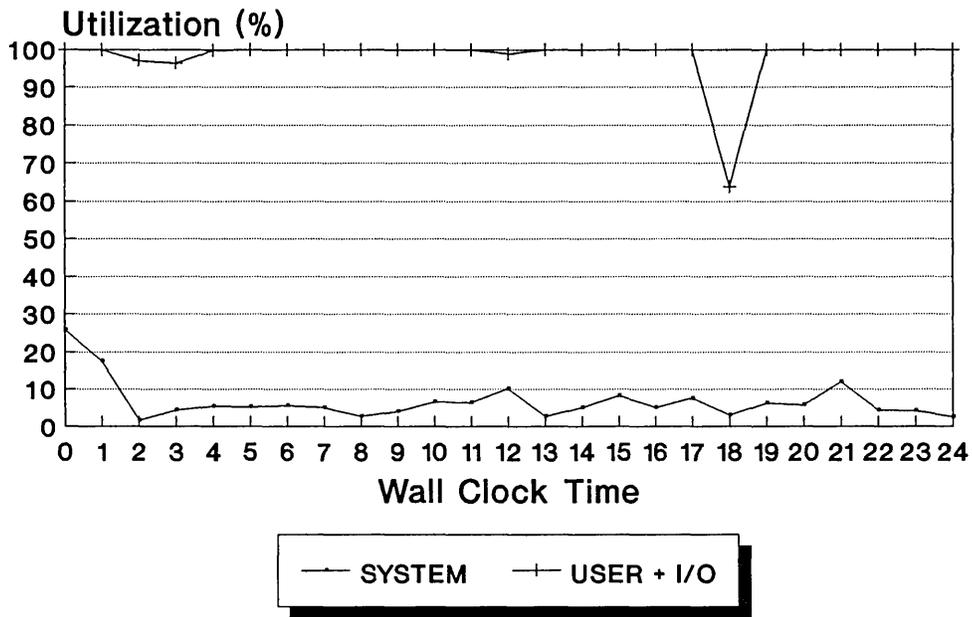


Figure 8

CRAY-2S SYSTEM UTILIZATION

(Oct. 25, 1993 - Nov. 18, 1993)

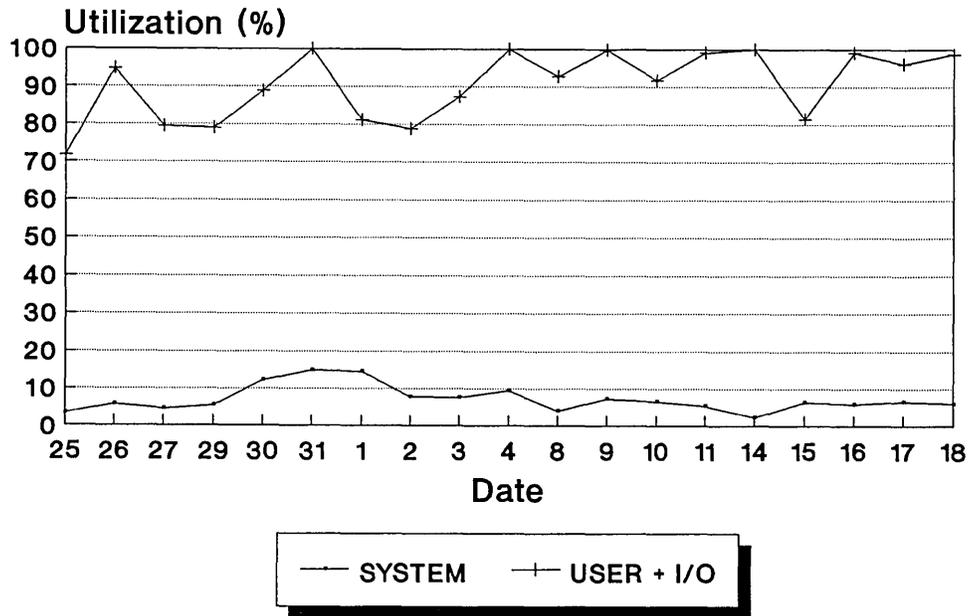


Figure 9

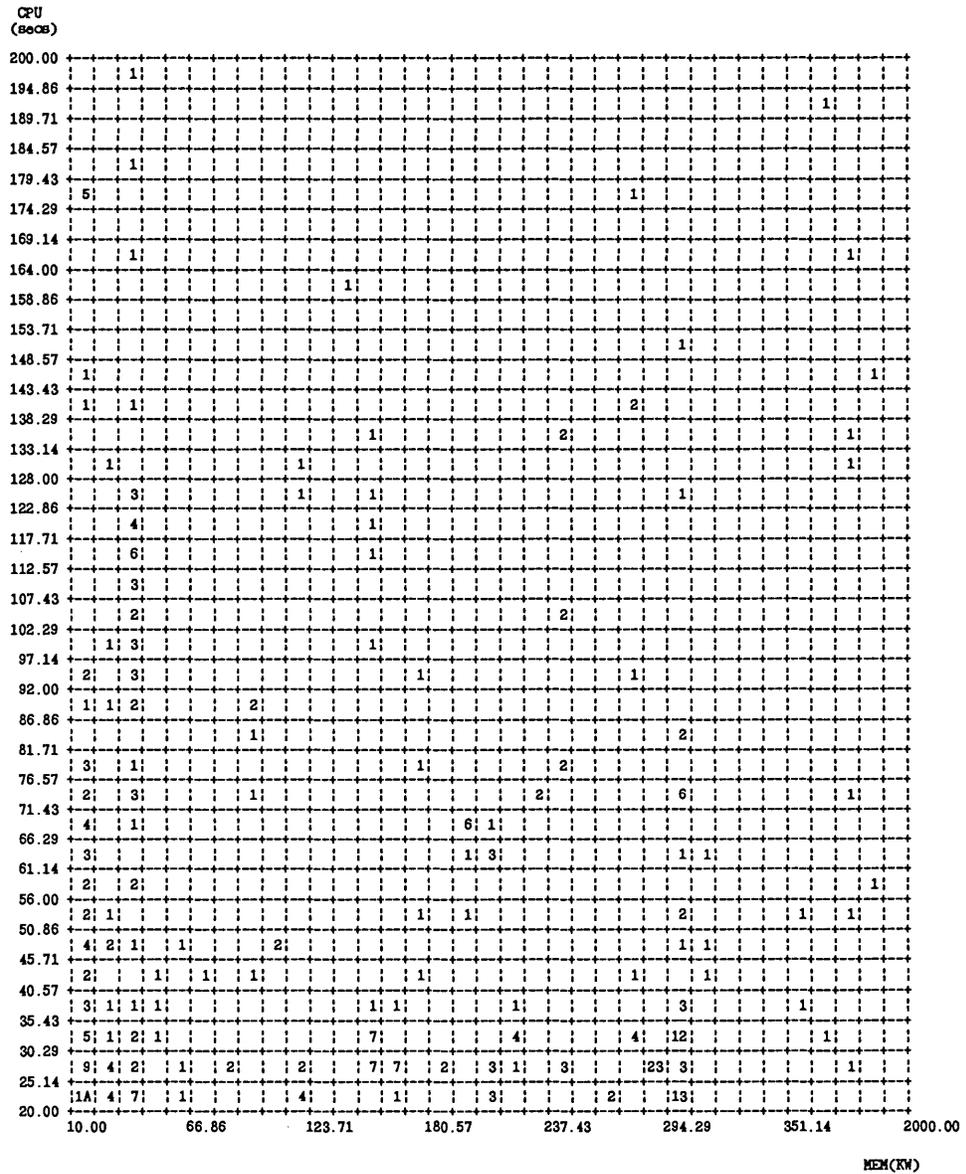


Figure 12

***** Final Clustering Results (data_size = 517) *****

The Cluster Size : 10 !!

```

[ 1 ] : 451 ;; 23 24 25 26 34 36 37 38 44 47
          49 50 51 54 55 56 57 58 59 62
          65 66 67 68 71 72 73 75 76 77
          78 79 80 81 83 84 86 87 88 89
          91 92 93 94 95 96 97 98 99 100
          101 102 103 104 107 108 109 110 111 112
          113 114 116 117 118 122 123 124 125 126
          127 128 129 130 131 132 133 134 135 137
          138 139 140 141 142 143 144 145 146 147
          148 149 150 151 152 153 154 155 156 157
          158 159 160 161 162 163 164 165 166 167
          168 169 170 171 172 173 174 175 176 177
          178 179 180 181 182 183 184 185 186 188
          189 190 191 192 193 194 195 196 197 198
          199 202 203 204 205 206 207 208 209 210
          211 212 213 214 215 216 217 218 219 220
          221 222 223 224 225 226 227 228 229 230
          231 232 233 234 235 236 237 238 239 240
          241 242 243 244 245 246 247 248 249 250
          251 252 253 254 255 256 258 259 260 261
          262 263 264 265 266 267 268 269 270 271
          272 273 274 275 276 277 278 279 280 281
          282 283 284 285 286 287 288 290 291 293

          294 295 296 298 299 300 301 302 304 305
          306 307 308 309 310 311 312 313 314 315
          316 317 318 319 320 321 322 323 324 325
          326 327 328 329 330 331 332 333 334 335
          336 337 338 339 340 341 342 343 344 345
          346 347 348 349 350 351 352 353 354 355
          356 357 358 359 360 361 362 363 364 365
          366 367 368 369 370 371 372 373 374 375
          376 377 378 379 380 381 382 383 384 385
          386 387 388 389 390 391 392 393 394 395
          396 397 398 399 400 401 402 403 404 405
          406 407 408 409 410 411 412 413 414 415
          416 417 418 419 420 421 422 423 424 425
          426 427 428 429 430 431 432 433 434 435
          436 437 438 439 440 442 443 444 445 446
          447 448 449 450 451 452 453 454 455 456
          457 458 459 460 461 462 463 464 465 466
          467 468 469 470 471 472 473 474 475 476
          477 478 479 480 481 482 483 484 485 486
          487 488 489 490 491 492 493 494 495 496
          497 498 499 500 501 502 503 504 505 506
          507 508 509 510 511 512 513 514 515 516
          517

[ 2 ] : 1 ;; 5
[ 3 ] : 6 ;; 11 12 13 14 15 18
[ 4 ] : 5 ;; 3 21 52 53 441
[ 5 ] : 4 ;; 4 7 9 16
[ 6 ] : 4 ;; 40 42 70 136
[ 7 ] : 9 ;; 1 2 6 8 28 29 31 39 45
[ 8 ] : 19 ;; 17 27 63 64 85 105 106 115 119 120
          121 187 200 201 257 289 292 297 303
[ 9 ] : 10 ;; 10 19 20 30 32 33 35 41 43 61
[ 10 ] : 8 ;; 22 46 48 60 69 74 82 90

```

Ind	Size	Freq.	Minimum	Maximum	Centroid	Std_dev	Median
1	451	399400	(19.24 , 0.00)	(1588.52 , 1.99)	(483.92 , 0.12)	(99.34 , 0.01)	(704.48 , 0.30)
2	1	2	(56146.94 , 92.06)	(56146.94 , 92.06)	(56146.94 , 92.06)	(56146.94 , 92.06)	(0.00 , 0.00)
3	6	6	(111.50 , 80.67)	(3843.77 , 109.89)	(1087.28 , 99.22)	(112.00 , 103.20)	(1752.25 , 10.06)
4	5	22	(13212.72 , 0.00)	(24031.75 , 21.24)	(17498.70 , 9.60)	(14385.01 , 11.87)	(5517.81 , 8.02)
5	4	10	(1350.50 , 48.12)	(8946.37 , 62.32)	(3564.98 , 55.28)	(2382.55 , 59.94)	(3346.35 , 7.58)
6	4	29	(8505.03 , 0.57)	(9888.04 , 22.87)	(8944.63 , 6.49)	(8727.89 , 1.73)	(591.73 , 10.60)
7	9	39	(111.00 , 19.37)	(5074.76 , 37.68)	(2467.61 , 26.90)	(2170.29 , 24.38)	(1425.03 , 6.28)
8	19	173	(1722.12 , 0.01)	(5020.46 , 4.76)	(2871.07 , 0.81)	(2400.06 , 0.62)	(1146.54 , 1.12)
9	10	44	(111.99 , 6.24)	(2007.10 , 17.80)	(1216.33 , 11.75)	(1493.24 , 11.28)	(785.34 , 3.59)
10	8	30	(24.00 , 2.75)	(745.53 , 8.07)	(253.54 , 4.22)	(43.50 , 3.42)	(358.90 , 1.97)

Figure 14

CRAY C90D PERFORMANCE

David Slowinski

Cray Research, Inc.
Chippewa Falls, Wisconsin

What is a CRAY C90D? Cray Research, Inc. (CRI) is most famous for vector computers built with the fastest memory technology available. For every new generation of systems CRI has designed numerous enhancements to add new capabilities, improve reliability, reduce manufacturing costs, and increase memory size. The CRAY-1 M, CRAY X-MP/EA, and CRAY Y-MP M90 are all computer systems that use denser, less expensive memory components to achieve big memory systems. The CRAY C90D computer is our big memory version of the CRAY C90 system with configurations that go up to 8 processors and 2 gigawords of memory.

C90D Configurations

C92D	2/512
C94D	4/1024
C98D	8/2048

How is the CRAY C90D different from a CRAY C90? The memory module is a new design that incorporates C90 circuit technology, 60 nanosecond 16 megabit commodity DRAM memory components, and the lessons learned on the successful CRAY Y-MP M90 series. There are minor changes to power distribution to handle the voltages required for the DRAM chips. And, we made changes to the CRAY C90 CPU to allow it to run with the new memory design. The new Revision 6 CPU includes CRI's latest reliability enhancements and can run in either a C90 or C90D system. Everything else is the same! This greatly reduces the costs and risks of introducing a new system for manufacturing and customer support. The C90D is bit compatible with the C90 supercomputer and runs all supported C90 software.

How is performance affected? The good news is that the C90D subsection busy time is the same as the C90 supercomputer. I wish I knew how to build a less expensive, more reliable, bigger memory system that is faster than a C90 system. Unfortunately, using a slower memory chip means a longer memory access time and less memory bandwidth. In my view, the C90D does have a good "balance." But, of course, every code would run faster with faster memory.

Performance (in clock periods)

	CRAY Y-MP	CRAY Y-MP M90	CRAY Y-MP C90	CRAY Y-MP C90D
Subsection Busy	5	15	7	7
Bank Busy	4	20	6	28
Scalar Latency	17	27	23	35

Livermore Loops

<u>Loop</u>	<u>Application</u>	<u>C94D/C916</u>
1	Hydro Fragment	.92
2	Incomplete Cholesky-CG	.81
3	Inner product	.83
4	Banded linear equations	.76
5	Tri-diagonal elimination	.76
6	General linear recurrence	.66
7	Equation of state	.95
8	A.D.I. integration	.69
9	Integrate predictors	.85
10	Difference predictors	.83
11	First sum, partial sums	.85
12	First difference	.92
13	2D particle in cell	.79
14	1D particle in cell	.93
15	Casual FORTRAN	1.03
16	Monte Carlo search	.76
17	Implicit, conditional computation	.87
18	2D explicit hydrodynamics	.85
19	Linear recurrence	.87
20	Discrete ordinates transport	.82
21	Matrix multiply	.71
22	Planckian distribution	.49
23	2D implicit hydrodynamics	.97
24	First minimum	1.01
	<i>Harmonic mean</i>	.80

Performance of standard benchmarks. There is a wide variation in performance depending on the code. The harmonic mean average performance of the loops relative to the CRAY Y-MP C916 is .80, but there is no single performance number that tells the whole story.

Let's look at these numbers a little closer. The scalar loops (5, 11, 16, 17, 19, and 20) are slower than the BiCMOS memory in the C90 computer as we might guess. Loops 13 and 14 use gather; these do relatively well due to the fast subsection busy time. Many factors affect the performance of the vector loops including memory demands of the code, access patterns, and data allocation.

**PERFECT Club Benchmarks
Single CPU Optimized Versions**

Code	D94/C916	Code	D94/C916
adm	.78	qcd	.72
arc2d	.89	trfd	.90
flo52	.80	dysesm	.77
ocean	.82	spice	.59
spec77	.77	mg3d	.89
bdna	.86	track	.89
mdg	.97	Average	.85

The performance of the C90D computer on the PERFECT Club benchmarks is similar to what we see on many customer programs. There is a wide range of performance depending on each code's demands on memory. Some, like *mdg*, run nearly as fast on the C90D as on the C90 supercomputer. Others, like *spice*, suffer from the longer memory latency.

Why big memory? I see three main reasons:

Run big jobs. Some jobs are simply too big to run on current systems. Even though there may be only a few big jobs these are often the ones poised to make new breakthroughs and advance the state-of-the-art. In my view, the justification for supercomputers is their ability to solve big problems and big memory is the feature that allows us to attempt unsolved problems. We cannot advance science by rerunning yesterday's jobs with better price performance.

Reduced coding complexity. Yes, it is theoretically possible to run any problem using only a small memory. I have heard one of the original authors of Nastran tell stories about running out of core problems with paper tape. It IS possible, but there is a high cost in coding complexity. Out-of-core solutions may require many months or even years to implement and they add nothing to the quality of the solution.

Improved performance. This may come in many different ways. Reduced I/O latency and system overhead may improve wall clock performance by an order of magnitude in some cases. The Cray EIE I/O library effectively uses large memory buffers to significantly reduce I/O wait time and system overhead for many programs without any user code changes. The biggest speedups come when big memory allows us to use a more efficient algorithm.

An exciting example of using a more efficient algorithm for large memory systems was implemented in Gaussian-92 from Gaussian Incorporated. Gaussian-92 is a computational chemistry code used by over one hundred Cray customers. Gaussian-92 has the ability to use different algorithms depending on the characteristics of the computer it is running on. The time consuming parts of a Gaussian-92 run are often the calculations of the 2-electron integrals.

The original "conventional" scheme generates the integrals once, writes them to disk, and then reads them for each iteration of a geometry optimization. This method uses the least floating-point operations and little memory but requires big disk files and lots of I/O.

The "direct" method was developed for vector computers and was released in Gaussian-88. It recomputes the integrals as they are needed. This method uses little memory and I/O, but requires lots of floating-point operations to recompute the integrals. The direct method is optimal for vector computers. The direct method also allows RISC workstations to run problems that previously could only be attempted using supercomputers.

With enough memory we could compute the integrals just once and save them in memory. This "incore" method was first developed for the big memory CRAY-2 systems and was released in Gaussian-90.

**Gaussian-92 Rev. C Benchmark
mp2=(fc/6-311+g(3df,2p))**

	Basis Functions	Direct CPU seconds 64 MW	Incure CPU seconds 1700 MW
Molecule A	249	8127	2121
Molecule B	240	7097	1712

Here are the timings for two real customer problems run on a CRAY M98 with 4 gigawords of memory using both the direct and incore methods. The direct runs used 64 million words of memory and would not benefit from more memory. The incore runs needed 1700 million words of memory. Using big memory with a more efficient algorithm for these problems gains about a factor of four performance improvement over the best performing algorithm with small memory.

I believe there are similar opportunities for significant speed increases in many other applications programs.

Conclusion. The cost for a million words of memory has dropped by a factor of 50,000 since the first CRAY-1 computer. With the many benefits of big memory and continuing dramatic improvements in memory cost, big memory will certainly be an important feature of future supercomputers.

Cray File Permission Support Toolset

David H. Jennings & Mary Ann Cummings

Naval Surface Warfare Center / Dahlgren Division
Code K51
Dahlgren, VA 22448

ABSTRACT

The Cray UNICOS Multilevel Security (MLS) Operating System supports the use of Access Control Lists (ACLs) to control permissions to directories and files. However, the standard commands (spacl, spget, spset) are difficult to use and do not allow all the capabilities needed in a multi-user environment. This paper describes a toolset which enhances the standard UNICOS commands by easily allowing multiple users the ability to give permissions to a directory or file (or multiple directories or files), even if they are not the owners.

1. Introduction

The Systems Simulation Branch (K51) is responsible for the design and programming of large software models which are typically composed of hundreds of C and FORTRAN source files. The source, header, and executable files are divided into separate directories, and often files from other directories are needed to execute the entire model. The Cray File Permission Support Toolset was designed to aid the K51 users in using the UNICOS¹ Access Control Lists to control file accesses to these large models. It is general enough that any user of the system can benefit from using this toolset for access control of any number of files or directories.

The Cray File Permission Support Toolset was developed because of a need for a development environment where a set of files and directories were created and modified by multiple developers. Other users also needed access to some of the files and directories. Also, in our environment the development teams and users of one set of files may need access to other sets as well. The traditional UNIX² permission scheme and the added ACL permission scheme provided with our UNICOS MLS system did not provide the functionality that was needed.

2. Terminology

Before we begin our discussion of the toolset, the following terms must be defined: ACL, project, model, POC, development group, and users.

- In a UNICOS MLS environment, an *Access Control List*

1. UNICOS is a registered trademark of Cray Research, Inc.
2. UNIX is a registered trademark of AT&T.

or *ACL* provides an additional level of permission control for files and directories. An ACL does not take the place of the traditional UNIX permission scheme where the owner of a file or directory controls access to the owner, group, and all other users of the system (i.e., world); instead it works in conjunction with the UNIX permissions. An ACL provides the ability to give one or more users permission to a file or directory.

- A *project* is a collection of files spread across multiple directories.
- A *model* consists of one or more projects.
- A *POC* is the point of contact for a project and owns all the directories (not necessarily all the files) under a project.
- A *development group* is a set of users who have the responsibility of changing a project's files. They are members of the UNIX group for the project's files.
- *Users* are the set of those who need access to certain files within the project. They are part of the world for UNIX permissions on files, but certain project directories will be given an ACL so users can navigate into directories where the files are located.

3. Requirements

In order to build a set of tools to aid in granting permissions in our complicated environment, a set of requirements were first created. These requirements included:

- Ability to change permissions for a project.
- Ability to change permissions for multiple projects (i.e., models).
- Ability to give different permissions to different individuals for a project.
- Ability to allow others besides the project's POC to

apply permissions to a project. These individuals are usually in the project's development group.

- Ability to allow files (not directories) within a project to be owned by different individuals. This allows the toolset to be used in conjunction with UNIX configuration management tools such as the Source Code Control System (SCCS).

4. Deficiencies with Cray Commands

Based on the above requirements, it was apparent that a set of tools must be written to compensate for the deficiencies in the traditional UNIX permission scheme and UNICOS MLS ACL commands.

With UNIX file permissions, the owner of a file or directory controls access to members of the group and to all other users on the system. The user's umask value determines the permission of the file or directory upon creation. No one other than the owner of the file/directory can grant permissions. Only three types of permissions can be given - owner, group, and world. If two individuals belong to the same group, those two individual cannot have different permissions to a particular file or directory. Also, two different groups cannot be given permission to the same file or directory.

The UNICOS MLS ACL permissions solve some of the problems with the UNIX file permissions, but they also introduce others. With an ACL, multiple groups can be given access to a file or directory, but only the owner can apply an ACL. An ACL works with the UNIX group permissions of a file or directory. In order to have the flexibility to grant an individual any permission to a file or directory, the UNIX group permission must be set to the highest permission needed. For example, if a file's UNIX group permissions were read and execute (rx), then no individual could be granted write (w) permission to that file with an ACL. By granting the highest UNIX group permission needed to a file or directory, the owner has now granted everyone in the UNIX group this permission. The owner must then use an ACL to restrict permissions to certain individuals in the UNIX group if necessary.

Finally, the UNICOS ACL commands (*spacl*, *spget*, *spset*) are difficult to use and force the user to apply the ACL to each file or directory. This could cause errors if the user fails to include a file or directory within the project.

5. Overview of Toolset

Figure 1 shows the basic flow of the tools within the Permission Support Toolset. Both **permit** and **permacl** are UNIX scripts written in the Bourne Shell programming language. The C executable program, **permit.exe**, can be defined to be a *setuid*¹ program if the owner decides he

would like other users to have the ability to place ACLs on the owner's project. The **permit** and **permacl** scripts are able to be shared by all users; however, the **permit.exe** program must be available under the project and owned by the POC.

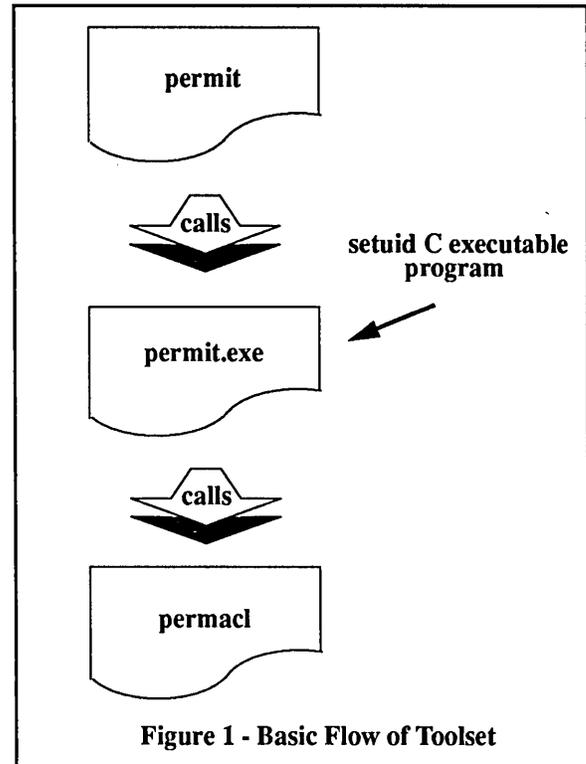


Figure 1 - Basic Flow of Toolset

6. permit Script

The **permit** script is the interface between the user and the ACL commands on the Cray. The following are the valid input parameters (with defaults in *bold italics*):

- project* path to project (required parameter)
- addrem* **add** | **rem** (option to grant/remove permissions)
- ckacl* **no** | **yes** (option to only check permissions on project)
- type* **users** | **project** | **permit** (different classes of permissions allowed)
- file* file containing names of user/group names
- name(s)* user/group names, with group names preceded by a colon (:). Note: either file parameter or at least one name must be present.

The input parameters are in *name=value* format. Except for the list of names (if present), the parameters can appear in any order.

Before executing the script with the desired input parameters, the user may define certain environment variables that the script will use. The environment variables used and their defaults are:

1. A *setuid* program is one that allows anyone who can execute the program to do so as the owner.

```

PERMITEXE  project/cpy_scpt/permit.exe
PERMISSION rx
LOCKFILE   /tmp/<each field of project>.lck
PROJFILE   project/cpy_scpt/projfile.txt

```

PERMITEXE signifies the location of the `permit.exe` program to execute. The **PERMISSION** applied via the ACL is read/execute by default. A lock file is used to prevent the possibility of multiple users changing the ACLs on the same project at the same time. **PROJFILE** defines a file which may contain a list of other projects to which the ACL should be applied. This saves the user from having to execute `permit` for each project; instead he can execute `permit` once for each model.

The `permit` script basically performs parameter error checks, then executes the file defined by the **PERMITEXE** variable. If `type=users` and **PROJFILE** exists, then `permit` will call **PERMITEXE** for each entry in **PROJFILE**.

7. permit.exe Program

The `permit.exe` file is a C executable program which will pass the parameters from `permit` to `permacl`. It can be made setuid by the owner if he wishes others to have the ability to place ACLs on the owner's files. This program will also pass the file name (**LISTS** macro) which contains the set of directories/files upon which to apply ACLs. The user cannot override the location of this file. Figure 2 shows the `permit.c` program, which is compiled to produce the `permit.exe` executable program.

```

/* @(#)permit.c 1.4 08:14:07 9/11/92 */
static char SCCS_Id[] = "@(#)permit.c 1.4 08:14:07
9/11/92";
#define LISTS "/home/k51/tools/permit/0/exec/lists.txt"
#define PERMACL "/home/k51/tools/exec/permacl"

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

main(int argc, char** argv) {
    int i,
        code,
        status;

    char **nargv = (char**) malloc(sizeof(char*) * (argc + 2));
    nargv[0] = (char*) malloc(sizeof(char) * (strlen(PERMACL)+1));
    strcpy(nargv[0],PERMACL);
    nargv[1] = (char*) malloc(sizeof(char) * (strlen(LISTS)+1));
    strcpy(nargv[1],LISTS);

    for (i = 1; i < argc; ++i) {
        nargv[i+1] = (char*) malloc(sizeof(char) * (strlen(argv[i]) + 1));
        strcpy(nargv[i+1],argv[i]);
        nargv[i+1] = NULL;
    }

    if (fork() == 0) {
        code = execv(PERMACL,nargv);
        fprintf(stderr,"ERROR: could not execute %s\n",PERMACL);
        wait(&status); }
}

```

Figure 2 - permit.c C Program

8. lists.txt File

The **LISTS** macro in the `permit.c` program is a file containing the set of directories/files upon which to apply ACLs. Figure 3 shows an example `lists.txt` file. It contains a maximum of three non-blank, non-commented lines representing the directories/files to apply users, project, and permit permissions. These lines correspond to the different classes of permissions allowed by the `type` parameter of the `permit` script. The **USERSLIST** would be used to grant permissions to the users of the project, those who just need to access the executable and perhaps header files. As shown in Figure 3, if `permit` is executed with `type=users`, then the given name(s)/group(s) would be granted permission to the include and exec directories. If `permit` is executed with `type=project`, then the name(s)/group(s) would be granted permission to the SCCS, src, include, obj, and exec directories. If `permit` is executed with `type=permit`, then permission is granted to the `cpy_scpt` directory and the `cpy_scpt/permit.exe` file. This is used in conjunction with a setuid `permit.exe` executable to allow those users the ability to place ACLs on the project. Any valid directory/file under the project can be specified in the `lists.txt` file. Wildcards can also be used (e.g., `exec/*` would signify all files within the `exec` directory under the project). However, it is much simpler to place ACLs only on the directories containing the files, instead of the files themselves. Then by making the files within that directory world readable, those given ACL permission to the directory will be able to access the files.

```

# @(#)lists.txt 1.2 9/11/92 08:01:03
# This file is located in ~k51/tools/permit/0/exec.
# It is used to define three classes of directories/files to be used with
# the K51 permit tool. Comments (preceded by #) can appear beginning in
# column 1 or at the end of the line containing the directories/files.
# They cannot appear on the same line, but preceding the data. Blank
# lines can also appear anywhere in the file.
# WARNING - THE ORDER OF THE DATA IS FIXED.
# If three or more non-commented, non-blank lines appear in file then
# USERSLIST is set to the first line
# PROJECTLIST is set to the second line
# PERMITLIST is set to the third line
# remaining lines (if any) are ignored
# If two non-commented, non-blank lines appear in file then
# USERSLIST is set to the first line
# PROJECTLIST and PERMITLIST are set to the second line
# If one non-commented, non-blank line appears in file then
# USERSLIST, PROJECTLIST, and PERMITLIST are set to that line
# USERSLIST
include exec
# PROJECTLIST
SCCS src include obj exec
# PERMITLIST
cpy_scpt cpy_scpt/permit.exe

```

Figure 3 - lists.txt File

9. permacl Script

The `permacl` script is called by the C executable program and is executed as the owner if the executable is setuid. All input parameters are passed from the `permit` script to `permacl`. After error checks are performed, an ACL modification file is created from the desired name(s)/group(s) input by the user. A lock file is created to lock other users from placing ACLs on the project's files until the current job is completed. A list of directories/files is defined based on the value of the `permit type` parameter. For each element of this list the following is performed:

- text version of ACL is retrieved.
- ACL file is created from the text version.
- name(s)/group(s) are removed from ACL file if *addrem* is set to *add*.
- ACL modification file is applied to the ACL file.
- ACL file is applied to the element.
- lock file is removed.

10. Toolset Setup - Owners of Directories/Files

In order to use the toolset, certain steps must first be taken to create setup files, to create necessary programs, and to set needed environment variables. These steps are different for owners of directories/files and for others who will apply permissions to these directories/files.

To use the toolset, the owner of directories/files must first:

- Create the setuid C executable program, `permit.c`, to pass parameters from the `permit` script to the `permacl` script.
- Ensure that the `LISTS` macro in `permit.c` is set appropriately.
- Compile `permit.c`; no special compilation parameters are required (`cc -o permit.exe permit.c`).
- Create the `LISTS` file that is specified in the above `LISTS` macro.
- Set the setuid permission bits on `permit.exe` to allow the UNIX group to execute the file as the owner. This is done with the command `chmod 4750 permit.exe`.

WARNING: Be careful with UNIX group and world permissions on setuid programs!

11. Toolset Use - Users

Once the owner has followed the above instructions, other users may apply permissions to the directories/files as well. To do this:

- Check/set the following environment variables: `PERMITEXE`, `PERMISSION`, `LOCKFILE`, `PROJFILE`.
- Execute the `permit` script with the appropriate parameters.

12. Toolset Example

The following example shows how the owner and users of a project can use the toolset. Figure 4 shows a project called `my_project`, which is divided into various directories, each containing files. The UNIX mode permissions on all directories/files are shown.

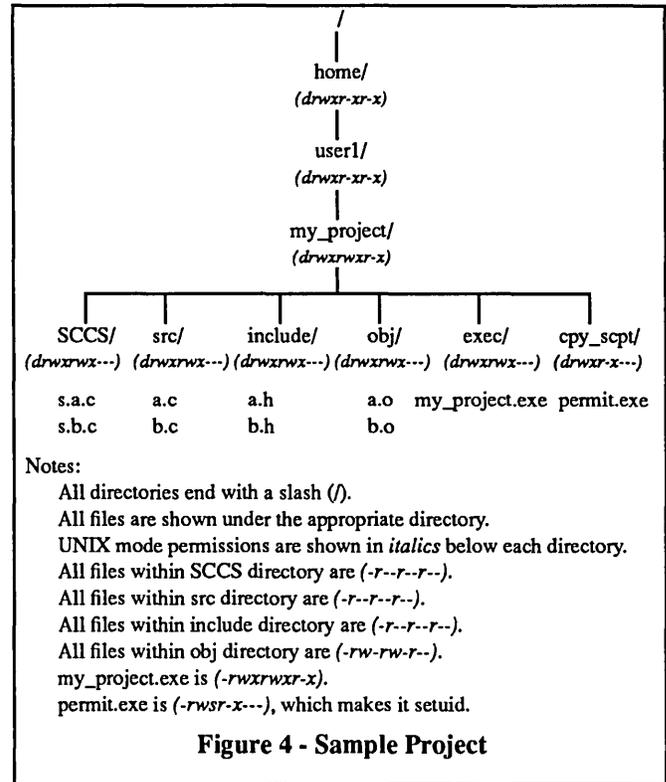


Figure 4 - Sample Project

The owner has set up this project with `cpy_scpt/permit.exe` as a setuid executable. If the `lists.txt` file of Figure 3 is used and the owner executes the command

```
permit project=/home/user1/my_project type=project joe mary
```

then the users `joe` and `mary` are given read/execute (rx) permission to all the directories except `cpy_scpt`. Since the files within those directories are at least read to the world, they can access all the files within those directories.

Next the owner executes the command

```
permit project=/home/user1/my_project type=permit bill
```

The user `bill` is given permission to execute the `cpy_scpt/permit.exe` executable. Since this is setuid, he can then grant ACLs on the project to other users.

Next, if the user `bill` or the owner executes

```
permit project=/home/user1/my_project don kate :math
```

then the users `don` and `kate` and the UNIX group `math` are given rx permission to the `include` and `exec` directories. If the `PERMISSION` environment variable was set to `rwX` before executing `permit`, they would have been given `rwX` permission to those directories.

13. Summary

The Cray File Permission Support Toolset was developed to enhance the standard ACL commands and meet the requirements of granting directory/file permissions in a complicated environment. The simplest use is to easily allow the owner of a project to set ACLs on the project's directories. Users who are allowed to navigate within the directory via the ACL can access all the files that have world read permission. A more complicated scenario would involve a setuid executable program allowing others beside the owner or members of the project's group to grant permissions to directories or files within the project. The toolset is general enough to be tailored to any developmental environment and any type of directory structure. It has allowed developers to easily permit different classes of permissions to users of multiple projects combined into models.

TOOLS FOR ACCESSING CRAY DATASETS ON NON-CRAY PLATFORMS

Peter W. Morreale

National Center for Atmospheric Research
Scientific Computing Division

ABSTRACT

NCAR has a long history of using Cray computers and as a result, some 25 terabytes of data on our Mass Storage System are in Cray-blocked format. With the addition of several non-Cray compute servers, software was written to give users the ability to read and write Cray-blocked and unblocked files on these platforms. These non-Cray platforms conform to the Institute for Electrical and Electronics Engineers (IEEE) standard describing floating-point data. Therefore, any tools for manipulating Cray datasets must also be able to convert between Cray data formats and IEEE data formats. While it is true that the Cray Flexible File I/O (FFIO) software can provide this capability on the Cray, moving this non-essential function from the Cray allows more Cray cycles for other compute-intensive jobs. This paper will outline a library of routines that allow users to manipulate Cray datasets on non-Cray platforms. The routines are available for both C and Fortran programs. In addition, three utilities that also manipulate Cray datasets will be discussed.

1. Introduction

The National Center for Atmospheric Research (NCAR) has been using Cray Supercomputers since a Cray-1A was installed in late 1976. NCAR now has a CRAY Y-MP 8/864, a CRAY Y-MP 2D/216, and a CRAY EL92/2-512 that are used for the bulk of computing by our user community. NCAR has an enormous amount of data in Cray format stored on NCAR's Mass Storage System. Currently, there are 40 terabytes of data on the NCAR Mass Storage System. Approximately 25 terabytes of that data in a Cray-blocked format.

Cray computers use their own format to represent data. On Cray computers, a 64-bit word is used to define both floating-point values and integer values. In addition, Cray computers support a number of different file structures. The most common Cray file format is the Cray-blocked, or COS-blocked, file structure. This file structure, known as a Cray dataset, is used by default when a file is created from a Fortran unformatted WRITE statement. The Cray-blocked dataset contains various 8-byte control words which define 512-word (4096 byte) blocks, end-of-record (EOR), end-of-file (EOF), and end-of-data (EOD). Cray also has an unblocked dataset structure that contains only data. No control words of any kind are present in an unblocked dataset.

In contrast to Cray systems, a number of vendors of other platforms use Institute for Electrical and Electronics Engineers (IEEE) binary floating-point standard for describing data in binary files. Generally, these vendors also use the same file structure for Fortran unformatted sequential access binary files. This file structure consists of a 4-byte control word, followed by data, terminated by another 4-byte control word for each record in the file. Because the same file structure and data representation are used by a number of

vendors, binary files created from Fortran programs are generally portable between these vendor's computers.

In recent years, a number of IEEE-based compute servers have been added to our site. In particular, NCAR now has a four-node cluster of IBM RISC System/6000 model 550 workstations, an eight-node IBM Scalable POWERparallel 1 (SP1), and a Thinking Machines, Inc. CM-5. Since many of the users on these platforms also use our Cray systems, the ability to use the same data files on all systems is extremely important.

One possible solution would be to use the Cray systems Flexible File I/O (FFIO) package. This software allows the user to create data files in binary formats suitable for direct use on different vendors platforms. The FFIO solution works for users creating new data files on the Cray; however, we have over 25 terabytes of Cray-blocked data already in existence on our Mass Storage System. If FFIO were the only solution, users would spend a good deal of their computing allocations just reformatting datasets. In addition, these Cray jobs would consume a large number of Cray cycles that would otherwise be used for compute-intensive work.

Another solution would be to use formatted data files. This solution poses several problems: 1) formatted files are generally larger than their binary counterparts, 2) formatted I/O is the slowest form of I/O on any computer since the text must be interpreted and converted into binary format, and 3) formatted files can incur a loss of precision.

A third solution would be to provide software that can interpret Cray file structures and convert the Cray data representation into the non-Cray data format. This solution has several benefits for the user. One advantage is that the user can use the same datasets on both the Cray and non-Cray machines. Another benefit is that even accounting for the data conversion, the I/O on the non-Cray platform is significantly faster than equivalent formatted I/O on the non-Cray platform.

At NCAR, we have implemented the third solution in the form of a library of routines that perform I/O to and from Cray datasets. This paper describes the library named NCAR Utilities (ncaru). The paper also describes three stand alone utilities that manipulate Cray datasets.

2. The ncaru software package

The ncaru library contains a complete set of routines for performing I/O on Cray datasets. In addition, a number of routines that convert data between Cray format and IEEE format are also included in the library.

The user can use the Cray I/O routines to transfer data in Cray format or have the routine automatically convert the data to the native format. Having the option of converting data allows the user to read datasets that contain both numeric and character data records.

The ncaru library is written in the C language. Since there is no standard for inter-language communication, the user entry points to the library must be ported to the different platforms for use in Fortran programs. The current implementation has been ported to IBM RISC System/6000 systems running AIX 3.2.2, Silicon Graphics Inc. Challenge-L running IRIX V5.1.1.2 and to Sun Microsystems, Inc. systems running SunOS 4.1.1.

The ncaru software package also includes three utilities that aid the user in manipulating Cray-blocked files on non-Cray platforms: cosfile, cosconvert, and cossplit. These utilities describe records and file structure of a Cray dataset (cosfile), strip Cray-blocking from a Cray dataset (cosconvert), and split multi-file datasets into separate datasets (cossplit).

Documentation for the ncaru package consists of UNIX man pages for each routine and utility. There is also a ncaru man page that describes the library and lists all the user entry point routine names.

3. The Cray I/O routines

The Cray I/O routines in the ncaru library allow the user to read, create, or append to a Cray dataset. The user also specifies whether the dataset uses a Cray-blocked or Cray-unblocked file structure.

The Cray I/O routines use a library buffer to block I/O transfers to and from the disk file. This buffer imitates the library buffer used in Cray system I/O libraries. Use of a library buffer can reduce the amount of system work necessary to perform I/O, with the trade-off being increased memory usage for the program.

The following is a list of the Cray I/O routines with their arguments.

```
ier = crayblocks(n)
icf = crayopen(path, iflag, mode)
nwds = crayread(icf, loc, nwords, iconv)
nwds = craywrite(icf, loc, nwords, iconv)
ier = crayback(icf)
ier = crayrew(icf)
ier = crayweof(icf)
ier = crayweod(icf)
ier = crayclose(icf)
```

The crayblocks routine allows the user to specify a library buffer size. The argument *n* specifies the number of 4096-byte blocks used by the buffer. This library buffer is dynamically allocated and is released when the file is closed with a crayclose routine. If the crayblocks routine is used, all Cray datasets opened with a crayopen use the specified block size until another crayblocks routine is executed. The crayblocks routine must be executed prior to a crayopen routine if something other than the default library buffer size (1 block) is desired.

The crayopen routine opens a dataset for either reading or writing. The *path* argument specifies the pathname to the file. The *iflag* argument specifies the transfer mode, whether the file structure is blocked or unblocked, and the position of the file. The *mode* argument specifies the file permissions and is used only if the file is being created. The crayopen routine dynamically allocates a data structure that contains fields used by the various I/O routines. The return from a successful crayopen is the address of this data structure. By returning the address of the structure as an integer, portability between Fortran and C is assured.

The crayread routine reads data from an existing Cray dataset. The *icf* argument is the return from a previously executed crayopen routine. The *loc* argument is the location where the first word of data is placed and must conform both in type and wordsize to the data being read. The *nwords* argument specifies the number of words being read. The *iconv* argument specifies the desired data conversion.

For blocked files, crayread is fully record-oriented. This means that if the user specifies a read of a single word, the first word of the record is transferred and the file is left positioned at the next record. This feature is useful for skipping records. The user can also specify a read of more words than the record actually contains, and only the actual number of words in the record are transferred. This feature is useful if the user is not sure of the exact number of words in the record. In all cases, crayread returns the number of Cray words actually transferred or an error code.

The craywrite routine writes data to a Cray dataset. Like crayread, the arguments *icf*, *loc*, *nwords*, and *iconv*, correspond to the crayopen return value, location of the data being written, the number of words to write, and the conversion flag. Both the crayread and craywrite routines use a library buffer to reduce the number of physical read or write requests to disk. For writing, when the buffer is filled, the library buffer is flushed to disk. This means that if the user does not close the file via a crayclose, the resulting Cray dataset may be unusable on the Cray computer.

If the *iconv* flag for both crayread and craywrite specifies a numeric conversion, then a conversion buffer is dynamically allocated. The initial size of the conversion buffer is set to the number of words in the request. The size of the conversion buffer is then checked for each subsequent I/O request, and if a subsequent request is larger than the current size of the conversion buffer, the buffer is re-allocated to the larger size. On every request, every byte of the conversion buffer is preset to zero to prevent bit conversion problems.

The `crayback` routine allows the user to backup a record in a Cray dataset. The `crayback` routine can be used on datasets opened for reading or writing. If the last operation to the dataset was a write, then `crayback` will truncate the dataset prior to positioning at the previous record. This allows the user to overwrite a record if desired and mimics the behavior of a Cray Fortran `BACKSPACE`.

The `crayweof` routine writes a Cray end-of-file control word. The `crayweod` routine writes a Cray end-of-data control word. These two routines are available for historical purposes and are seldom used directly by the user. One possible use of the `crayweof` routine would be the creation of a multi-file dataset.

The `crayclose` routine properly terminates and, if necessary, flushes the library buffer. The `crayclose` routine then closes the dataset and releases all dynamically allocated memory used for that file.

4. The numeric conversion routines

In addition to the Cray I/O routines, a number of routines to convert Cray data formats to IEEE data formats are included in the `ncaru` library. These routines are implemented as Fortran subroutine calls and in C as void functions. Here is a list of the routines and their arguments:

```
ctodpf(carray, larray, n)
ctospf(carray, larray, n)
ctospi(carray, larray, n, zpad)
dptocf(larray, carray, n)
sptocf(larray, carray, n)
sptoci(larray, carray, n, zpad)
```

In all the routines, the first argument is the location of the input values and the second argument is the location for the converted values. The third argument to all the routines is the number of words to convert. If the routine has a fourth argument, it is used to tell the conversion routine whether the IBM RISC System/6000 double-padded integer option was used during compilation.

In all cases, the `carray` argument is a pointer to an array containing 64-bit Cray values and the `larray` argument is a pointer to an array containing the IEEE 32-bit or IEEE 64-bit values.

The `ctodpf`, `ctospf`, and `ctospi` routines convert Cray data to local DOUBLE PRECISION, REAL, and INTEGER values. The `dptocf`, `sptocf`, and `sptoci` routines convert local data to Cray REAL and INTEGER values.

For the routines that convert to IEEE format, any values that are too large to be properly represented are set to the largest value that can be represented with the correct sign. Any values that are too small are set to 0 (zero).

Both the Sun Microsystems and IBM RISC System/6000 Fortran compilers allow the user to specify a command line option that automatically promotes variables declared as REAL to DOUBLE PRECISION. This causes the word size to double from the default 4 bytes to 8 bytes. The routines with "dpf" in their names should be used in these cases. In addition, the IBM xlf compiler has an option that allows the

compiler to increase the size of Fortran INTEGERS to 8 bytes, 4 bytes to hold the data and 4 bytes of alignment space. For the integer conversion routines, the `zpad` argument is used to inform the routine whether the compiler option was used.

The numeric conversion routines can either be executed directly from the user program or automatically called through the use of the Cray I/O routines via the `iconv` argument to `crayread` and `craywrite`.

5. Example Fortran program

The following sample Fortran program creates a Cray-blocked dataset with a single record. The IEEE 32-bit REAL values are converted to Cray single-precision REAL values prior to being written.

```
PROGRAM TST
REAL    a(1024)
INTEGER CRAYOPEN, CRAYWRITE, CRAYCLOSE
INTEGER ICF, NWDS, IER

ICF = CRAYOPEN("data", 1, 0'660')
IF (ICF .LE. 0) THEN
    PRINT*, "Unable to open dataset"
    STOP
ENDIF

NWDS = CRAYWRITE(ICF, A, 1024, 1)
IF (NWDS .LE. 0) THEN
    PRINT*, "Write failed"
    STOP
ENDIF

IER = CRAYCLOSE(ICF)
IF (ICF .NE. 0) THEN
    PRINT*, "Unable to close dataset"
    STOP
ENDIF

PRINT*, "Success!"
END
```

6. Cray dataset utilities

To assist users with manipulating Cray datasets, three utilities that operate on Cray-blocked files were created. These utilities are `cosfile`, `cosconvert`, and `cosplit`.

The `cosfile` utility verifies that the specified file is in a Cray-blocked format and gives information about the contents of the dataset. The number of records and their sizes are displayed for each file in the dataset. In addition, `cosfile` attempts to determine whether the file contains ASCII or binary data and reports the percentages of each. Here is a sample `cosfile` command and its resulting output:

```
% cosfile -v /tmp/data

Processing dataset: /tmp/data
  Rec#   Bytes
    1     800
    2    8000
EOF 1: Recs=2 Min=800 Max=8000 Avg=4400 Bytes=8800
      Type=Binary or mixed -- Binary= 99% ASCII= 1%
EOD. Min=800 Max=8000 Bytes=8800
```

The `cosconvert` utility converts a Cray-blocked data set into one

of several formats. Most often, cosconvert is used to strip Cray control words from a dataset, leaving only data. In some cases, Cray datasets may contain character data with Blank Field Initiation (BFI). BFI was used under COS to compress datasets by replacing three or more blanks in a row with a special two character code. The cosconvert utility can be used to expand the blanks in those datasets.

The cossplit utility creates single file datasets from a multi-file dataset. Each output single file dataset will have a unique name.

7. Acknowledgments

The ncaru software package is the result of the work of several people at NCAR. Charles D'Ambra of the Climate and Global Dynamics (CGD) division of NCAR wrote the original Cray-IEEE numeric conversion routines. Craig Ruff of the Scientific Computing Division (SCD) wrote the original Cray I/O routines for the purpose of adding and stripping Cray-blocking from files. Dan Anderson and Greg Woods of SCD combined both the numeric conversion routines and the Cray routines into a single interface. Tom Parker (SCD) originally wrote cosfile, cosconvert, and cossplit for use on Cray systems. The author, also of SCD, modified the library code to handle Cray-unblocked files, added backspacing functionality, rewrote the utilities to use the library, and added other enhancements.

8. Availability

This package is available to interested organizations without charge. Please contact Peter Morreale by sending email to morreale@ncar.ucar.edu for details.

CENTRALIZED USER BANKING AND USER ADMINISTRATION ON UNICOS

Morris A. Jette, Jr. and John Reynolds

National Energy Research Supercomputer Center
Livermore, California

Abstract

A Centralized User Banking (CUB) and user administration capability has been developed at the National Energy Research Supercomputer Center (NERSC) for UNICOS and other UNIX platforms. CUB performs resource allocation, resource accounting and user administration with a workstation as the server and four Cray supercomputers as the current clients. Resources allocated at the computer center may be consumed on any available computer. Accounts are administered through the CUB server and modifications are automatically propagated to the appropriate computers. These tools facilitate the management of a computer center as a single resource rather than a collection of independent resources.

The National Energy Research Supercomputer Center

NERSC is funded by the United States Department of Energy (DOE). The user community consists of about 4,800 researchers worldwide and about 2,000 high school and college students. The user community is divided into about 600 organizations or accounts. The available compute resources include a Cray Y-MP C90/16-256, Cray-2/8-128, Cray-2/4-128, Cray Y-MP EL and an assortment of workstations. Other resources include an international network, a Common File System (CFS) with 12 terabytes of data and the National Storage Laboratory (NSL, a high-performance archive being developed as a collaborative venture with industrial partners and other research institutions).

General Allocation and Accounting Requirements

While our student users are generally confined to use of the Cray Y-MP EL, the researchers are free to use any of the other computers. The DOE allocates available NERSC compute resources for the center as a whole, not by individual computer. Researchers are expected to consume their computer resources on whichever computers are most cost effective for their problems.

Resources are allocated in Cray Recharge Units (CRUs), which are for historical reasons based upon the compute power of a Cray-1. A "typical" problem may be executed on any available resource and be charged a similar number of CRUs. The CPU time consumed is multiplied by a CPU

speed factor to normalize these charges. These factors have been derived from benchmarks. Our CPU speed factors are as follows:

<u>CPU SPEED</u> <u>FACTOR</u>	<u>COMPUTER</u>
3.50	Cray Y-MP C90/16-256
1.63	Cray-2/4-128
1.44	Cray-2/8-128
1.00	Cray-1 A (no longer in service)

The charge rates vary with the level of service desired. Our machines have been tuned to provide a wide range of service levels depending upon the nice value of a process and whether the work is performed interactively or under batch. A charge priority factor permits users to prioritize their work based upon the level of service desired and be charged accordingly. This scheme has proven very popular with our clients. The lowest priority batch jobs have a charge priority of 0.1. The highest priority interactive jobs have a charge priority of 3.0. The actual formulas for charge priority factors are as follows:

<u>JOB</u> <u>TYPE</u>	<u>NICE</u> <u>VALUE</u>	<u>CHARGE</u> <u>PRIORITY</u> <u>ALGORITHM</u>	<u>CHARGE</u> <u>PRIORITY</u>
Interactive	0 to 10	$0.2*(10-nice)+1.0$	3.0 to 1.0
NQS	5 to 19	$0.1*(10-nice)+1.0$	1.5 to 0.1

Charges are made on a per process basis. It must be possible to alter the nice value of a process up or down at any time. NERSC has developed a simple program called CONTROL which alters nice values of processes and sessions up and down. CONTROL runs as a root process and restricts nice values to the ranges specified in the above table. The NICEM program has been replaced by a front-end to the CONTROL command. The system call to increase nice values has not been altered. The charge rate must change when the nice value is altered. It is a requirement to be able to allow changes in service level or charge rate once a process has begun execution. The ability to react to changing workloads is considered essential.

CPU use must be monitored in near real time. This can be accomplished on many UNIX systems by reading the process and/or session tables and noting changes in CPU time used.

Waiting for timecards to be generated on process completion, as in standard UNIX accounting, is not considered acceptable. With some processes running for hundreds of CPU hours, monitoring timecards would occasionally result in substantially more resources being consumed than authorized. Additionally, many users have discovered that they can avoid charges at some computer centers by preventing their jobs from terminating. Their jobs can complete useful work and sleep until the computer crashes, avoiding the generation of a timecard.

The disadvantage of near real time accounting is that jobs are charged for resources consumed, even if they fail to complete due to system failure. The Cray hardware and software is reliable enough to make this only a minor issue. Refunds are not given in the cases of job failure due to user error. Refunds are not given in cases where less than 30 CRU minutes are lost, since the user benefit is probably less than the effort involved in determining the magnitude of the loss. The maximum refund is generally 600 CRU minutes. Users are expected to provide adequate protection from greater losses through checkpoints and backups. Since the UNICOS checkpoint/restart mechanism is fairly fragile, many users have developed their own checkpoint/restart mechanism. In most cases, user checkpoint/restart mechanism are quite robust and have modest storage requirements.

At present, we are charging only for CPU use. The net charge is calculated by the following formula:

$$\text{CRU charge} = \text{CPU time} * \text{CPU speed factor} * \text{charge priority}$$

The DOE allocates resources annually by account. In order to maintain a relatively uniform workload throughout the year and insure that an account does not prematurely consume its entire allocation, the allocated resources need to be made available in periodic installments. Each account has one or more managers who are responsible for the account administration. A wide range of administrative controls are essential, including: the rate at which the annual allocation is made available (weekly, monthly or other periodic updates plus allocation infusions on demand), designating other managers, and allocating resources to the individual users in the account.

Each user has access to some percentage of the account(s) to which he has access. In order to simplify administration, the sum of percentages allocated to each account's user need not equal 100. For example, the manager may want to make the account available in its entirety to each user. In this case, each user would have access to 100 percent of the account's allocation.

Once a user or his account has exhausted his allocation, he should be prevented from consuming significant additional resources. Rather than preventing the user from doing any work when his allocation has been exhausted, running interactive jobs must be suspended, running and queued NQS jobs must be held and the login shell must be changed to a Very Restricted Shell (VRSH). VRSH permits execution of a limited set of programs such as: MAIL, NEWS, RM, LS, KILL, CAT, LPR, MV, TAR, QSTAT, QDEL, CUB user

interfaces, archive user interfaces and a few others. These permit the user to finish working in some sort of graceful fashion whenever necessary. Once a user or his account have additional resources available, held or suspended jobs must be restarted automatically and the login shell restored to its former value. Interactive jobs are kept suspended only for 24 hours, then killed in order to release their resources. NERSC clients execute substantial interactive jobs. The temporary suspension of interactive jobs minimizes the impact of resource controls. In practice, a user with an important interactive job suspended would contact an account manager, be allocated additional resources and continue working in short order. E-mail and terminal messages must be sent to notify users of each action.

It must be possible to monitor resource availability and usage in real time. It must be possible to alter the percentage of an account allocated to a user in real time. It is highly desirable that the manager of an account have the ability to control the interval at which periodic installments of allocation are made. It is highly desirable that CUB user interfaces be available in both X window and command line forms.

Monthly accounting reports must be produced and mailed to each manager. A variety reports are required, including the following information: raw CPU time consumed, average charge priority and CRUs consumed by user and computer. The precision of all accounting information must be maintained to within seconds.

General User Administration Requirements

The NERSC user community is quite large and has a substantial turnover rate, particularly for the students. Managing 6,800 user accounts on several machines with different environments is very time consuming. NERSC formerly had a centralized user administration system for a proprietary operating system (the Cray Time Sharing System, CTSS, developed at Lawrence Livermore National Laboratory). When NERSC installed UNICOS, our support staff immediately found its workload increased by a factor of about three. Rather than increase our support staff accordingly, we decided that the implementation of a centralized user management system was essential.

The desired functionality was basic UNIX user administration: adding and deleting users and groups, changing user groups, etc. NERSC was also required site specific administration: assigning email aliases and propagating them to mail hubs; updating local banking databases on each CRAY; putting postal mail addresses into an address database. Ideally, all of the required user administration would be minimally specified once, within one user interface, and the required actions would be performed automatically on the appropriate computers. Local daemons would eliminate the need for repetitive actions and insulate the support staff from peculiarities of user administration on any computer.

Basic Design Criteria

Previous experience developing and using a similar system

led us to believe that having a single centralized database would be desirable. This makes for simple administration of the entire computer center. Clients on various computers would insure that the necessary allocation and user management functions are performed, maintaining a consistent state across all computers at the center in an highly reliable fashion. Centralized accounting, consistent with the allocation system information, would follow quite naturally.

While we were willing to take advantage of special features available with UNICOS, the operating system providing most of our compute capacity, the system would have to support UNIX platforms of any variety. Specialized software requirements (e.g., database licenses) could be required on the CUB server, but standard UNIX software should be the only requirement for CUB clients. The system would have to be user friendly. Use of commercial software, wherever possible, was encouraged.

Centralized User Banking Development

While no available system at the start of this project satisfied our needs, there were two resource allocation system which provided partial solutions. The San Diego Supercomputer Center (SDSC) had developed a CPU quota enforcement system in 1990. NERSC adopted this system in 1991 and found it generally satisfied our needs, except for managing allocations on each machine independently. We enhanced this system to permit the transfer of resources between machines, but it still was not entirely satisfactory. The Livermore Computer Center used SDSC's allocation system as a model for the allocation sub-system of their Production Control System (PCS). We found the PCS software gave us somewhat greater flexibility and decided to use that as the basis of CUB's banking client.

We knew that the design of CUB was certain to be quite complex, involving several programmers from different NERSC groups. It was decided that a CASE tool would provide for easier software development and more reliability. After examining several options, we selected Software Through Pictures from Interactive Development Environments. While the training cost was substantial in terms of initial programmer productivity, there was general agreement that the use of CASE was a real advantage. A small portion of the CUB database is shown in Figure 1 to indicate its complexity.

The banking client is a daemon which reads the process and session tables at intervals of ten seconds. The user ID, process or session ID, system and user CPU time used, and a few other fields are recorded. Changes in CPU time used are noted each time the tables are read. Periodically, CPU time used and CRU charge information is transferred for all active users and accounts to the banking server. The banking server updates its database then transfers updated records of resources available by user and account.

Communications are UDP/IP based, for performance reasons. Additional logic has been added to provide for sequencing and message integrity. A very simply RPC has been developed along with an API which insulates the application

level programmers from the complexity of the network. Each connection is fully authenticated and a unique session key is established. Transactions are protected by a crypto-checksum utilizing the session key to insure security. The result is efficient, easily portable, and fairly secure.

The record of previous process and session table information permits one to note the starting of new processes and sessions. The system and user CPU times are reestablished shortly after a held NQS job is restarted. We avoid charging for these restarted jobs by logging, though not charging, in cases where the record of CPU time consumed between checks exceeds a "reasonable" value. We consider anything over the real time elapsed multiplied by the number of CPUs multiplied by 2.0 to be an "unreasonable" value. The only CPU time not accounted for is that consumed prior to the first snapshot on a restarted NQS job and that consumed after the last snapshot of any session. Since CUB's sampling interval is ten seconds, the CPU time not accounted for is typically under 0.2 percent. The precision of the data can be increased in proportion to the resources consumed by the CUB local banker. The resources consumed by the local banker itself are 6.92 CPU minutes per day on a Cray Y-MP C90.

A client daemon called ACCTMAN performs updates to local databases as required for user administration. It performs system administration actions such as adding, deleting or modifying users or groups. This includes creating home directories, copying start-up files, creating mail directories, etc.

NERSC has used ORACLE for most of its database work over the past few years. The logical choice for the CUB server database was thus ORACLE. We had an initial design goal of making CUB completely database independent, but quickly found that to be impractical given the urgency of our need for the capabilities represented by CUB and the knowledge base of the available programmers. Using ORACLE's 4GL FORMS proved to be the quickest way to develop an interface for our support staff, and thus the quickest way to get a base-line version into production. The result is that CUB's BANKER, and ACCTMAN daemons, and the SETCUB utility suite can run on any UNIX host; the server code is completely written in standard SQL and could employ any UNIX SQL engine; but the support staff interface is wedded to ORACLE (but not to UNIX) for the foreseeable future.

Accounting reports are generated monthly based upon the CUB database and mailed to the primary manager of each account.

In order to insure reliability, three machines are available as CUB servers: a production machine, a "warm backup" and a development platform. The "warm backup" has a database that is only 15 minutes old and can be made into the production machine within about 30 minutes, if needed. The clients can continue operation for an extended period without a server. The lack of an operating server merely prevents the clients from synchronizing their databases and prevents other database updates. Once communications are reestablished, the databases are synchronized. In practice, the servers have

been quite reliable, in spite of the continuing development effort to complete and enhance CUB.

Database alterations (other than consumption of resources) are recorded in a journal in ORACLE to insure persistence across restarts. The database is also backed up regularly, including the journals. If a client goes down for an extended period of time transactions destined for it are not lost.

No alterations to the UNICOS kernel were required for CUB. The LOGIN.C program was modified to request an account name or number if the user belongs to more than one account. A default account is used if the user enters RETURN.

The primary user interface to CUB is called SETCUB. This program can be used by account managers to modify the CUB database. Other users can execute SETCUB to monitor resource allocations. In order to make CUB easier to use, the SETCUB program can also be executed by the name VIEWCUB or USERINFO. Both VIEWCUB and USERINFO provide a subset of SETCUB's functionality. VIEWCUB can only be used to report allocation information. USERINFO only reports user information, such as telephone number and postal address. Examples of SETCUB reports as shown in Figures 2, 3 and 4.

The overall CUB architecture is shown in Figure 5 showing the interrelationships between the major components.

Future Plans

While a GUI CUB interface is available to NERSC support staff, only a command line interface is available to most users. An X window interface is currently under development.

Most NERSC users have login names of the form "u" followed by a number. We plan to permit users the ability to modify their login name to something they find more suitable, on a one-time-basis. We will maintain a flat namespace that includes login names, email aliases, and certain reserved words.

NERSC found that charging for archival storage was ineffective in controlling growth. We instituted a quota by account in 1993, which has substantially reduced the archive growth rate and resulted in the elimination of a substantial amount of old data. At present, this quota is administered only by account (not user) and is not yet integrated with CUB. This will be rectified in the second quarter of 1994.

Problems of controlling storage use exist not only in the archive, but also on-line disks. We are planning to impose charges for disk space. This will be accomplished by a daemon periodically calculating on-line storage by user and account then relaying that information to the CUB server.

At present, a user can only be associated with a single account. This will soon be changed to permit use of multiple accounts. The UNICOS NEWACCT command can alter the account ID on process table entries. Other platforms would rely upon a user program relaying to the local banking client

which account should be associated with each session or process.

Once a single user can be associated with multiple accounts, we will enable an "add/delete" capability through SETCUB that will grant PI's and their managers the ability to add/delete existing NERSC users to their accounts, without NERSC support staff intervention.

We plan to implement Kerberos V5 as a centralized authentication service for NERSC users on all NERSC platforms. Having Kerberos will provide much greater network security for those users able to take advantage of it, because with Kerberos, passwords are never transmitted in the clear. Kerberos V5 also allows for a higher level of convenience. Once the Kerberos security token is obtained, and assuming the proper client software is installed on the users workstation, the user can securely telnet or ftp to NERSC machines without providing a password. There are some problems that must be solved regarding use of Kerberos at NERSC. What to do about the many users who will not have Kerberos on their local hosts is the biggest one. We are working on this now

CUB records resource usage only by user and account. In order to determine which processes consumed the resources, it is necessary to rely upon UNIX timecards. At some point in the future, we would like to be able to keep track of resource use by program. Given the volume of data required to record each process executed, this would likely only be used on long running processes.

The client daemons were originally designed for use with UNICOS. They currently being ported to Hewlett Packard workstations. Ports are planned for Cray T3D and Intel Paragon.

Accounts are presently independent of each other. We regard hierarchical accounts as something desirable. In PCS, account managers can create and delete sub-accounts, move users between sub-accounts, and move allocated resources between sub-accounts. At some time in the future, we may incorporate hierarchical accounts patterned after the work of PCS.

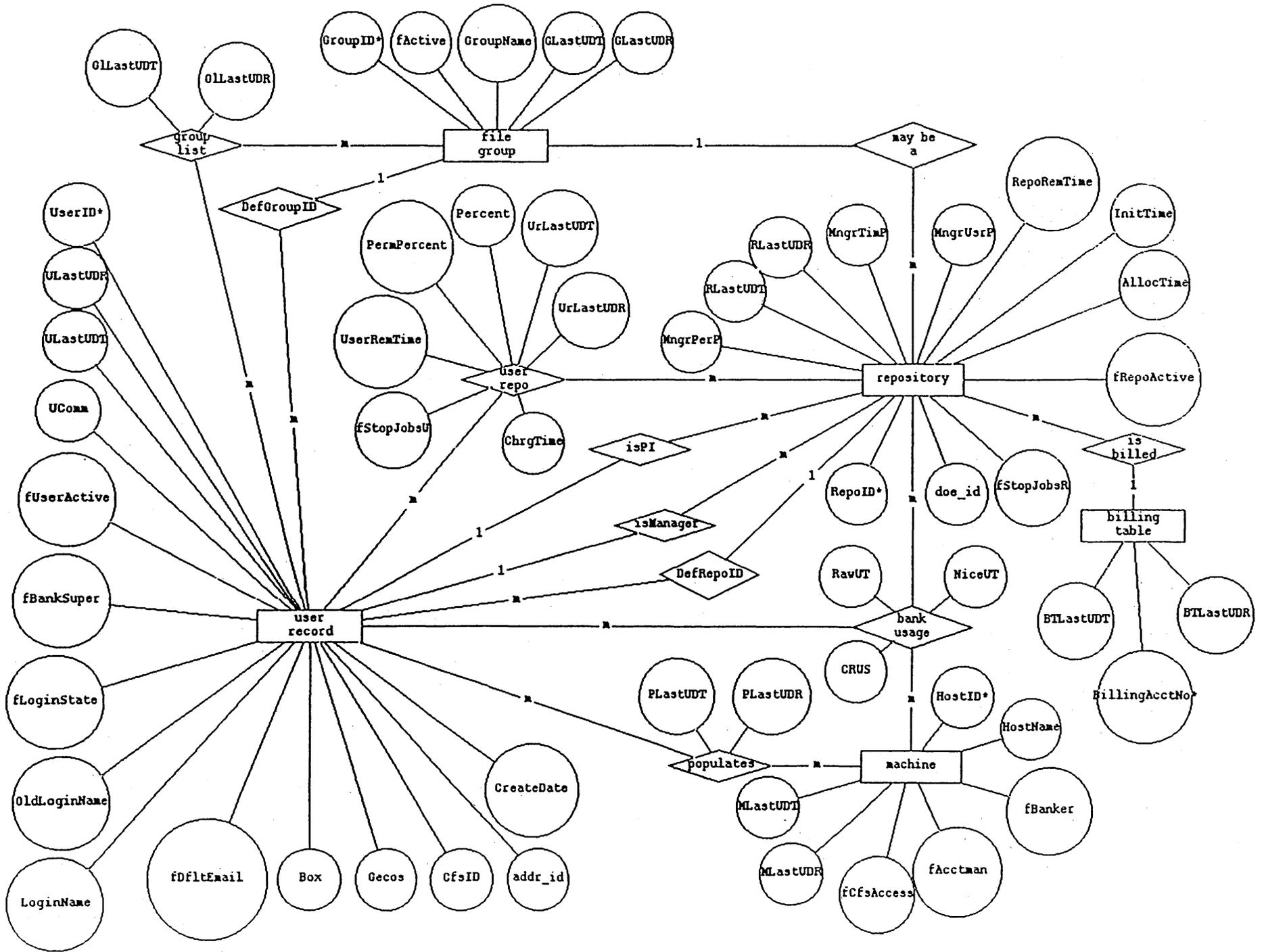
Acknowledgments

CUB's development has been the result of substantial efforts on the part of many programmers. The following programmers developed CUB: Patty Clemo, Steve Green, Harry Massaro, Art Scott, Sue Smith and the authors.

This work was funded by the U.S. Department of Energy under contract W-7405-Eng-48.

References

- 1 Hutton, Thomas E., "Implementation of the UNICOS CPU Quota Enforcement System," Proceedings, Twenty-Fifth Semi-Annual Cray User Group Meeting, Toronto, Canada, April 1990.
2. Wood, Robert, "Livermore Computing's Production Control System Product Description," Lawrence Livermore National Laboratory Report, November 1993.



Object: Generate allocation information about the users in account p6.

Input: setcub view repo=p6 members

Output:

Users In Repository p6 (id 1032)

<u>Login Name</u>	<u>Name</u>	<u>Initial Alloc</u>	<u>User Time Remaining</u>	<u>Charge Time</u>	<u>% Used</u>	<u>% of Alloc.</u>	<u>Perm%</u>
u445	M. CHANCE	1674:06	1667:35	6:31	0.18	46.00	46.00
u447	J. MANICKAM	3093:28	2512:09	581:19	15.97	85.00	85.00
u4100	J. MANICKAM	727:52	727:52	0:00	0.00	20.00	20.00
u4150	R. DEWAR	727:52	727:52	0:00	0:00	20:00	20:00
u4477	J. MANICKAM	1455:45	1455:45	0:00	0:00	40:00	40:00
Current amounts for p6		3639:23	3051:33	587:50	16:15		

Object: Generate detailed allocation information about the account p6.

Input: setcub view repo=p6 long

Output:

<u>Repository</u>	<u>Total Year Allocation</u>	<u>Remaining Year Alloc.</u>	<u>Initial Period Alloc.</u>	<u>Remaining Period Alloc.</u>	<u>Update Period</u>
p6 1032	42540:00	34007:00	4525:10	305:56	monthly
	<u>Period Increment</u>	<u>Period Carry Over</u>	<u>Next Update</u>	<u>PI/ Login Name</u>	
	3420:00	412:10	01-DEC-93	STEVE JARDIN u431	

Managers (privileges)

u44444 - LISA HANCOCK (t,u,c)

Object: Generate information about a specific user

Input: setcub locate user=u7145

Output:

NERSC DATABASE INFORMATION

Common Name

Loginname

MOE JETTE

u7145

Title:

LCode: 561 Bldg. 451 Rm: 2062

BOX: C86

CFSid: 7145

NERSC Domain Email Aliases: @nersc.gov

u7145

jette <Preferred 822 Alias>

U.S. Postal Information:

NERSC

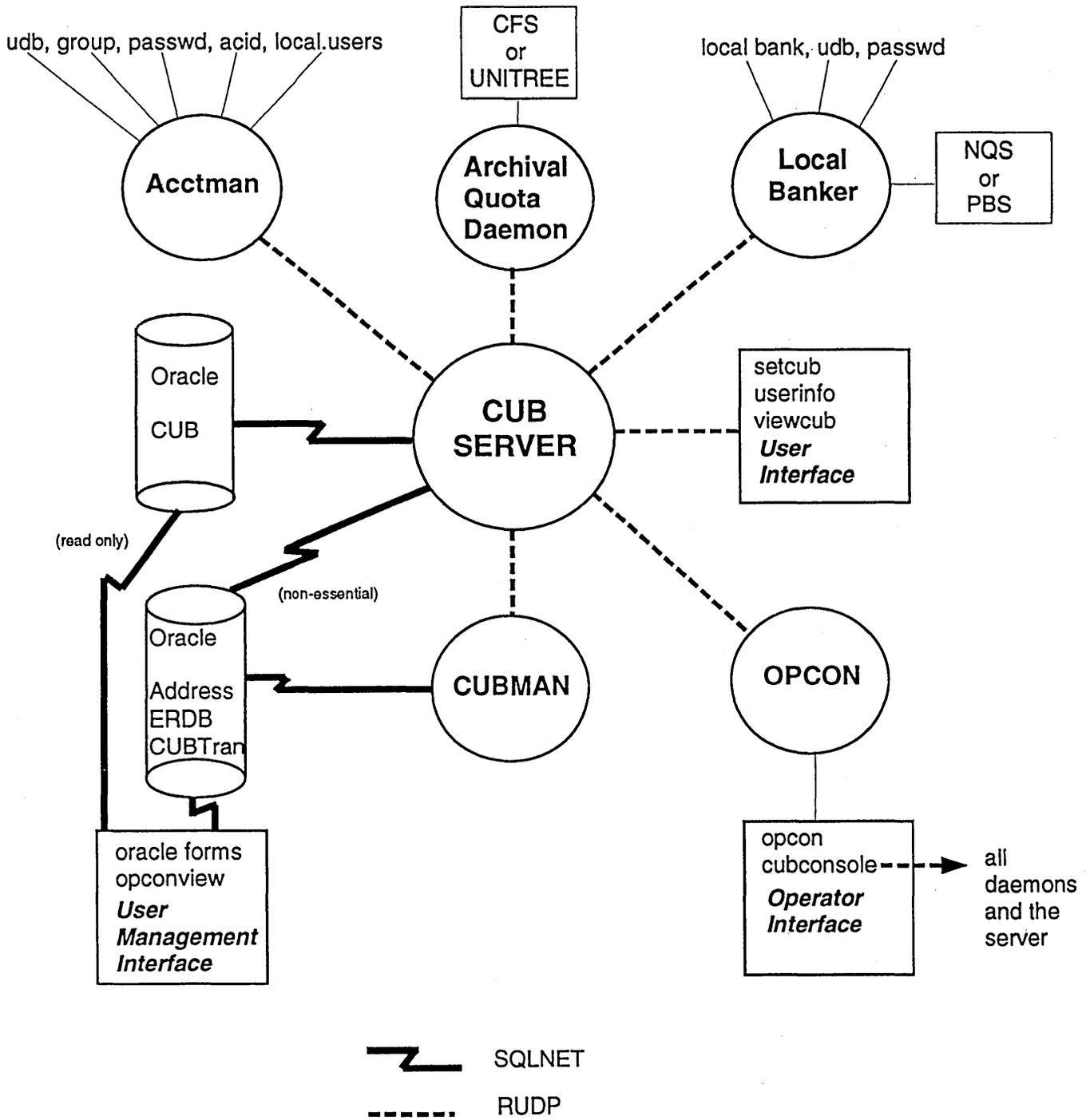
P.O. Box 5509, L-561

Livermore, CA 94550 USA

Work Phone: +1 510-423-4856

CUB ARCHITECTURE

CUB Platform: CRAY (soon HP, MPP)



FORTRAN 90 UPDATE

Jon Steidel
Compiler Group

Cray Research, Inc.
Eagan, Minnesota

1.0 Introduction

The first release of the Cray Fortran 90 Programming Environment (CF90 1.0) occurred in December of 1993. CF90 will replace Cray's current production Fortran product (CF77) in future releases. This paper discusses the current status of the CF90 product, its performance compared to CF77, the CF77 to CF90 transition plan, and current feature plans for the second release of the CF90 programming environment.

2.0 CF90 1.0 Status

The CF90 1.0 status is divided into three subsections. These are the release status of CF90 on various platforms, the performance of CF90 1.0 on parallel vector platforms compared to CF77, and the status of Fortran 90 interpretation processing by the American National Standards Institute (ANSI) and the International Standards Organization (ISO). Potential impact of Fortran 90 interpretations on CF90 users is discussed in the third subsection.

2.1 CF90 Release Status

The CF90 Programming Environment 1.0 was released December 23, 1993 on Cray Y-MP, C90, and YMP-EL systems. To date, there have been forty-five CF90 licenses purchased. These licenses represent twenty-five Cray Y-MP and C90 systems installed at twenty-one customer sites, plus twenty licenses on YMP-EL systems. As of early March, three upgrade releases have been made to the product to address bugfixes, and the first revision level release is planned for early second quarter 1994.

The CF90 programming environment will be released as a native programming environment on SPARC Solaris systems in third quarter 1994. Components of the CF90 programming environment also serve as the basis of the Distributed Programming Environment (DPE) which will be also be released in the third quarter of 1994. Please see the Distributed Programming Environment paper prepared by Lisa Krause for more information about DPE.

Initial experiences with CF90 1.0 have been favorable. A third party vendor's Fortran 90 version of their library package (NAG)

containing over two hundred thousand lines of Fortran 90 code has been ported and tested using CF90 with very few (less than 10 distinct) problems. A small number of these problems were due to CF90 problems. The remainder were due to differences in implementation due to interpretation of the Fortran 90 standard or differences in machine arithmetic between Cray and the platform where these libraries were developed. While it is very early in the release cycle of the CF90 product, the cumulative failure profile, based on weighted severity of SPRs, looks very favorable compared with the last three major CF77 releases. There are very few Fortran 90 production codes running currently; this may also influence the favorable cumulative failure profile.

2.2 Performance of CF90 1.0

CF90 release criteria included four measures of the programming environment's performance. These were based on CF77 5.0, as CF77 6.0 was not released until late in the CF90 1.0 development cycle. These criteria were for the performance of generated code, size of generated code, compile time, and compile size. Specifically, these criteria were:

- The geometric mean ratio (GMR) of execution time for code generated by CF90 compared to that of CF77 5.0 shall not exceed 1.15 for the six performance benchmark suites: Perfect Club, Livermore Kernels, NAS Kernels, Linpack Kernels, Cray 100 kernels, and the New Fortran Performance Suite (NFPS). Note: NFPS is a collection of Cray applications and some Cray user proprietary codes used to measure Fortran performance by the Performance Test and Evaluation section.
- The size of code generated by CF90 shall not exceed that of CF77 5.0 by more than 20% when measured using the Perfect Club and NFPS benchmark suites.
- Compile time of CF90 shall not exceed twice that of CF77 5.0 when measured using the Perfect Club and NFPS benchmark suites.

- Compile size shall not exceed twice that of CF77 5.0 when measured using the Perfect Club and NFPS benchmark suites.

The rationale for these criteria was based on the fact the CF90 uses new performance technology for optimization, vectorization, and tasking which is not nearly as mature as that of the CF77 compilation system. Also, these measurements are based on the CFT77 compiler alone, and CF90 has integrated the analysis and restructuring capabilities of FPP and FMP into the CF90 compiler itself. Since the CF90 compiler combines the three phases of CF77 into a single phase, it was expected that CF90 would exceed the compile time and space requirements of the CFT77 compiler.

Three of four of these criteria were exceeded. Execution performance for the various suites is as follows (GMR of CF90 to CF77):

Perfect Club	1.06
NFPS Suite	1.09
Livermore Kernels	0.97
NAS Kernels	1.02
LINPACK Kernels	0.99
Cray 100 Kernels	1.09

A number less than one indicates that CF90 took less time than CF77.

For the Perfect Club Suite, the execution size, compile time, and compile sizes are respectively, 1.05, 2.22, and 1.82 when compared to CF77 5.0. For the NFPS tests, the execution size, compile time, and compile sizes are 1.08, 2.46, and 1.72 respectively compared with CF77 5.0. The compile time goals which were not met are being addressed in revisions of the 1.0 release, and in the 2.0 release of CF90.

An additional known performance issue involves codes that make use of the Fortran 90 array transformational intrinsic functions. These intrinsic functions are generic in their definition, operating on many differing data types, arbitrary dimensions of arrays of any dimensionality, with one or more optional arguments which change the behavior of the function based on the presence or value of the argument. In short, each intrinsic may exhibit a number of special cases that can be optimized differently based on the specific way in which it is called. CF90 1.0 has implemented these intrinsic functions as external calls. As external calls, all local optimization is inhibited. Loop fusion of neighboring array syntax statements cannot occur; statements calling the array intrinsic functions must be broken up into multiple

statements, and each call of these functions may require allocation and deallocation of array temporaries as well as copy-in/copy-out semantics for the array arguments. Inlining of these intrinsics removes many of the optimization barriers and the need for many of the array temporaries. Revisions of 1.0 and release 2.0 will implement automatic inlining of many instances of these functions. The 1.0.1 revision targets the MATMUL, CSHIFT, TRANSPOSE, and DOT_PRODUCT intrinsic functions. Future revisions will inline additional array intrinsic functions. The benefits of this inlining are seen not only in execution time, but also in compile time and size.

2.3 Status of Fortran 90 Interpretation Processing

Since Fortran 90 became an American and international standard, there have been approximately 200 formal requests for interpretation or clarification of the intent of the standard. At present, about 50 of these requests are still under initial consideration. The majority of these requests are for clarification. Some have required edits to the standard which will appear in later revisions of the Fortran language specification. Cray has attempted to take a conservative approach in our implementation of the Fortran 90 language in areas open to interpretation. Thus, the current implementation may be more restrictive than what it may need to be when the interpretations are resolved. Most outstanding interpretations involve end cases in the implementation, and if resolved differently than the current CF90 implementation, changes would result in a syntax or semantic error in future versions of CF90. However, there are a small number of interpretations pending which may be resolved in a manner which could result in runtime behavior different from current CF90 runtime behavior. If these interpretations are resolved in a manner incompatible with the current CF90 implementation, an appropriate mechanism will be provided so that current behavior will continue to be supported for at least one year after the change occurs. This may take the form of supporting old and new behavior within a compilation with a message issued stating old behavior will be removed at some release level, or, in some cases may require a compile time switch to select old and new behavior.

3.0 CF77 to CF90 Transition Plan

Cray Research has selected CF90 as our primary Fortran compiler for the future. CF90 is based on the new Fortran 90 standard.

Portability is maintained with CF77 codes as Fortran 90 is a superset of FORTRAN 77, and CF90 supports CF77 language extensions. CF90 will further promote portable parallel applications as it is introduced on SPARC platforms, providing the same programming environment on workstations as on Cray platforms. As CF90's integrated optimization, vectorization and tasking technology matures, it will provide better performance than that of CF77, without the use of preprocessors.

CF77 release 6.0 was the final major feature of the CF77 compilation system. There will be no more major feature releases, only bugfixes. New hardware support will be provided in revisions as necessary. Revision support of CF77 6.0 will continue through third quarter of 1996.

The CF77 to CF90 transition period is implemented in two phases beginning in fourth quarter 1993 and continuing through third quarter 1996. This allows gradual migration from CF77 to CF90. During this transition period, new codes can be developed using Fortran 90 features, and CF77 applications can be gradually ported to the CF90 environment.

The first phase of the transition began with the release of CF90 1.0 and will end when CF90 provides full functionality of the CF77 compilation system and delivers equal or greater performance than CF77. This phase is planned to end with a revision of CF90 2.0 in 1995. Phase two begins at this time and continues through the subsequent release of CF90.

Existing customers can upgrade their CF77 license to a CF90 license during phase one by paying the difference in price of a CF90 license over a CF77 license for their system. CF90 maintenance fees will include CF77 maintenance. Customers who received CF77 bundled with their system receive full credit for a CF77 paid up license. During phase two, existing customers must pay the full price for a CF90 license. The CF90 maintenance price will then include CF77 maintenance also.

New customers and customers upgrading their systems during phase one can purchase a CF90 license and will also receive CF77. Maintenance will only be paid for CF90, but will include both CF77 and CF90 maintenance. After the initial release of CF90, CF77 will not be sold separately to new customers. During phase two, new customers and customers upgrading their systems can only purchase CF90. CF77 will not be available to these

customers during phase two. 4.0 Feature plans for CF90 2.0

Release 2.0 of the CF90 programming environment is planned for mid-year 1995. The primary focus of this release is to provide features of the CF77 programming environment not available with CF90 1.0, and to equal or surpass the performance provided by the CF77 compiling system. Initial MPP support is planned, and cross compilers running on workstations targeting Cray platforms will be introduced in the second release of the Distributed Programming Environment.

The features planned for CF90 2.0 include automatic inlining of external and internal procedures, runtime checking of array bounds and conformance, runtime checking of argument data types and counts, and runtime character substring range checking. Conditional compilation facilities will be provided by the compiler. This feature is available to CF77 users through the gpp preprocessor. User defined VFUNCTIONs written in Fortran are planned, though the HPF syntax of PURE and ELEMENTAL procedures may be used instead of the VFUNCTION directive. No support is planned for CFPP\$ directives, though similar functionality may be provided for tuning PDGCS optimization and restructuring through a new directive set.

Inlining will be similar to CFT77's automatic inlining capabilities. CF90 will use a textual interface that produces files in an intermediate format. Inlining will work off these intermediate files. This will allow the frontend of the compiler to run as a stand alone process producing intermediate text files. The inliner and optimization/code generation phases can also be run independently. The use of the textual interface is intended to allow for cross language inlining and additional interprocedural optimizations in future releases. No source to source inlining capability (similar to FPP's inlining) is planned in the 2.0 time frame.

CF90 2.0 for SPARC platforms will provide 128 bit floating point support (Cray DOUBLE PRECISION) and 64 bit integer and logical data types. CF90 1.0 on SPARC platforms supports only 32 bit integer and logicals, and 32 and 64 bit floating point representations.

For MPP platforms, CF90 2.0 will be the first release of the full Fortran 90 language. CF90 will provide packed 32 bit integer, real, and logical data types which will allow applications to achieve twice the memory and I/O bandwidth permitted by 64 bit data types. In addition, CF90 will support 128 bit

floating point data types, which are not currently available with CF77 on MPP platforms.

5.0 Summary

The Cray Fortran 90 Programming Environment was released in December 1993, beginning the transition from CF77 to CF90. Initial experiences have been promising. Performance of generated code is near that of CF77, and in some cases exceeds that of CF77. Subsequent releases of CF90 will move the programming environment to additional platforms, aiding portability of applications across platforms. The CF90 2.0 release will match the capabilities and performance of CF77, and introduce cross compilers. In 1995, CF90 should become the Fortran environment of choice for Cray users.

Autotasking, CRAY, CRAY Y-MP, and UNICOS are federally registered trademarks and CF77, CFT77, CRAY-2, SEGLDR, Y-MP, and Y-MP C90 are trademarks of Cray Research, Inc.

 **Fortran 90 Update**

Jon Steidel

Cray Research
Software Development



 **Fortran 90 Update**

- ◆ Status
- ◆ CF77/CF90 Transition Plan
- ◆ Plans



 **Fortran 90 Status**

- ◆ **CF90 Programming Environment 1.0**
 - Cray Y-MP, C90, and YMP-EL release December 23, 1993
 - 45 Licenses
 - 28 Systems, 21 Customers
 - 29 EL Systems
 - CF90 Sparc Native programming environment 3Q94
 - Initial Fortran90 MPP support release 2.0
- ◆ **Distributed Programming Environment Release 1.0**
 - 3Q94
 - Lisa Krause Thursday 8:30



 **Fortran 90 Status**

- ◆ **1.0 Performance**
 - Execution Times
(GMR of CF90 1.0 to CF77 5.0)
 - CF90 1.0 Goal 1.15
 - Perfect Club 1.06
 - NFPS 1.08
 - Livermore Kernels .97
 - NAS Kernels 1.02
 - LINPACK Kernels .99
 - Cray 100 Kernels 1.09



 **Fortran 90 Status**

	Execution Size	Compile Time	Compile Size
CF90 1.0 Goal	1.2	2.0	2.0
Perfect Club	1.05	2.22	1.82
NFPS	1.08	2.46	1.72



 **Fortran 90 Status**

- ◆ **Array intrinsics**
 - 1.0 performance poor
 - External calls inhibit optimization
 - Heavy use of temporaries, copy in/out
 - Difficult to inline
 - Optional arguments
 - Variable number of dimensions
 - Many special cases
 - Inlining of some cases in 1.0 revisions
 - MATMUL, CSHIFT, DOT_PRODUCT, TRANSPOSE in 1.0.1





Fortran 90 Status

Fortran 90 Interpretation processing

- ◆ Over 200 formal questions about meaning of the standard
 - Some edits to the standard required
- ◆ X3J3 and WG5 approval required
 - WG5 approved will be in F95 standard
 - Approximately 50 yet unresolved
- ◆ Some interpretations may require changes to CF90



21-03877



CF77 to CF90 Transition

- ◆ Cray Research has selected CF90 as our primary Fortran compiler for the future
 - New Fortran standard
 - Portability
 - Fortran 90 is a superset of Fortran 77
 - CF77 extensions supported
 - Portable path to parallel applications
 - Performance
 - Advanced compiler technology
 - Integrated Autotasking and optimization



21-03874



CF77 to CF90 Transition

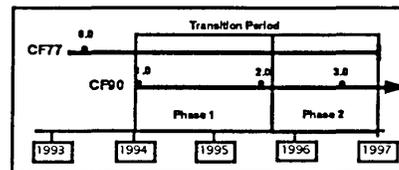
- ◆ CF77 6.0 will be the final major release of the CF77 compiler (no more major features, only bug fixes). Actively supported through 3Q96.
- ◆ Transition period will be implemented in two phases from 4Q93 through 3Q96 to allow for gradual migration from CF77 to CF90.
- ◆ Development of F90 applications
- ◆ Porting of CF77 applications



21-03875



CF77 to CF90 Transition Plan



21-03876



CF77 to CF90 Transition (for existing CF77 customers)

- ◆ Phase 1:
 - Existing CF77 customers can upgrade to CF90 by paying the "Delta" licensing and maintenance fees.
 - CF90 upgrade licensing price = CF90 price - CF77 price
 - CF77 support is bundled with CF90 maintenance.
 - Customers who received CF77 bundled with system received full credit for CF77 paid up license
- ◆ Phase 2:
 - CF90 license upgrades will be full price
 - CF77 support is bundled with CF90 maintenance.



21-03877



CF77 to CF90 Transition (for new customers and system upgrades)

- ◆ Phase 1:
 - New customers and system upgrades purchase CF90 license only. CF77 is bundled with CF90.
 - Maintenance will only be paid for CF90.
 - After the initial release of CF90, CF77 will not be sold separately to new customers.
- ◆ Phase 2:
 - New customers and system upgrades can only purchase CF90. CF77 will not be available to these customers.



21-03877



CF90 2.0 Plans

- ◆ 2Q95 release
- ◆ Full CF77 functionality
 - Inlining (with textual interface)
 - Runtime checking
 - Array bounds and conformance checking
 - Runtime argument checking
 - Substring range checking
 - Conditional compilation
 - User defined VFUNCTIONS
 - May use HPF ELEMENTAL and PURE syntax
 - No support for CFPPS directives



231108411



CF90 2.0 Plans

- ◆ MPP support
 - Full Fortran 90 feature support
 - Packed 32 bit data types
 - INTEGER, LOGICAL, REAL
 - 64 bit COMPLEX (2 32 bit reals)
 - 128 bit floating point representation



231108411



CF90 2.0 Plans

- ◆ SPARC features
 - 128 bit floating point
 - 64 bit integers
- ◆ DPE Features
 - Target machine arithmetic simulation (constant folding)
 - Cross compilers
 - SPARC host, Cray PVP target
 - Usa Krause, Thursday 8:30



231108411

C AND C++ PROGRAMMING ENVIRONMENTS

David Knaak
Cray Research, Inc.
Eagan, Minnesota

Overview

Cray Research has released Standard C programming environments for both Cray parallel vector (PVP) systems and Cray massively parallel (MPP) systems that include high-performance, reliable compilers and supporting tools and libraries. A C++ compiler for PVP systems has been released as a first step towards a high-performance C++ compiler. A C++ compiler for MPP systems will soon be released. A transition will occur over the next few years from separate C and C++ compilers to a single high-performance C/C++ compiler, which will be part of a rich programming environment. High-performance C++ class libraries from Cray Research and third-party vendors will complement the C++ environment. The direction for parallel C and C++ is still under study.

Historical Perspective on C and C++ Programming Environments

Prior to 1993, Cray Research compilers were released asynchronously from the libraries and tools. This made it difficult to coordinate the delivery of new functionality when it required changes to both the compiler and to the tools or libraries. In 1993, Cray Research released several "programming environments", integrating the compiler, the supporting tools, and the supporting libraries into a single package.

The compilers in these programming environments need to be high-performance compilers. While the C and C++ languages are considered good for system implementation, they are not necessarily good for numerical programming. Several years ago, Cray Research committed to delivering a high performance C compiler suitable for numerical programming. We have achieved this goal with a combination of language extensions, optimization directives, and automatic optimizations. We are now committed to also delivering a high performance C++ compiler. As was our goal for C, our goal for C++ is to achieve performance at or near Fortran performance (within 20%) for equivalent codes.

The transition to a high-performance C++ compiler starts with standard-compliant functionality and will progress in each release with greater functionality and with better performance that requires less programmer effort. Supporting tools and libraries will also be enhanced. A specific product transition plan will be presented at a future CUG meeting. We will collaborate with customers on important applications to help guide our functionality and performance enhancements.

Standard C Programming Environment 1.0 for PVP

The Standard C Programming Environment 1.0 was released for Cray PVP systems in December of 1993. The components of the PVP version are:

- SCC
- CrayTools: CDBX, xbrowse, PVP performance tools, [cclint in 1.0.1 release]
- CrayLibs: libm, libsci

The original goal for the Cray Standard C compiler was to provide an ISO/ANSI compliant compiler that delivers Cray performance at or near the performance level of equivalent Fortran code. This goal was achieved with the 2.0 release and improved on with the 3.0 and 4.0 releases. Several language extensions were added to provide some Fortran-like capabilities that were necessary for delivering the performance. These language extensions have been proposed to the ISO/ANSI C committees.

Reliability of the compiler has improved with each release and is quite good.

Standard C Programming Environment 1.0 for MPP

The Standard C Programming Environment 1.0 was released for Cray MPP systems in November of 1993. The components of the MPP version are:

- SCC
- CrayTools: TotalView, xbrowse, MPP Apprentice, [cclint in 1.1 release]
- CrayLibs: libm, libsci, libpvm3

The 1.0 programming environment for MPP supports multiple PE execution only through message passing. Two message passing models are available: PVM and shared memory get and put. Architectural differences between PVP and MPP systems necessitate some significant compiler feature differences. On MPP systems, memory is distributed so distinctions must be made between local and remote memory references. On PVP systems, type "float" is 64-bits but on MPP systems, type "float" is 32-bits. The computation speed for 32-bit floats is the same speed as for 64-bit doubles, but arrays of floats are packed 2 values per 64-bit word and so less memory is used and bandwidth to and from memory or to and from disk is double. Same story for shorts. Vector capabilities don't exist on MPP systems. MPP systems have additional intrinsics and don't support PVP-specific intrinsics.

Cray C++ 1.0 for PVP

Cray C++ 1.0 for PVP was released in August of 1992. It was not released as part of a full programming environment, rather, it depends on the C programming environment for compilation of the generated C code and for the supporting tools and libraries. C++ 1.0 is an enhanced version of USL C++ 3.0.2 (cfront). The major enhancements are pragma support, additional inlining, and restricted pointers.

The functionality of Cray C++ 1.0 matches that of the emerging C++ standard as it was at that time and the performance is adequate where performance is not a critical concern. Where performance is a critical concern, good performance has been achieved in some cases, often with programmer intervention. There is still room for improvement. Currently, performance is highly dependent on programming style. The tech note SN-2131 provides some guidance for which techniques work best.

The C++ debugging support was significantly enhanced with the release of SCC 4.0, CDBX 8.1 (both in the C programming environment 1.0) and with C++ 1.0.1.

There are currently about 40 CRI customers licensed for C++. If we had not released C++ 1.0, many customers would have ported the USL code themselves, duplicating work and not benefiting from our enhancements.

Cray C++ 1.0 for MPP

C++ 1.0 for MPP will be released mid-1994. As with SCC for MPP, C++ 1.0 for MPP supports PVM and shared memory get and put models only. TotalView will support C++, including handling of mangled names. At about the same time, the MathPack.h++ and Tools.h++ class libraries for MPP will be released as separate unbundled products.

Some C++ Customer Experiences

Some significant performance successes have been achieved with the current compiler technology. But this has required changes to user code, changes to C++ 1.0, and changes to SCC. To illustrate that C++ codes can perform quit well, two examples are described below.

Application 1

This application contained the following C++ statement in the kernel of its code:

```
t(i, j) += temp * mat1.index(i, j);  
// t, temp, mat1 are complex class objects
```

Though this seems like a very simple statement, it is expanded by C++ 1.0 into about 150 lines of C code that SCC must compile. (It really doesn't need to be quite that complicated.) For this statement, the equivalent Fortran or C code would contain a loop and perhaps function calls. The class involved here, complex, is a relatively simple class but the optimization issues are general and apply to many classes. In particular, though it is easier for the compiler to optimize operations on objects that contain arrays, the more natural style of writing C++ codes is usually to have an array of objects. This code uses arrays of objects. The initial performance (on a single CRAY C90 CPU) for this part of the code with SCC 3.0 was about 9 MFLOPS. Using a more aggressive optimization level, -h vector3, resulted in partial vectorization of the loop and the performance reached about 135 MFLOPS. Using SCC 4.0 and the compiler options -h ivdep,vector3 resulted in performance of about 315 MFLOPS.

Application 2

This application is a hydrocode that simulates the impact of solid bodies at high velocities. The code has been written to be run on a wide variety of architectures including scalar, parallel vector, and massively parallel. A major focus was developing appropriate classes for the application and tuning the classes for different architectures. This keeps the main code quite portable. Another advantage of this approach is that all the architecture-dependent features are buried in the class definitions and remain invisible to the class users. The initial performance of the entire application on a CRAY Y-MP was a about 1 MFLOP. By using reference counting and management of temporaries to eliminate unnecessary memory allocations and deallocations, the performance reached about 25 MFLOPS. Using restricted pointers and more aggressive inlining brought that up to 75 MFLOPS on 1 processor of the Y-MP. The code has also been ported to a CRAY T3D using a message passing system based on get/put. The code runs a typical real-life problem at about 5.8 MFLOPS per PE with excellent scaling to at least 128 PEs.

C++ Programming Environment 2.0

The focus for C and C++ programming environments in 1994 and 1995 is to provide a C/C++ programming environment with a native, high performance C++ compiler and enhanced tools support of C++. The 2.0 release is planned for mid-1995. In order to focus our resources on the 2.0 environment, there will be no major release of the C programming environment in 1994 and 1995. There will be revision releases of the C and C++ environments as necessary to fix problems, to support new hardware, and perhaps to provide some performance enhancements.

The 2.0 environment will include full tools support for C++. This will include TotalView, xbrowse, MPP Apprentice, and possibly a class browser.

For the MPP environment, distributed memory versions of some libsci routines will be available in 2.0. These will include some or all of BLAS2, BLAS3, LAPACK, FFT, and sparse solvers.

The C++ 2.0 compiler will utilize more advanced compiler technology. The advantages of the new compiler technology are:

- eliminates the step of translation from C++ code to (messy) C code
- able to pass C++ specific information to the optimizer
- same optimization technology as CF90 compiler
- new internal data structure has potential for:
 - lower memory usage
 - higher throughput
 - inter-procedural optimization
 - inter-language inlining

C++ 2.0 will be functionally equivalent to C++ 1.0 with somewhat better performance, and with exception handling. C++ 2.0 will be an ISO/ANSI C compliant compiler but won't have all of the functionality of SCC 4.0. The C++ 2.0 programming environment will not be dependent on the Standard C programming environment. The target date for C++ 2.0 is mid-1995.

C++ Programming Environment 3.0

The target date for C++ 3.0 is mid-1996. C++ 3.0 will keep up with emerging C++ standard and will have the functionality of SCC 4.0 with maybe a few minor exceptions. C++ 3.0 will outperform SCC 4.0 for C code. C++ 3.0 will outperform C++ 2.0 for C++ code.

C++ and Class Libraries

The productivity advantage of C++ is realized when well defined and optimized classes are available to the end user that match his or her discipline. C++ can be used as a meta language that allows the class implementer to define and implement a new language that suits the discipline. The class user can then think in terms of the discipline rather than in terms of computer science. It does put a greater burden on the class implementer. This division of labor is a net gain if the classes are used for more than one application. The more complicated the hardware gets, the more important it is that software help hide this complexity from the user.

Cray Research has been and will be very selective in which C++ class libraries we distribute. Any that we do support are, and will be, separately licensed and priced. We believe that there should be, and will be, a smorgasbord of class libraries from various vendors in the future. This gives customers the greatest choice

and will allow even small vendors to compete in niche markets.

The 2 class libraries that Cray has released are MathPack.h++ 1.0 and Tools.h++ 1.0 for PVP, released in December 1993. MathPack.h++ 1.0 and Tools.h++ 1.0 MPP will be released mid-1994. As with the evolution of the compilers, this first release of class libraries focused primarily on providing functionality and then as good of performance as time would allow. These libraries are based on Rogue Wave libraries and therefore have the benefit of portability of code across many platforms. Industry standards (even de facto standards) for class libraries are needed so that the portability benefits can be reaped. More tuning of the libraries will improve the performance.

Parallel C and C++

Cray Research has not committed to any C or C++ language extensions for parallelism, or for distributed memory systems. We are still looking at various models, experimenting with some, and encouraging others to experiment. Any model we select must have potential for high performance. We are also watching the parallel Fortran models to see what techniques and paradigms prove most successful. We want to hear about customers' experiences with parallel extensions.

Summary and Conclusion

The Standard C programming environments for PVP and MPP include high-performance and reliable compilers. We have not yet achieved a high level of automatic high-performance for C++. We are making the transition of the next few years to a single, high performance C/C++ compiler that will be part of a rich programming environment. In the short term, good C++ performance still requires work by the programmer. We will work with customers on important applications to guide our efforts at enhancing the compiler, tools, and libraries.

In the long run, the success of C++ will depend on:

- compilers that deliver the performance of the hardware with minimal programmer intervention,
- good supporting tools and libraries,
- well defined and implemented class libraries for different disciplines,
- and overall, a significant improvement over other languages in ease of use and time to solution. Cray

Research has taken on the challenge to do its part to make C++ a success.

The MPP Apprentice™ Performance Tool: Delivering the Performance of the Cray T3D®

Winifred Williams, Timothy Hoel, Douglas Pase

Cray Research, Inc.
655-F Lone Oak Drive
Eagan, Minnesota 55121

ABSTRACT

The MPP Apprentice™ performance tool is designed to help users tune the performance of their Cray T3D® applications. By presenting performance information from the perspective of the user's original source code, MPP Apprentice helps users rapidly identify the location and cause of the most significant performance problems. The data collection mechanism and displays scale to permit performance analysis on long running codes on thousands of processors. Information displayed within the tool includes elapsed time through sections of code, time spent in shared memory overheads and message passing routines, instruction counts, calling tree information, performance measures and observations. We will demonstrate how the MPP Apprentice's guides the user's identification of performance problems using application examples from benchmarks and industry.

1 Introduction

Fast hardware and system software are great, but the performance that really counts is that which the end user can achieve. Early MPPs were notoriously difficult to program, and it was even harder to approach the peak performance of the systems. While this has been improving across the board, Cray has a particularly strong story to tell. The Cray T3D has very fast hardware [CRI93-1], system software, and libraries which overcome the limitations of many other systems. Cray's MPP Fortran programming model [Pa93], along with hardware support from the Cray T3D, attacks the programming issue by letting the user program a distributed memory system as though the memory were shared. Good compilers provide the performance. But there is always more the user can do to improve the performance of a code.

The focus of the MPP Apprentice tool is to deliver the performance of the Cray T3D to the user. The tool assists the user in the location, identification, and resolution of performance problems on the Cray T3D. Its instrumentation and displays were designed specifically to permit performance analysis on thousands of processors, to handle the longer running codes that large systems enable, and to support work-sharing, data parallel, and message passing programming models. Performance characteristics are related back to the

user's original source code, since this is what the user has the ability to change in order to improve performance, and are available for the entire program, or a subroutine, or even small code blocks.

2 Possible approaches

Many performance tools today collect a time-stamped record of user and system specified events, more commonly called event traces. AIMS[Ya94], GMAT [CRI91], Paragraph [He91], and TraceView [Ma91] are examples of event trace based tools. These tools present information from the perspective of time, and try to give the user an idea of what was happening in every processor in the system at any given moment. While event trace tools have great potential to help the user understand program behavior, the volume of data is difficult to manage at run-time and when post-processing. The size of an event trace data file is proportional to the number of processors, the frequency of trace points, and the length of program execution. Handling such large data files at run-time can perturb network performance, interprocessor communication, and I/O. At post-processing time, the volume can be difficult to display and interpret. The data volume also raises concerns about how event trace tools will scale to handle longer running programs and larger numbers of processors.

Other tools record and summarize pass counts and elapsed times through sections of code, and will be

Copyright © 1994. Cray Research Inc. All rights reserved.

called "stopwatch" tools for the purposes of this paper. ATEExpert [Wi93] and MPP Apprentice [CRI93-2, Wi94] are examples of these tools. Stopwatch tools present information from the perspective of the user's program and provide the user with performance characteristics at a particular point in the program. Profiling tools present information from a similar perspective, although their data collection mechanism is very different. The data volume of stopwatch tools, which is proportional to the size of the program, is much smaller than event trace tools. The reduced data volume intrudes less on program behavior and permits much finer grained data collection. The data volume is also more manageable for a tool to post-process and a user to interpret, and scales well to growing numbers of processors and lengths of program execution. The MPP Apprentice tools is a stopwatch tool.

3 MPP Apprentice method

A compile-time option produces a compiler information file (CIF) for each source file and turns on MPP Apprentice instrumentation. A CIF contains a description of the source code from the front end of the compiler and encapsulates some of the knowledge of the user's code from the back end of the compiler, such as instructions to be executed and estimated timings. The instrumentation occurs during compilation after all optimizations have occurred. Instructions are added strategically to collect timing information and pass counts while minimizing the impact on the user's program.

During program execution, timings and pass counts for each code block are summed within each processor and kept locally in each processor's memory, enabling the MPP Apprentice to handle very long running codes without any increase in the use of processor memory. At the end of program execution, or when requested by the user, the power of the Cray T3D is used to sum the statistics for each code object across the processors, keeping high and low peaks, and a run-time information file (RIF) is created. The MPP Apprentice post-processes the RIF, the CIFs, and the user's source files.

4 Visualization of data

When a user initially runs the MPP Apprentice on their code, the tool provides a summary of the statistics for the program and all subroutines, sorting them from the most to the least critical. The list of subroutines includes both instrumented as well as uninstrumented subroutines, such as math and scientific library functions. MPP Apprentice defines long running routines as "critical", and permits the user to redefine "critical" if desired. The summary breaks down the total time for each instrumented subroutine into time spent in over-

head, parallel work, I/O, and called routines. For uninstrumented subroutines, only the total time is available. Overhead is defined as time that would not occur in a single processor version of the program, and includes PVM communication, explicit synchronization constructs (such as barriers), and implicit synchronization constructs (such as time spent waiting on data, or implicit barriers at the end of shared loops). MPP Apprentice details the exact amount and specific types of overhead.

Figure 1 shows a sample main window of MPP Apprentice. The upper panel of the window, or navigational display, shows summarized statistics for the program and each of its subroutines. The legend window identifies the breakdown of the total time for each code object. A small arrow to the right of a subroutine name indicates that the subroutine has been instrumented and that it may be expanded to see performance characteristics of nested levels. All of the detailed information available for the program and subroutines is also available for nested levels. Nested code objects are identified by a name identifying the code construct, e.g., If or Do, and a line number from the original source code. The user may ask to see the full source code for a code object. A source code request invokes Cray's Xbrowse source code browser, loads the appropriate file automatically, and highlights the corresponding lines of source code (See Figure 2).

The middle panel of MPP Apprentice details the costs for the code object selected in the navigational display. It toggles between providing information on instruction counts, shared memory overheads, and PVM overheads. When displaying instruction counts, the exact number of each type of floating point and integer instruction is available, as well as the number of local, local shared, and global memory loads and stores. These values assist the user in balancing the use of the integer and floating point functional units and maximizing local memory accesses to fully utilize the power of the Cray T3D. The shared memory overheads display gives the amount of time spent in each type of synchronization construct as well as time spent waiting on the arrival of data. The PVM overheads display gives the amount of time spent in each type of PVM call. Samples of each of these displays is available in Section 5.

A call sites display helps look at algorithmic problems related to the calling of one subroutine from another. It lets the user see all the sites from which a selected subroutine was called, and all the subroutines to which the selected subroutine makes calls. Timings and pass counts are available with this information.

MPP Apprentics

File Displays Navigate Options **CRAY** HELP

NAVIGATIONAL DISPLAY

Time in called subroutines: Exclude Include Legend

Total Time	Object
1.93e+09	PROGRAM
1.2e+09	PVMFRECVC
8.58e+08	GLOBAL SUM
3.55e+08	MY_PROC
3.31e+08	COORD
1.09e+08	fcd cmp eq
1.01e+08	_SORT
5.19e+07	ELEMENT
2.55e+07	GETINPUT
2.14e+07	REDUCT

9.67e+08 1.93e+09
Time (usec)

COSTS: Instructions Shared Memory Overhead PVM Overhead

Time	Name
5.05e+04	PVMFBARRIER
1.02e+03	PVMFBCAST
1.94e+06	PVMFINITSEND
7.45e+04	PVMFJOINGROUP
218	PVMFMYTID
4.69e+06	PVMFPACK
1.2e+09	PVMFRECVC
6.28e+06	PVMFSEND
1.65e+07	PVMFUNPACK
220	PVMFGETPE

6.01e+08 1.2e+09
Time (usec)

INFORMATION

menu.
Opening file: /home/sunmac8/ww/chem/app.rif

Legend

NAVIGATIONAL LEGEND

- Overhead Time
- Parallel Work Time
- Called Routine Time
- I/O Time
- Uninstrumented Routine Time

Close

Context Sensitive Help

CONTEXT SENSITIVE HELP

Navigational Display
Selecting items (code objects) in this window updates all other windows accordingly. Code objects with arrows maybe be expanded to provide performance information on nested levels by selecting the arrow or using the navigate menu.

Close

Figure 1: Main window of the MPP Apprentice tool

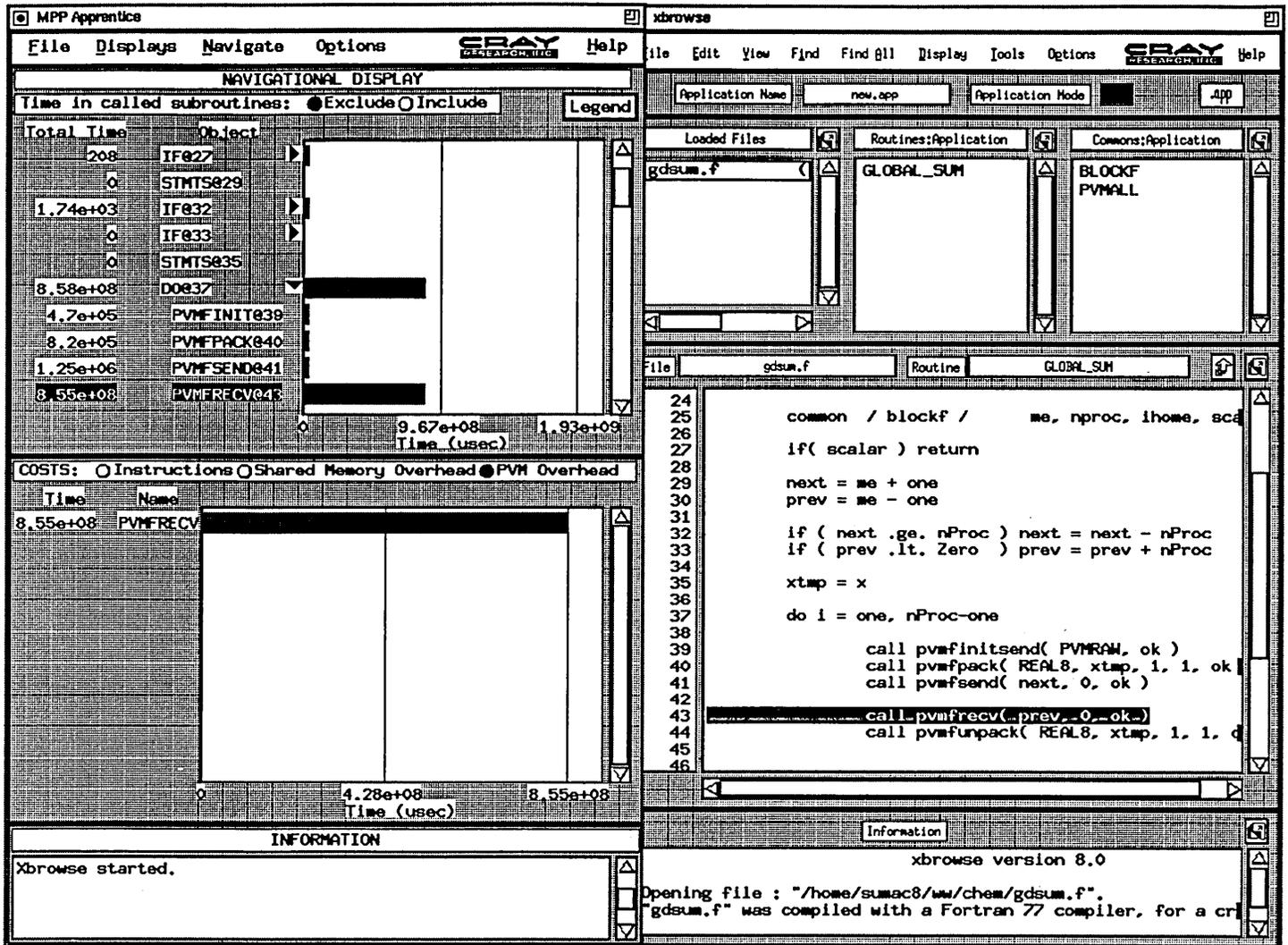


Figure 2: MPP Apprentice tool working cooperatively with Xbrowse

A knowledge base and analysis engine built into MPP Apprentice derives secondary statistics from the measured values. It provides the user with performance measures (such as MFLOPs ratings), analyzes the program's use of cache, makes observations about the user's code, and suggests ways to pursue performance improvements. It attempts to put the knowledge of experienced users into the hands of novices. Since the Cray T3D is a relatively new machine, even experienced users have a lot to learn, so the knowledge base is expected to grow.

5 Identification of Performance Problems

Many of the commercial and benchmark codes optimized with the MPP Apprentice so far have had a large amount of time spent on a synchronization construct as their primary performance bottleneck. Time at synchronization constructs indicates some type of imbalance in the code, a problem that is typical of MPP codes in general. The interconnection topology of the Cray T3D is much faster than other commercially available MPPs, but these performance problems while reduced substantially, still exist.

5.1 Load imbalance with a message passing code

Figure 1 shows the initial MPP Apprentice output for a chemical code. With the subroutines sorted in order of decreasing total time, it is evident that PVMFRCV, a blocking message receive function, is consuming the most time, more than half of the total time of the program. Intuitively it seems undesirable to spend more than half of program execution time waiting for messages. By selecting subroutine PVMFRCV in the Navigational Display and taking a look at the call sites display, we can see each call to PVMFRCV and the amount of time spent in it.

In Figure 3 we can see all of the calls to PVMFRCV. There are not many of them, but two calls are taking substantially more time than the others. If we were to select the units button and switch to viewing pass counts we would see that for the same number of calls to PVMFRCV from different call sites, call times vary significantly. Some type of a load imbalance

exists. By resolving this problem developers realized a speedup of three and a half times.

5.2 Load imbalance with Fortran programming model code

The main window in Figure 4 is from the NAS SP benchmark code. This code uses one of the synchronization constructs available as part of Cray's Fortran Programming Model, a barrier. The navigational display makes it evident that the barrier is the most critical subroutine in the user's program. With the middle panel toggled to show shared memory overheads, the time spent in the barrier shows up as a type of overhead. Since a barrier is a subroutine call, the user could approach this similarly to the PVM problem shown previously by using the call sites display to view the locations from which the barrier was called, and the time spent at each location.

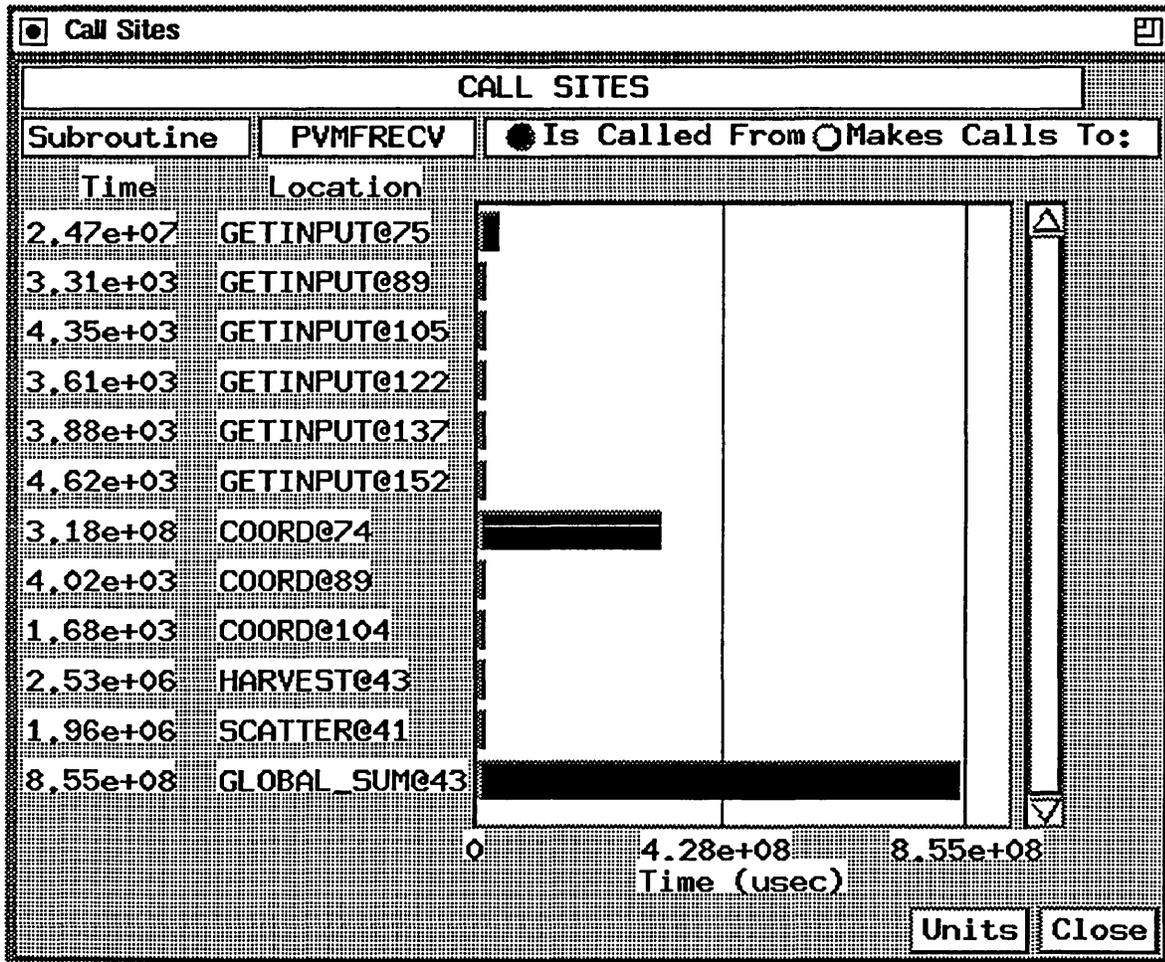


Figure 3: Call sites display for PVMFRCV

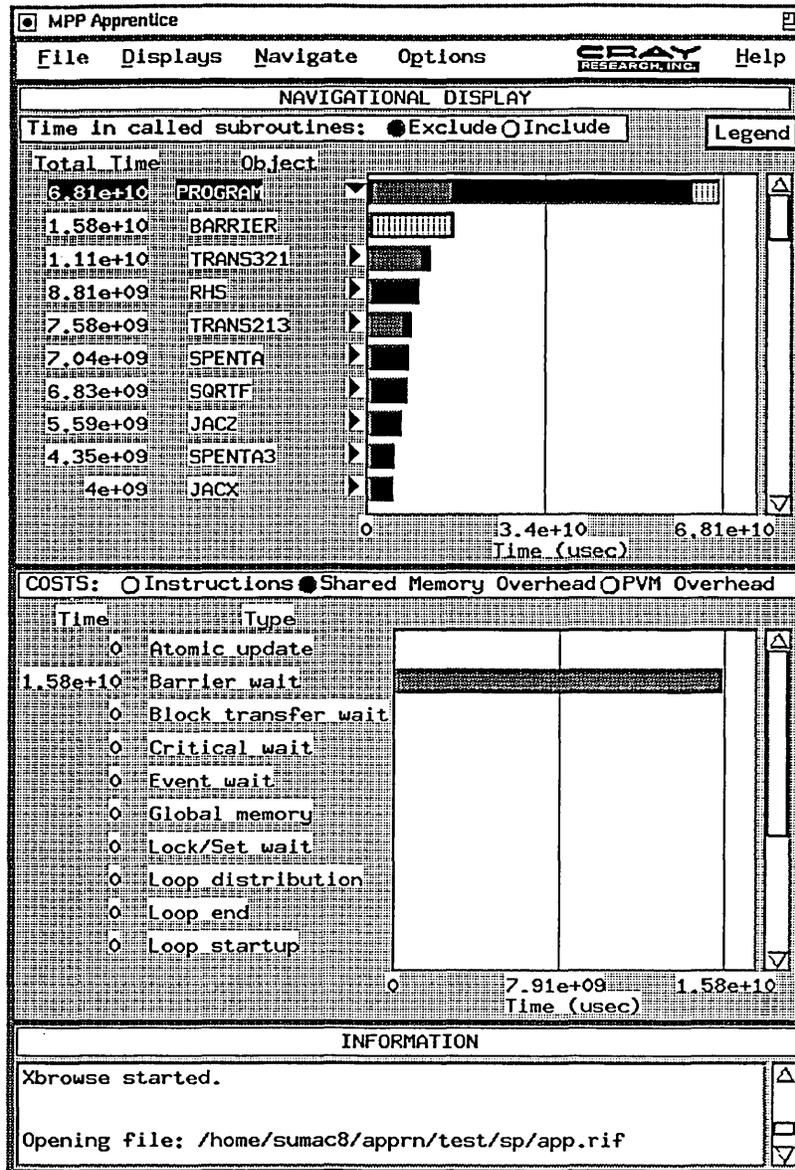


Figure 4: Main window of CRAFT code with a barrier

5.3 Poor balance of floating point and integer calculations

Figure 5 shows the main window for a hydrodynamics code. The most critical subroutine is an uninstrumented subroutine called `$$sldiv`. If the user were to search the source code, `$$sldiv` would not be found.

The observations window in Figure 6 notes and explains the time spent in `$$sldiv`. Since there is no integer divide functional unit on the alpha chip used in the Cray T3D, a subroutine call must be made to do the divide. Since the cost to call a subroutine is signif-

icantly greater than the direct use of a functional unit, the program performance will benefit by limiting the number of integer divides. If we return to Figure 5, and look at the middle panel which is currently displaying instruction counts, we will note a large number of integer instructions relative to floating point instructions. Since there are both integer and floating point functional units which may be used simultaneously, it is desirable to balance their use. The instructions display indicates an underutilized floating point unit. When this is combined with the large

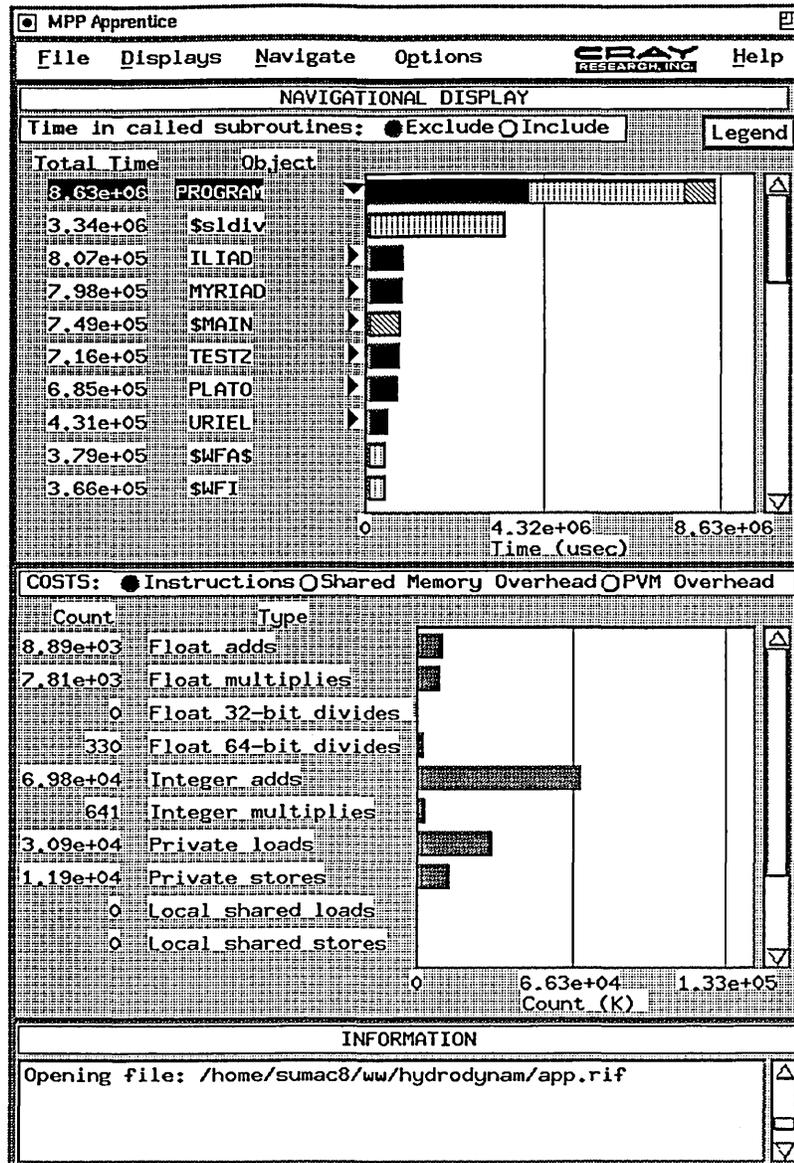


Figure 5: Main window of a hydrodynamics code

amount of time spent in \$sldiv, a strong case could be made for converting some integers to floats.

6 Conclusion

Initial users have had a tremendous amount of success with the MPP Apprentice on substantial codes. The success of developers working on the chemical code has already been mentioned. Several developers working on an electromagnetics benchmark were surprised to find that the routines they had been working to optimize were the two running most efficiently, and their biggest bottleneck was elsewhere. Another user

was able to take his code from 29.1 MFLOPs to 491 MFLOPs in a short period of time by resolving problems with PVM communication and barriers that MPP Apprentice pointed out.

The method of data collection, and the choice of the data being collected, permits the MPP Apprentice to scale well to long-running programs on large numbers of processors. By presenting data from the perspective of the user's original source code, the user is able to quickly identify the location of performance problems. Detailed information on instructions, shared

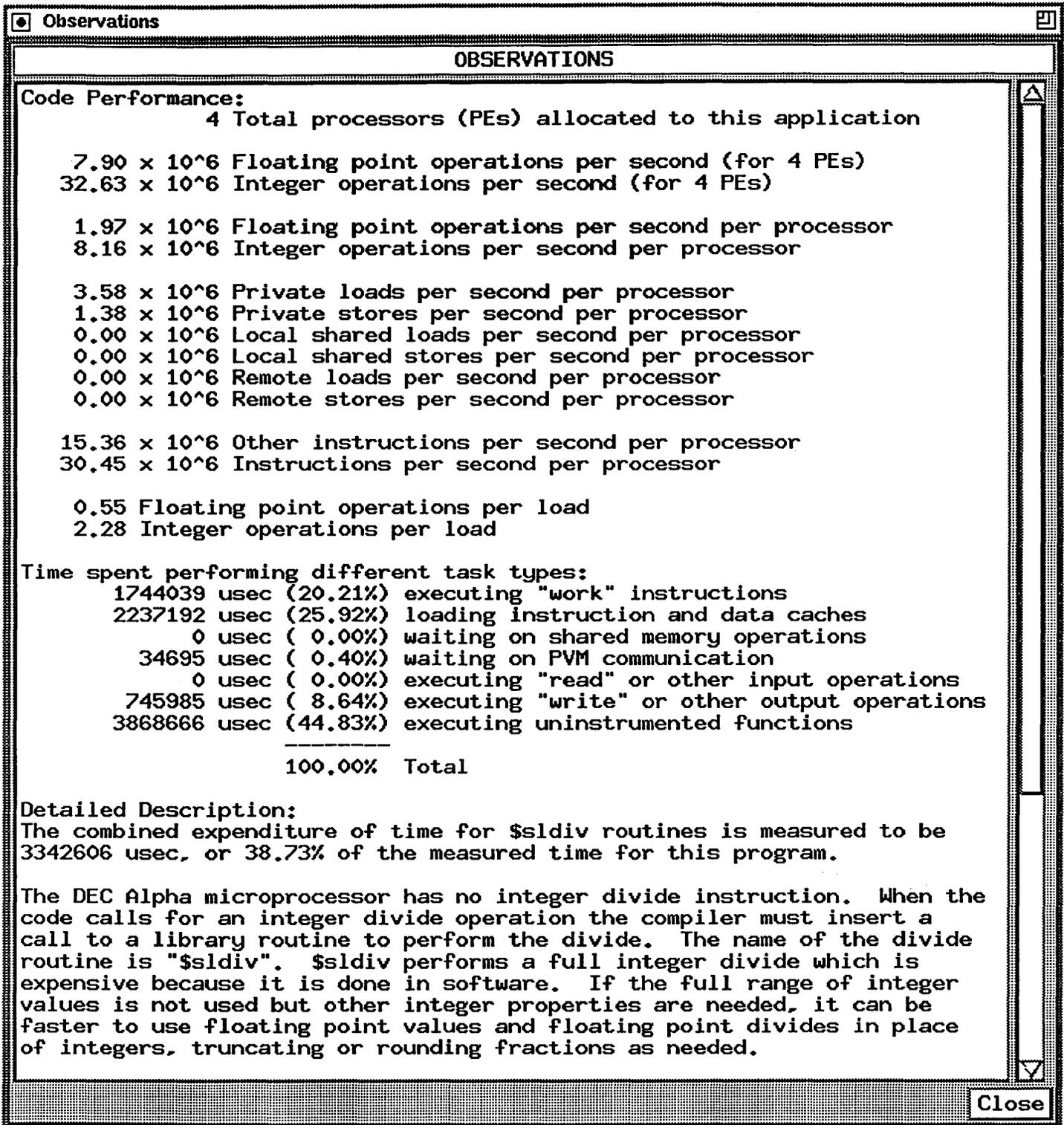


Figure 6: Observations window for the hydrodynamics code

memory overheads, and PVM communication allow a user to quickly identify the cause of performance problems. The knowledge base encapsulated in observations provides performance numbers and helps the user identify and resolve more difficult problems. As demonstrated above, the identification of problems in Cray T3D codes today can be achieved efficiently using the MPP Apprentice.

7 References

[CRI91] *Unicos Performance Utilities Reference Manual*, SR-2040 6.0, Cray Research, Inc., Eagan, Minnesota, 1991.

[CRI93-1] *Cray T3D System Architecture Overview*, HR-04033, Cray Research, Inc., Eagan, Minnesota, 1993.

[CRI93-2] *Introducing the MPP Apprentice Tool*, IN-2511 1.0, Cray Research, Inc., Eagan, Minnesota, 1993.

[He91] *Visualizing the Performance of Parallel Programs*, M. Heath and J. Etheridge. Software. IEEE Computer Society, Silver Spring, MD, September 1991, Volume 8, #5, pp. 28-39.

[Ma91] *Traceview: A Trace Visualization Tool*, A. Maloney, D. Hammerslag, and D. Jablonowski. Software. IEEE Computer Society, Silver Spring, MD, September 1991, Volume 8, #5, pp.19-28.

[Pa93] *MPP Fortran Programming Model*, Douglas M. Pase, Tom MacDonald, and Andrew Meltzer. CRAY Internal Report, February 1993. To appear in "Scientific Programming," John Wiley and Sons.

[Re93] *The Pablo Performance Analysis Environment*, D. Reed, R. Ayt, T. Madhystha, R. Noe, K. Shields, and B. Schwartz. Technical Report, University of Illinois at Urbana-Champaign, Department of Computer Science.

[Ya93] *Performance Tuning with AIMS -- An Automated Instrumentation and Monitoring System for Multicomputers*, Jerry C. Yan. HICSS 27, Hawaii, Jan 1994.

[Wi93] *ATExpert*, Winifred Williams and James Kohn. *The Journal of Parallel and Distributed Computing*, 18, Academic Press, June 1993, pp. 205-222.

[Wi94] *MPP Apprentice Performance Tool*, Winifred Williams, Timothy Hoel, and Douglas Pase. To appear in *Proceedings of IFIP*, April 1994.

Cray Distributed Programming Environment

Lisa Krause
Compiler Group

Cray Research, Inc.
Eagan, Minnesota

Abstract

This paper covers the first release of the Cray Distributed Programming Environment. This will be one of the first releases of Cray software that is targeted for a user's workstation and not a Cray system. The paper covers the goals and functionality provided by this initial release and shows how the user's environment for doing code development will move to his workstation. This paper also briefly describes plans for upcoming releases of this distributed environment.

Introduction

In response to customer requests, Cray Research, Inc. will provide the interactive portion of the Cray Programming Environments on the user's desktop system. This first release of the Cray Distributed Programming Environment (DPE) 1.0 is scheduled to coincide with the Programming Environment 1.0.1 release in the third quarter of this year. The predominant goal for DPE 1.0 is to provide the Cray Programming Environment on the user's desktop. Although not all of the programming environment components can be moved to the desktop with this initial release, the focus for DPE 1.0 is on Fortran 90. By providing Cray's Fortran 90 on both the user's workstation and Cray system, we hope to enhance the user's ability to do code development on whatever system is currently available and useful to them.

Programming Environments

Currently, without having the Distributed Programming Environment on the user's desktop, all program development that is done to confirm code correctness and to optimize performance is done on their Cray parallel vector system. To be able to fully utilize the capabilities of the program browser and the performance analysis tools, an interactive session on the Cray system is needed. If no interactive sessions are available at the site, the user must try to perform all of his tasks through a batch interface. Although batch use is valuable for determining Cray system utilization for big jobs, using batch for program development can be slow and contain numerous delays.

The Cray Distributed Programming Environment will seek to alleviate the interruptions and

delays that can occur when doing code development through a batch environment or even a heavily loaded interactive Cray environment. With DPE 1.0, the user now has the performance tools, such as ATExpert, flowview, perfview, profview, procview, and jumpview residing directly on his workstation. The program browser, xbrowse, also is on the workstation and can utilize the Fortran 90 "front-end" of the compiler to examine, edit and generate compilation listing. Although not a full compiler, this "front-end" component parses the code and diagnoses syntactic and semantic errors. The CF90 "front-end" also produces a partial CIF file which can be used through xbrowse or cflist on the desktop. Users can communicate with their Cray system when they need to finish compilations, load their programs, and execute them to obtain performance results by using scripts that facilitate this communication. The communication mechanism used is ToolTalk which is provided in the CrayTools package. CrayTools is included with the CF90 Programming Environments and is bundled with the host platforms targeted for DPE, with the exception being SunOS.

The advantages provided with the addition of the Cray Distributed Programming Environment to a user's working environment are numerous. With DPE 1.0 the user can now utilize Cray's Fortran 90 compiler on both the workstation and their Cray system. Access to the interactive performance analysis tools that are contained in the Cray Programming Environments is easier, since these tools reside on the user's workstations. By having these components of the environment available for the user on their workstation, code development can move off of the Cray systems. This frees up the Cray system cpu cycles to be utilized for large jobs that cannot run

elsewhere. Using the program browser and the tools interactively on their workstation frees users from needing to have an interactive session on their Cray systems. This favorable environment for the tools also provides a consistent interactive response time when they are invoked.

The release plan for the Cray Distributed Programming Environment 1.0 is consistent with the release plans for the other Cray Programming Environments. DPE 1.0 is due to be released in the third quarter of 1994 and DPE 2.0 will be released in coordination with the Cray Programming Environment 2.0 releases. The goals for this first release are to provide the interactive tools on the desktop and to gather customer feedback on this distributed environment.

Functionality of DPE 1.0

The Cray Distributed Programming Environment will provide many features of the Cray Programming Environments on the user's desktop. With the first release of DPE, the user will be able to browse and generate a compilation information file (CIF) by using the program browser, `xbrowse`. Using the interface provided by `xbrowse`, or with the `cflist` tool, the user can examine this CIF to obtain information on syntax and semantic errors regarding their codes without having to go to the Cray system. When a full compilation of the code is needed, the user can initiate this Fortran 90 compilation for the Cray system from their workstation. Additional executable components and commands contained in the first release of DPE provide both a framework and a method for communicating what is to be accomplished on the workstation and on the Cray system. Besides obtaining binaries from a full Fortran 90 compilation, commands can also be used to generate an executable and move this to the Cray system to be run for performance information. Performance analysis can then be done on the desktop, especially the analysis that can be obtained utilizing the ATExpert performance tool. To understand and learn more about the performance tools and the Cray Fortran 90 compiler, the user will be able to reference on-line documentation using the CrayDoc reader.

For the first release of DPE, the Sun SPARC workstation platform was chosen as the host platform, as well as the CRS CS6400 platform. These workstations can be running either SunOS 4.1 or Solaris 2.3 or later to be compatible with DPE 1.0. For the SunOS workstations ToolTalk will need to be obtained, as this communications tool does not come bundled into the operating system. The FLEXlm license manager is needed to control access to the DPE

product and can be obtained through CraySoft. The Cray Distributed Programming Environment 1.0 will need to have the CF90 Programming Environment installed on a Cray Y-MP, Cray C90, or Cray EL system, running a UNICOS release level of 7.0.6, 7.C.3, or 8.0.

Future Plans

The second release of the Cray Distributed Programming Environment, which is planned for the first half of 1995, will focus on creating a goal-directed environment. Instead of simply providing tools needed to support code development and analysis, the second release will move to provide a process where the user can simply indicate what the final goal should be for the code being provided. If the goal is to "optimize" the code, this goal-directed environment, directed by a visual interface tool, will lead the user through the steps needed to provide optimization to the code. Tentative plans for DPE 2.0 will strive to support this goal-directed emphasis by providing cross-compilers, distributed debugging capabilities and a visual interface tool. The plans may also include expanding the target platforms to encompass future MPP ; systems. Analysis is ongoing to determine the value of increasing the host platforms that will support DPE 2.0 to workstation platforms such as RS6000 and SGI.

Summary

In summary, by responding to customer requests, Cray Research, Inc. will release the major components of the Cray Programming Environment to run on desktop systems. The first release will provide tools for performance and static analysis including the Fortran 90 front-end. The second release is currently planning to provide a goal-directed environment highlighted by cross-compilers and a distributed debugging capability.

DPE Cray Distributed Programming Environment

Cray User Group
San Diego, CA
March 1994

Uwe Krause
Compiler Performance Section Leader



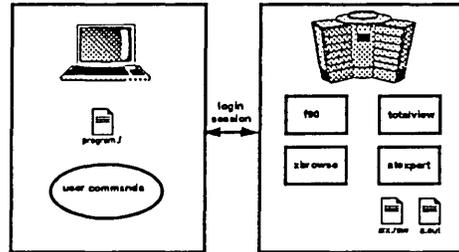
Provide the Cray Programming Environment on the User's Desktop



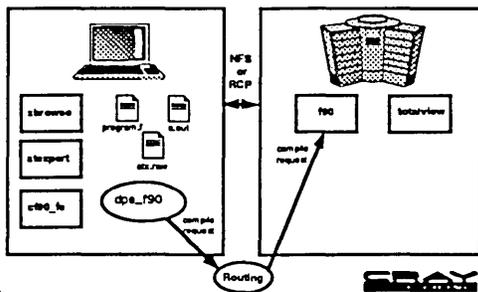
DPE 1.0 focus is on Fortran 90



Without DPE



DPE 1.0



Customer Advantage

- ◆ Cray's Fortran 90 available from desktop to Cray systems
- ◆ Easy access to the Cray Programming Environment
- ◆ Development moves off of Cray systems, allowing more Cray cycles for large jobs that cannot run elsewhere
- ◆ Users can take advantage of the Cray programming environment without an interactive session
- ◆ Consistent interactive response time when running the tools





Cray Distributed Programming

Release Plans

- ◆ Release 1.0 - 3Q94
- ◆ Release 2.0 - 1H95



Goals

Release 1.0

- ◆ Provide interactive tools on the desktop
- ◆ Gather customer feedback on distributed environment



Functionality

Release 1.0

- ◆ Browse and generate compilation information from the desktop
- ◆ Perform syntax checking and static analysis from the desktop
- ◆ Initiate the compilation of Fortran 90 programs on Cray parallel vector systems



Functionality

Release 1.0

- ◆ Generate an executable and analyze the performance from the desktop
- ◆ Execute ATEExpert from the desktop
- ◆ Reference online documentation from the desktop



Platforms

Release 1.0

- ◆ Host
 - Sun SPARC
 - Solaris 2.3 or later
 - Sun OS 4.1
 - CRS CS6400
 - Solaris 2.3 or later
- ◆ Target
 - Cray Y-MP, C90, EL
 - Future Cray PVP Systems



System Requirements

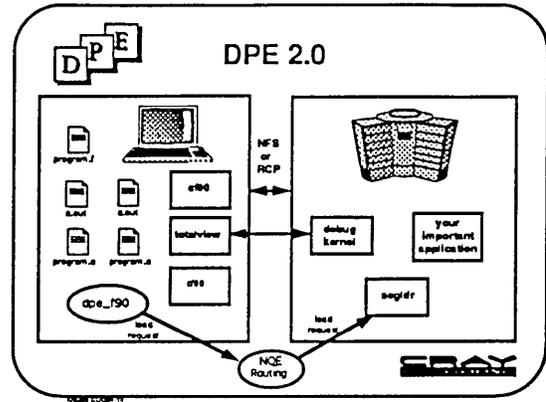
Release 1.0

- ◆ Host
 - Tooltalk for SunOS
 - FLEXIm
- ◆ Target
 - CF90 Programming Environment 1.0
 - UNICOS 7.0.6, 7.C.3, or 8.0



DPE

DPE 2.0 Focus is on Goal-Directed Environment

DPE **Goals**

Release 2.0

- ◆ cross-compilation
- ◆ distributed debugging
- ◆ visual interface tool
- ◆ target platform extends to MPP systems
- ◆ host platforms extended



DPE **Summary**

Responding to customer requests, Cray Research Inc. will release major components of the Cray Programming Environment to run on desktop systems.

- ◆ Release 1.0: tools for performance and static analysis, CF90 front-end
- ◆ Release 2.0: goal-directed environment, cross-compilers and distributed debugger



Cray TotalView™ Debugger

Dianna Crawford

Cray Research, Inc.
655-F Lone Oak Drive
Eagan, Minnesota 55121

ABSTRACT

Cray TotalView is a window-oriented multiprocessor symbolic debugger. First released in late 1993, Cray TotalView is a key component in the Cray Programming Environments. In 1994, it will be available on all current hardware platforms and will debug program written in Fortran 90, Fortran 77, C, C++ and assembler. This paper gives an overview of the features included in Cray TotalView, describes the releases plans and plans for the CDBX debugger which Cray TotalView will eventually replace.

1 Introduction

Cray Research has released a new debugger, Cray TotalView. This new debugger will span all current and new Cray supported hardware platforms and compiler languages and will eventually replace the Cray CDBX debugger.

Faced with the challenge of providing a debugger on Cray Research's massively parallel system, the Cray T3D™, we realized that it would be difficult to provide a useful product by porting CDBX to that environment. CDBX was designed to debug serial programs and the extensions to provide debugging of moderately parallel programs were of limited success. Also, CDBX was designed for a single architecture and was not very portable. We evaluated the currently available debuggers and selected the TotalView debugger from Bolt Beranek and Newman Inc. (BBN) as the starting point for the development of the new Cray parallel debugger. TotalView was selected because it was a powerful, easy to use debugger that had been designed to debug parallel programs, and it was portable enough to provide a good base for developing a debugger that would span multiple languages and hardware architectures.

Cray TotalView was released in the fourth quarter of 1993 and provided initial support for Cray T3D programs and Cray Fortran 90 programs running on the Cray Y-MP and C90 platforms. In 1994, Cray TotalView will support Fortran 90, FORTRAN 77, C, C++ and assembler programs running on Cray massively parallel, parallel vector and SPARC systems. This will

provide a common debugger across all Cray supported platforms, allowing programmers to leverage their learning investment when they program for more than one type of system.

Section 2 describes several of the features available with Cray TotalView. Section 3 describes the Cray TotalView release features and schedules. Section 4 describes the current plans for support of the CDBX debugger. Section 5 provides a summary of the paper.

(Note: To simplify the terminology, through the remainder of this paper the term TotalView is used to refer to the Cray TotalView debugger.)

2 Product Overview

TotalView is a window-oriented multiprocessor debugger. The TotalView debugger has an X Window System™, graphical interface that is easy to learn and displays as default the information most commonly needed when debugging. The window interface conveys the basic concepts of parallel codes, providing a separate window for each process the user is currently debugging. Execution control, including breakpoints and stepping through code, is available across all processes or for a single process. TotalView allows the user to work within their language of choice and supports expression evaluation for both Fortran and C. Viewing large, complex data arrays is supported through an array browser which displays a two-dimensional slice of a multidimensional array. TotalView functionality is described below using the screen images shown in Figures 1 through 5.

Figure 1 shows an example of the Root window. This is the main TotalView window and the first that the user sees when invoking TotalView. From this window, the user can start a debug session by loading an executable or a core file or attaching to a running process, access the on-line help, and view process information. In the example Root window, the display shows process information for each of the four processes that are part of a single program. One of the processes, `cray_transpose`, is stopped at a breakpoint, designated by the "B" after the process id. The user can debug any number of the processes by clicking the mouse on that process status line which will open a Process window for the selected process. TotalView also allows debugging of multiple programs from a single interface, which means, for example, that the user can debug both sides of a program distributed between a Cray Y-MP and T3D.

An example of the Process window is shown in Figure 2. There is one Process window for each process the user is actively debugging. From this window, the user can control processes, examine code and data and manipulate program data. Much of TotalView's functionality is available from this versatile and powerful window, only a small fraction of which is described here.

The source pane is at the bottom of the Process window. This displays the source code of the routine the user is currently stopped in. The user can change the display to show the assembler code or source code interleaved with the corresponding assembler code. Lines where breakpoints can be set are designated with a small hexagon to the left of the line number. The user clicks the mouse on the hexagon breakpoint symbol to set or remove a breakpoint, conditional breakpoint or expression evaluation. The user can run to a breakpoint or a specific line and can single step through the program,

including stepping across function calls, controlling all processes or just the process shown in the window. The text within the source pane is also active. Clicking the mouse on a variable name allows the user to display and change its value. Clicking the mouse on a function call displays the source for that function. The example in Figure 2 shows a display of the source code for the function `do_work`. The program is stopped at a breakpoint at line 278. An evaluation point is set at line 274 (Figure 4 shows the contents of this evaluation point) and another breakpoint is set at line 281.

The pane at the top right of the Process window is the current routine pane. It displays the calling parameters, local variable information, current program counter and access to the registers. The user can click the mouse on the information shown to display additional information or change values. For example, Figure 2 shows that the current routine, `do_work`, has one calling parameter, `prob_p`, which is a pointer to a structure. The user can click the mouse on (Structure) to view that structure's elements and values. Clicking the mouse on the value 1 of local variable `i` allows the user to change the value.

The top left pane of the Process window, called the sequence pane, displays the stack trace. The user can walk the stack by clicking the mouse on an entry point stack name which updates the source pane and current routine pane with information for the selected routine. In the example shown in Figure 2, the user is currently stopped in the routine `do_work`, which was called by `$TSKHEAD`. `$TSKHEAD` is the tasking library routine which initiates all slave tasks.

Figure 3 shows an example of the Action_point window. TotalView allows the user to save and load breakpoints across debug sessions. The Action_point window shows all of the breakpoints and evaluation points currently set within the program. From this win-

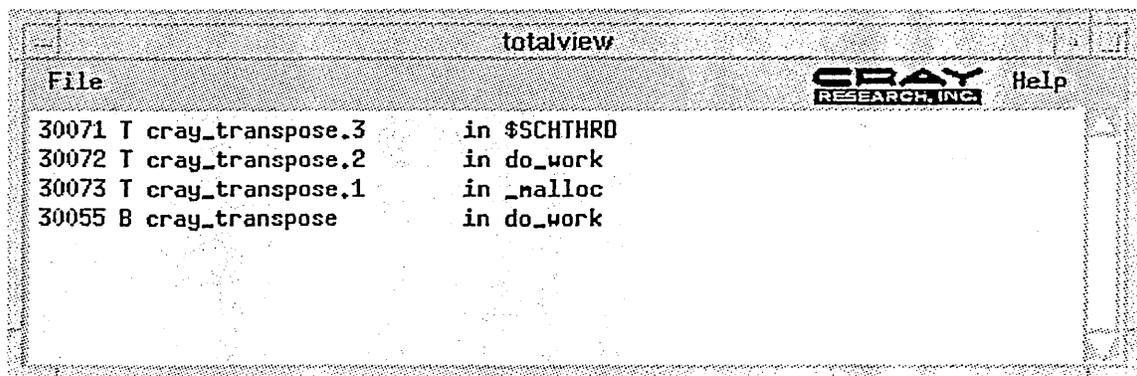


Figure 1: Root window

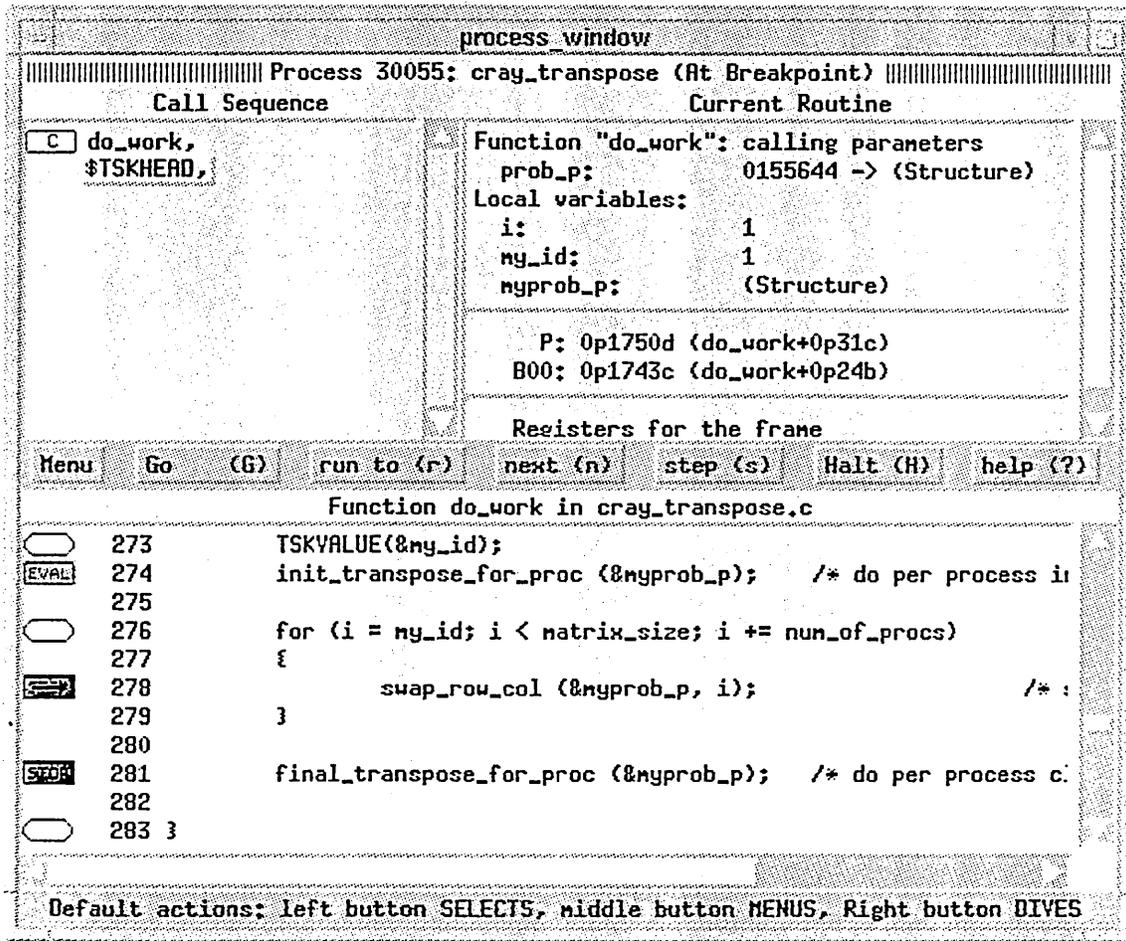


Figure 2: Process window

down the user can disable or delete breakpoints and evaluation points and can locate and display a breakpoint within the source.

The Breakpoint window is shown in Figure 4. From this window the user can set a conditional breakpoint or an expression evaluation point, using either C or Fortran syntax. These action points can be shared across all processes or a single process and can STOP all processes when the breakpoint is hit or only a single process. The example shows a conditional breakpoint (set at line 274 in Figure 2) which is shared across all processes and will STOP all processes when it is hit.

Figure 5 shows the array browser. The array browser displays a two dimensional slice of a multidimensional array. The user can specify the slice to display, the array location to display in the upper right hand corner, can scroll through the array, locate specific values and change values. The array browser has been optimized for large arrays and will only read in the portion of the

array being displayed plus a small buffer zone for scrolling, with additional portions of the array read in only as needed.

3 Release Plans

TotalView 0.1 is available today; it was released in the fourth quarter of 1993 with CrayTools 1.1. It provides full functionality for source-level debugging of parallel codes. All of the functionality described in section 2 is provided in this first release. TotalView 0.1 provides the initial support for Cray T3D programs and Cray Fortran 90 applications on Cray Y-MP and C90 systems.

The first general release of TotalView is release 1.0, scheduled for the second quarter of 1994 with CrayTools 1.2. TotalView 1.0 will support Cray Fortran 90, FORTRAN 77, C, C++ and assembler on Cray T3D, Y-MP, C90, EL and SPARC systems. With this release TotalView will support all current Cray languages and system architectures. Additional features include

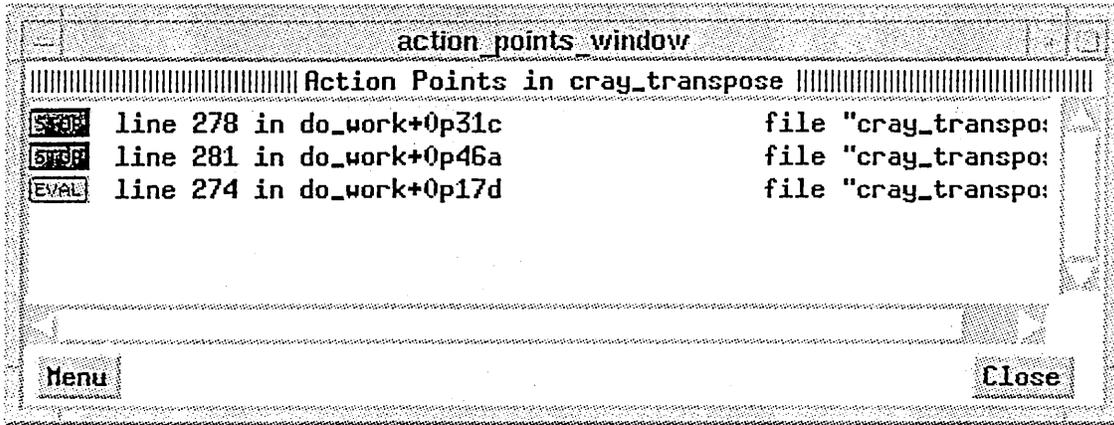


Figure 3: Action_point window

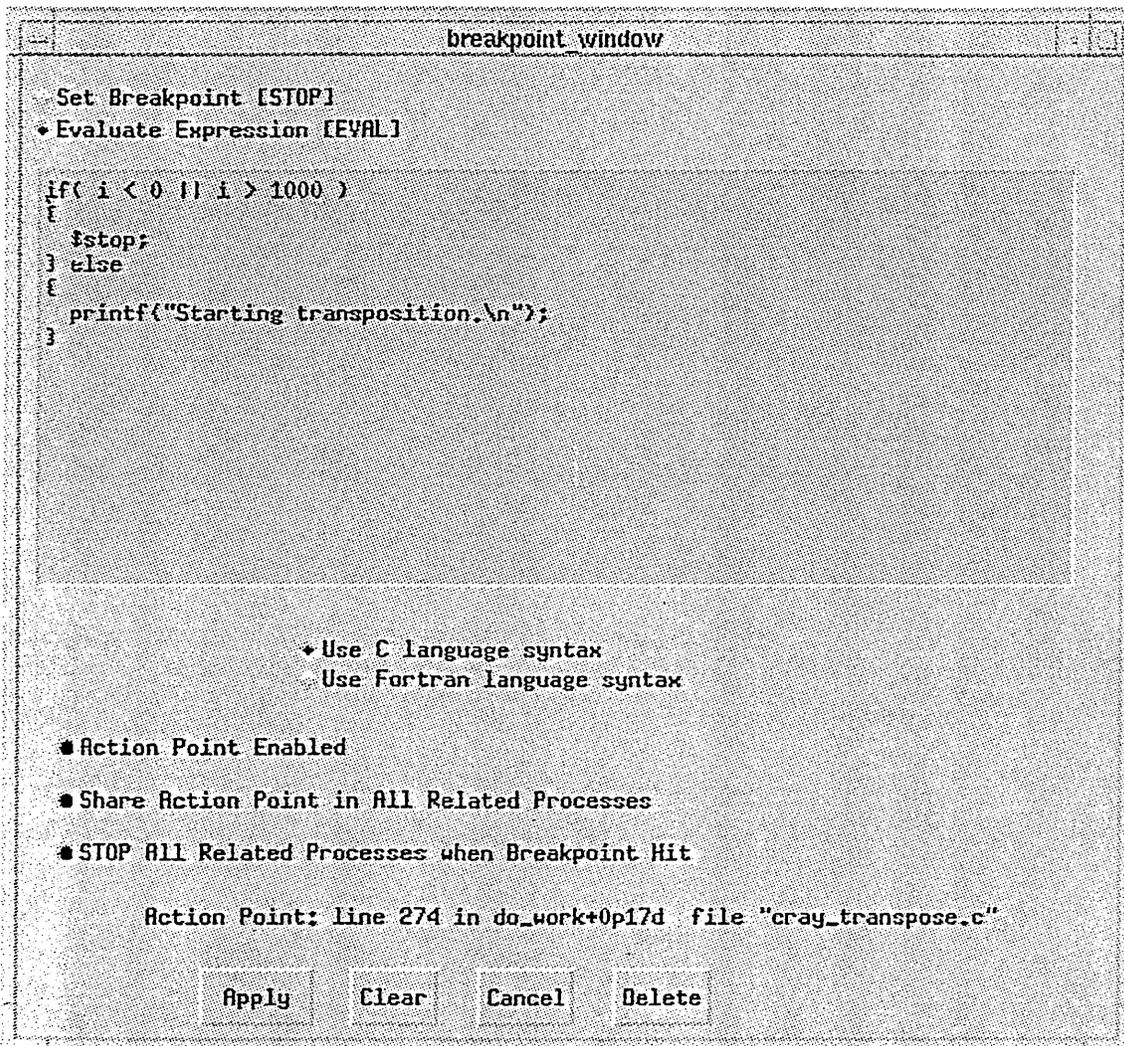


Figure 4: Breakpoint window

data matrix window												
big3												
int[30][50][10]												
[0][0][0]												
0	10	20	30	40	50	60	70	80	90	100	110	120
100	110	120	130	140	150	160	170	180	190	1100	1110	1120
200	210	220	230	240	250	260	270	280	290	2100	2110	2120
300	310	320	330	340	350	360	370	380	390	3100	3110	3120
400	410	420	430	440	450	460	470	480	490	4100	4110	4120
500	510	520	530	540	550	560	570	580	590	5100	5110	5120
600	610	620	630	640	650	660	670	680	690	6100	6110	6120
700	710	720	730	740	750	760	770	780	790	7100	7110	7120
800	810	820	830	840	850	860	870	880	890	8100	8110	8120
900	910	920	930	940	950	960	970	980	990	9100	9110	9120
1000	1010	1020	1030	1040	1050	1060	1070	1080	1090	10100	10110	10120
1100	1110	1120	1130	1140	1150	1160	1170	1180	1190	11100	11110	11120
1200	1210	1220	1230	1240	1250	1260	1270	1280	1290	12100	12110	12120
1300	1310	1320	1330	1340	1350	1360	1370	1380	1390	13100	13110	13120
1400	1410	1420	1430	1440	1450	1460	1470	1480	1490	14100	14110	14120

Dimension: [*][*][0] big3[1][1][0]

Figure 5: Array browser

CRAFT T3D programming model support and a batch interface for debugging core files. Significant effort has been made in developing reliable, portable code and a strong regression test suite in order to provide a stable, usable product across a broad platform base for this first general release of TotalView.

A minor release, TotalView 1.1, is scheduled for the fourth quarter of 1994 to provide two important features, data watchpoints and fast conditional breakpoints. This will be delivered in the CrayTools 1.3 release.

TotalView 2.0, scheduled for the middle of 1995 with CrayTools 2.0, will provide support for new Cray hardware. Release 2.0 will enhance the array browser to provide visualization of the array data and will provide enhancements to aid in debugging large numbers of processes. It will also provide distributed debugging for the Cray Distributed Programming Environment. This will allow the user to run TotalView on their desktop workstation to debug a program running on a Cray system.

(Note: CrayTools is a component of the Cray Programming Environment releases. Each Cray Programming Environment release provides a complete development environment including a compiling system, CrayTools development tools, and CrayLibs high performance libraries.)

4 CDBX Plans

CDBX will eventually be replaced by TotalView. The following outlines the current plans for CDBX support. CDBX 8.1 which was released in late 1993 is the last feature release of CDBX. The 8.1 release supports Fortran 90, FORTRAN 77, C, C++, Pascal and assembler on Cray X-MP, Cray-2, Cray Y-MP, C90 and EL systems. No new language or new hardware support is planned. CDBX does not support the Cray T3D or SPARC systems, for example, nor will it support Cray's new native C++ compiler planned for 1995. Minor changes will be made to CDBX to support compiler and UNICOS upgrades. CDBX will be supported with fixes through 1996.

5 Summary

Cray Research has released a new window-oriented multiprocessor debugger called Cray TotalView. Cray TotalView will support all current and new Cray languages and architectures, providing a common debugger across the entire range of products.

TotalView 0.1 is available today, providing full functionality for source-level debugging of parallel T3D and Fortran 90 codes. TotalView 1.0, scheduled for second quarter of 1994, will support Cray Fortran 90, FORTRAN 77, C, C++ and assembler on Cray T3D, Y-MP, C90, EL and SPARC systems. With this release, TotalView will support all current Cray languages and system architectures. Data watchpoints and fast conditional breakpoints will be supported in the TotalView 1.1 release scheduled for the fourth quarter of 1994. In mid-1995 we will deliver data visualization capabilities and a distributed debugger for debugging Cray programs from the desktop.

Fortran I/O Libraries on T3D

Suzanne LaCroix

Cray Research, Inc.
655-F Lone Oak Drive
Eagan, Minnesota 55121

ABSTRACT

The fundamentals of Fortran I/O on the CRAY T3D system will be covered. Topics include physical I/O environments, I/O paradigms, library functionality, performance considerations, and tuning with the I/O library.

1 Introduction

The CRAY T3D is a Massively Parallel Processor system with physically distributed, logically shared memory. The CRAY T3D is attached to, or "hosted by", a CRAY Y-MP or CRAY C90 computer system. This paper covers the fundamentals of Fortran I/O on the CRAY T3D. Section 2 provides background information about the CRAY T3D physical I/O environment, section 3 discusses T3D I/O paradigms, section 4 gives a high level description of library functionality available on the T3D, section 5 describes performance considerations and tuning opportunities, and section 6 provides a summary.

2 Background

Cray Research offers two physical I/O environments for T3D systems, Phase I and Phase II. The Phase I I/O environment is the default and is available today. In this environment, all disk data flows from an I/O Cluster through the host Y-MP or C90 system to the T3D. The host maintains one common set of filesystems so that all files available to a user on the host are also available to a user on the T3D.

The Phase II I/O environment is optional and is currently under development. In this environment, the optional second high speed channel (HISP) on an I/O Cluster may be connected to the T3D. This allows disk data to flow directly from the I/O Cluster to the T3D,

bypassing the host. However, just as in Phase I, the host maintains one common set of filesystems. Phase II augments Phase I by providing additional connectivity for T3D systems hosted by small Y-MP or C90 systems.

3 I/O Paradigms

Two I/O paradigms are offered for use in T3D applications, Private I/O and Global I/O. Private I/O is the default and is available today. With Private I/O, each process element (PE) opens and accesses a file independently. That is, each PE establishes its own file connection and maintains its own file descriptor. Another way to describe this paradigm is that the PEs behave like unrelated processes on a Y-MP. The user provides any necessary coordination of file access among the multiple PEs. This I/O paradigm is analogous to the message passing programming model. It is available to both C and Fortran programs.

The Global I/O paradigm is currently under development. Global I/O allows multiple PEs to share access to a single file in a cooperative fashion. Global I/O must be selected by the user by inserting a directive prior to the Fortran OPEN statement for which global I/O is desired. In this paradigm, all PEs open a file cooperatively. The file connection and the library buffer are global, shared resources. Each PE may access the file, or not, independently and the library coordinates the access. Another way to describe this paradigm is that

the PEs behave like the cooperative processes in a multitasked program on a Y-MP. Global I/O is available to Fortran programs following Cray Research MPP Fortran Programming Model rules.

4 Library Functionality

Cray Research supports a rich set of Fortran I/O library functionality on CRAY Y-MP and CRAY C90 systems. All of this functionality is available or planned to be available on the CRAY T3D.

Functionality that is available now includes support for formatted and unformatted I/O, sequential and direct access I/O, word addressable I/O, Asynchronous Queued I/O (AQIO), and Flexible File I/O (FFIO).

Features that will be available in the short term include support for BUFFER IN/BUFFER OUT, implicit floating point format conversion, and Global I/O.

Support for namelist I/O will be available in the longer term, in conjunction with support of the Fortran 90 compiler on T3D.

5 Performance Considerations

All I/O requests initiated on the T3D are serviced on the host. This results in a longer transaction latency for T3D users than for users on the host. Thus, T3D users should consider these three closely related I/O performance factors: optimum request size, latency hiding, and load balancing.

On the T3D, users should minimize the number of I/O system calls generated, and make a sufficiently large request to amortize the cost of the system call. The *assign* command has several options that allow a user to select a library buffer size and library buffering algorithm to help reduce the number of system calls. These options specify FFIO buffering layers, and require no changes to the application source code. Some of the more notable FFIO buffering layers are

- assign -F mr (memory resident file)
- assign -F bufa (asynchronous buffering)
- assign -F cache (memory cached I/O)
- assign -F cachea (asynchronous memory cached I/O)
- assign -F cos (asynchronous buffering, COS blocking)

Latency hiding is another important consideration. The FFIO layers which use an asynchronous buffering algorithm are effective at hiding latency. The user could also employ either AQIO or BUFFER IN/BUFFER OUT to overlap I/O with computation.

Load balancing refers to balancing the number of PEs doing I/O with the number of PEs doing computation.

The user must make sure that the compute PEs have data delivered at a reasonable rate to avoid idle time. With Global I/O, an application should scale up or down with little user effort to create a good load balance.

6 Summary

This paper has described the fundamentals of Fortran I/O on the CRAY T3D. The T3D I/O environment takes advantage of the filesystem management and I/O capabilities on the host. Users can mix and match the Private I/O and Global I/O paradigms to best fit their application. A powerful set of Fortran I/O library packages and tuning features is available on the T3D.

USER CONTACT MEASUREMENT TOOLS AND TECHNIQUES

Ted Spitzmiller
Los Alamos National Laboratory
Los Alamos, New Mexico

Overview

Determining the effectiveness of user services is often a subjective assessment based on vague perceptions and feelings. Although somewhat more objective, formal periodic surveys of the user community may serve more to harden stereotypical attitudes than to reveal areas where effective improvement is needed. Surveys tend to bring out perceptions of the most recent interaction or long standing horror stories, rather than long term performance. This paper examines how statistical data, surveys, and user performance can be used to determine areas which need improvement.

The *user services* function is often the focal point for several aspects of a computing operation:

- It provides information on how to use the various computing resources through personal interaction by phone or E-mail. (Documentation and training may also be a part of user services.)
- It provides feedback as to the efficiency of computing resources and the effectiveness of the documentation and training for these resources.

How well are we serving the user community? We occasionally ask ourselves this question, and management will sometimes wonder as well, should an irate user complain.

Perhaps equally important is the ability of user contact measurement tools to effectively assess trends in the use of computing resources. A sudden increase in queries relating to a specific resource can be an important indicator. Has the user community discovered the existence of a particular utility or feature? Is the documentation inadequate? Is training needed? These are questions that a simple set of tools can help answer.

This paper will limit the scope of user services analysis to that portion of which deals directly with the customer. This function may be referred to as the "Help Desk" or "Consulting".

Evaluation Methods and Criteria

Several criteria may be applied to evaluate the performance of user services:

- How well does the staff handle queries (effectiveness)?

- Are responses to queries timely and accurate (efficiency)?
- Are user concerns accurately and effectively fed back into the organization (closing the loop)?
- Are future user needs being anticipated?

Several methods may be used to evaluate these aspects of the user services operation.

- A formal survey
- An informal survey
- A statistical database
- The ambient complaint rate

All these methods have a place in the evaluation process and relying on any one method can lead to a distorted view of the effectiveness of the operation.

Any "survey" whether formal or informal involves varying degrees of subjectivity. Perhaps the greatest constraint in using a survey is that the responses have to be carefully weighed against the demographics of the respondents. Without understanding the context in which responses are submitted, evaluation can be virtually meaningless or, even worse, misinterpreted. Evaluating demographic profiles can be as time consuming a task as the survey itself.

Small, informal, and personalized surveys may elicit the most accurate response. These are typically one-on-one interviews in which the demographics can be determined as a function of the interview. The interviewer can determine the users knowledge level and degree of interaction with the computing facility and the user services operation. Probing questions based on previous responses may be asked. There are several caveats to this method.

- The interviewer must be skilled with interviewing techniques and must understand the customer service environment as well as the various computing resources available to the respondent.
- Only a limited number of users can be contacted in this manner thus, selection can be the critical factor. Two dozen "selected" users could make the computing facilities look "bad" or "good".

Formal "broadcast" surveys are often distributed to a large percentage of the user base and may be biased by human nature. If an individual is satisfied with the computing facilities they will often not bother to respond to the survey. Those who have an "axe to grind" will most always respond and be quite vocal.

Statistical data from some form of "call-tracking" system should be added to the subjective answers provided by a survey. In its most elementary form, a call tracking system is simply a log of all calls received by the help desk. The following information is considered minimal to provide useful analysis.

- The total number of calls handled per unit time (a business day is a good unit of measure).
- The type of resource about which information is needed. This can be a list of supported resources such as operating systems, network functions (E-mail, ftp, and etc.), mass storage, or graphics output facilities.
- The resolution time for the query.
- The need to refer the query to another person for additional expertise.
- The organization to which the person belongs.

Several systems are available to automate this process. These systems typically provide additional levels of sophistication to include report generation and statistical analysis.

Performance Metrics

Assuming that a valid survey can be constructed and administered, what are some of the key responses that will provide strong indicators of areas that may need improvement? Responses may be categorized in the following areas of user satisfaction:

- Are problems handled effectively by the consulting staff?
 - Are they polite, helpful, and friendly?
 - Do they exhibit knowledge of most resources?
 - Are problems diagnosed with minimal probing?
 - Are calls responsibly referred to more appropriate individuals?
 - Do they follow-up and effectively 'close' calls".
- - Are the computing facilities adequate?
 - Does the user need some capability that is not currently being provided?
 - Are the computing resources easy to use?

- Is the information provided adequate?
 - Are problems consistently resolved with the initial information provided?
 - Is a documentation reference given to the user for future needs?
 - Is documentation available which covers the information needed by the query?
- Is there an effective feedback mechanism to the development and engineering people to permit enhancements based on observed user performance?

This paper reviews some data retrieved from the Los Alamos National Laboratory user services team to illustrate the use of survey and statistical data as well as user performance.

Analysis of User Services

Example One

The following user responses are often the focal point for the argument to reorganize the operation around phone numbers of specialists instead of a single generic number.

- User "A" says "Consulting by phone has proven very ineffective - the consultant cannot easily visualize my problem, and not being extremely computer knowledgeable, I can't communicate the problem. Usually, I ask one of my colleagues for advice. I suggest setting up consultants to deal with specific levels of computer expertise."
- User "B" says "It is very difficult to get detailed consulting information over the phone. It would be nice to have a list of the consultants along with their areas of expertise so as to narrow the 'turkey-shoot' somewhat."

These appear to be valid perceptions and, based on comments like these, some organizations have tried alternative arrangements. But before making a decision to change, let's look at what the statistical data can add.

Consider that of 62 queries per day, 52% are concluded within 6 minutes and 85% with 12 minutes. These numbers indicate that most queries need relatively simple explanations. Note also that the typical consultant refers less than 23% of queries to others within the help desk team and less than 8% to "outside" expertise. This data doesn't support a "turkey-shoot" assessment.

Demographics of user "A": A Physicist who is sensitive about his lack of computer literacy. He feels uneasy about bothering a "high powered" consultant with his trivial questions.

Demographics of user "B": A sophisticated user whose queries require a higher level of expertise and thus almost every time he calls he gets referred to "another person."

Now let's add comments from users "C", "D", and "E". "Talking to John Jones [consultant] is like going to the dentist- I'd rather not"; "John Jones is arrogant and not user friendly", "Some [consultants] are extremely responsive, courteous, and knowledgeable, while others are terrible."

Action item: Clearly there is a problem. John Jones a very knowledgeable consultant but did not relate well to the user community, especially the neophyte. He was eventually moved to another area.

Having people with the right mix of technical and social skills is an important aspect of user services.

Example Two

User "F" reports that "There is sometimes a long wait for consulting services".

User "G" reports "The consulting phone is always busy".

The statistical data indicates that these users are frequent callers (F had 22 queries and G had 14 queries in a two month period).

Action Item: Install voice mail on the consulting phone numbers to handle the overload and avoid the "busy" signal.

Example Three

The call-tracking log (or database) can be used to determine specific activity areas. Knowing which resources are requiring significant assistance can assist in pinpointing poorly designed resources or the need for improved documentation or training. The following table represents the percent of various major resource areas for a recent two month period:

Resource	Percent of Queries
UNICOS	24
ALL-IN-1 Mail	18
Network	12
SMTP/UNIX Mail	8
Registration/Validation	7
PAGES/Graphics	7
Workstations	6
Other	6
Common File System	5
VMS	5
ADP/Admin	5
UNIX/ULTRIX	4

An analysis of the highest activity category, UNICOS, reveals the top 10 query areas for the past three years.

Query Area	Percent of UNICOS Queries		
	1991	1992	1993
PROD (job scheduler)	8	6	8
VI (editor)	4	9	6
Fortran	5	7	7
CLAMS (math software)	8	7	7
FRED (editor)	3	4	5
Login/logout	3	9	5
Environment (.cshrc etc.)	2	9	5
/usr/tmp	3	3	4
CFT/CFT77	5	7	4
FCL	5	3	3

Looking more closely at the UNICOS queries reveals that many users attempt to use UNICOS without any formal training and without referring to available documentation. This phenomena, which may be termed *pseudo-similarity*, is not unique to UNICOS. Most computer users who have had some experience on one system will attempt to use another operating system with the assumption that the similarities and intuitive nature of the beast will permit an acceptable degree of progress. Although there is an element of truth in this presumption, it causes the consulting effort to be loaded with elementary level questions.

An example of the effect of the pseudo-similarity syndrome can be seen by analyzing the queries related to the *vi* editor. Less than 10% of *vi* queries actually relate to the use of the *vi* editor itself. Typical symptoms of *vi* problems relate to the failure of *vi* to properly initialize as a result of one of the following:

- Not correctly defining the appropriate TERM value.
- Using a "non-default" window size (other than 80 X 35) without *resize*.
- Not establishing the *-tabs* parameter (*stty*).
- Using an incorrect version or option with the Los Alamos CONNECT utility.

Pseudo-similarity type problems are good indicators of poor design or implementation. In this case the user has four chances to fail. The real problem is most likely the UNICOS environment, which the user failed to properly establish in the *.login* or *.cshrc* files.

This example also points to the possible ambiguity of logging queries into a database. One consultant may log the problem as *vi*, while another may define it as *environment*. This lack of a "common view" can spawn widely divergent descriptors for similar problems, making analysis difficult.

The origins of the pseudo-similarity problem appears to lie with the traditional approach to preparing new users or introducing new resources into an established environment. The computer is a tool and, like any tool, the user wants it to be easy to use. They don't want to spend time learning how to use it. Historically computer documentation and training has been so poorly implemented that many users will simply apply what they do know and forge ahead.

When Los Alamos users were asked where they obtained most of their current knowledge, 53% cited co-workers and "trial and error". Only 39% said they would most likely refer to documentation for new information. Online "complete text" documentation was ranked *last* in importance by the user community.

Example Four

A review of the database shows that the category OTHER has shown a dramatic increase over the past six months.

Year	Total OTHER Queries	Charge Code Queries
1991	754	407
1992	958	560
1993 (first half)	755	620

Closer examination revealed that the majority of OTHER queries were for assistance in using or understanding the charge codes associated with computing. The increase was directly attributed to tight budgets which have caused many departments to look more closely at their computing expenses. The financial analysts lacked adequate training in the use of utilities to retrieve available data and, therefore required heavy phone consultations.

Action Item: A training class was provided for the FIN representatives in the use of the COST accounting program. A separate category for CHARGES was established in the tracking log to make analysis of this problem are easier

Summary

The most recent survey of Los Alamos computer users revealed that the Consulting Team ranked fourth out of 52 computing resources for overall satisfaction and tenth for importance. This is a strong indicator that personalized user services are needed and valued in a large and diverse environment. Consulting activity has likewise grown over the past four years from 40 to 62 queries per day.

Los Alamos users are relying more on the Consulting Team and less on documentation for their informational needs. This trend is troubling in that it is a strong indicator of:

- poor quality documentation available to users (specifically man pages) and
- poor user interface design.

With respect to improved user interface design, software development should examine the call-tracking database when making changes to an existing product (or when starting a new product) to review the problem areas of similar resources.

It is imperative that at least a modest effort be made periodically to evaluate the effectiveness of the computing resources and the user services activities in particular. Simple tools and techniques can make this task relatively easy and cost effective.

Balancing Services to Meet the Needs of a Diverse User Base

Kathryn Gates

Mississippi Center for Supercomputing Research

Oxford, MS

Abstract

The Mississippi Center for Supercomputing Research (MCSR) is in the unique position of providing high performance computing support to faculty and professional staff, as well as students, at the eight comprehensive institutions of higher learning in Mississippi. Addressing the needs of both students and veteran researchers calls for a varied approach to user education. MCSR utilizes on-line services and makes available technical guides and newsletters, seminars, and personal consultation. In spite of limited funding, MCSR maintains a high level of service through carefully considered policy designed to make the most efficient and proper use of computer and network resources.

Overview

Established in 1987, the Mississippi Center for Supercomputing Research (MCSR) provides high performance computing support to the faculty, professional staff, and students at the eight state-supported senior institutions of higher learning in Mississippi. The center originated with the gift of a Cyber 205 supercomputer system from Standard Oil to the University of Mississippi and is located on the UM campus in Oxford. A single processor Cray X-MP system was added in January 1991, and its capacity was doubled in April 1992 with the acquisition of a second processor. MCSR currently supports a Cray Y-MP8D/364 system, a front-end Sun workstation, an SGI Challenge L Series workstation and visualization equipment located at the eight IHL universities and the UM Medical Center.

MCSR was among the first sites to operate a Cray Research supercomputer system without permanent, on-site Cray Research support. The MCSR systems staff consists of two analysts and a student intern, and the user services staff consists of an administrative director, three consultants, and a student intern. Together, these groups evaluate user needs, plan operating system configurations, install and maintain operating systems and software applications, define user policies, and provide all other user support. Other groups provide operations, hardware, and network support.

A Diverse User Base

As with many supercomputing centers, the MCSR user base includes university faculty, professional staff, and graduate student researchers. The MCSR user base also includes undergraduates and a growing number of high school students and teachers. With major high performance computing centers at the John C. Stennis Space Center in Bay St. Louis and

the U.S. Army Corps of Engineers Waterways Experiment Station in Vicksburg, an important MCSR function is to aid in preparing students for careers utilizing supercomputing capabilities. At the close of Fiscal Year 1993, the MCSR user base included about 350 high school and undergraduate students (referred to as instructional users) and 150 graduate students, professional staff, and faculty members (referred to as researchers).

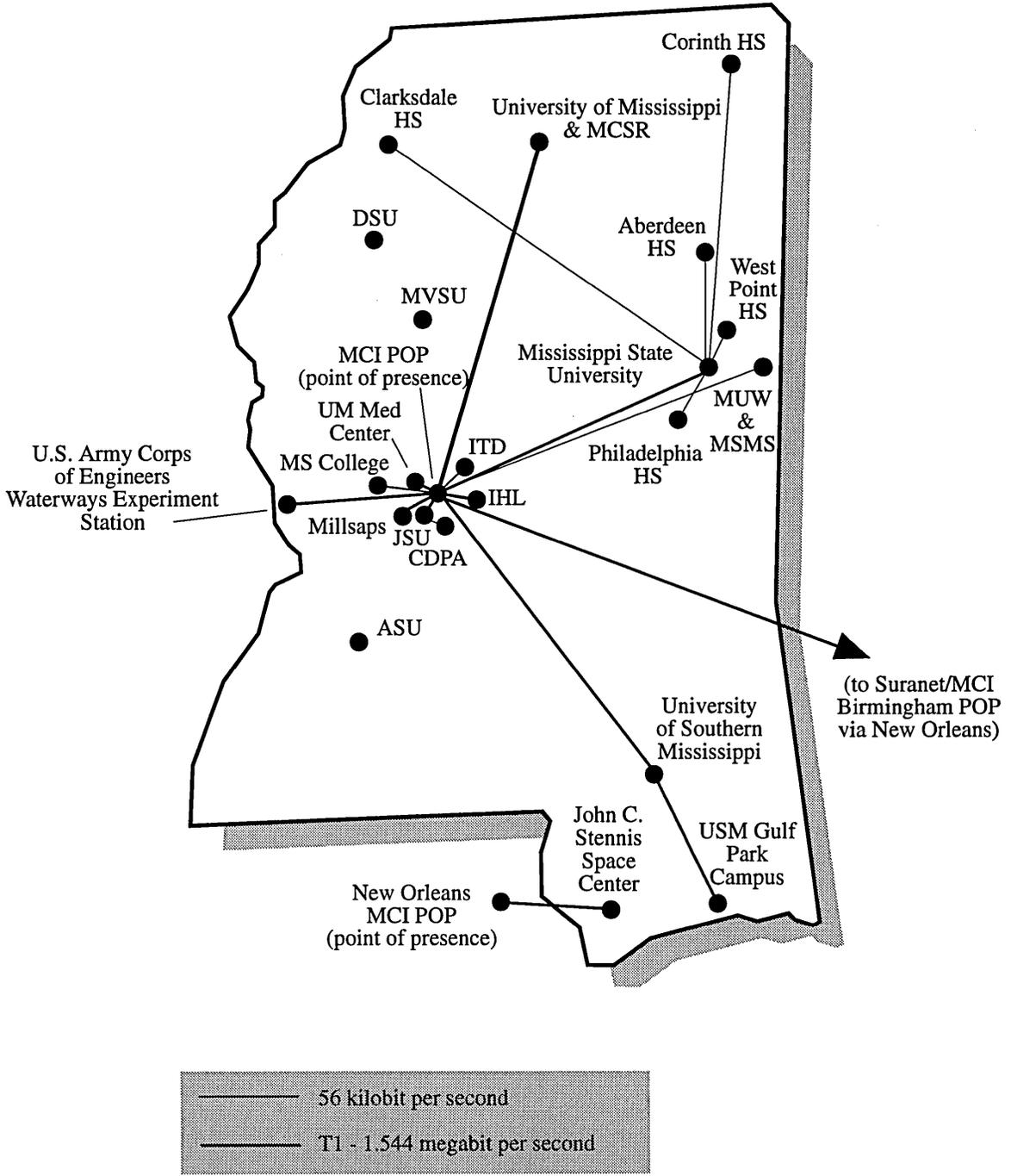
The experience level among users ranges from beginners who have some proficiency in using personal computers to researchers who are active on systems at national supercomputing centers. The jobs running on MCSR computing platforms represent many disciplines including computational chemistry, computational fluid dynamics, computer science, economics, electrical engineering, mechanical engineering, forestry, mathematics, ocean and atmospheric science, and physics.

A Geographically Dispersed User Base

Some users are located on the University of Mississippi campus and access MCSR machines through the campus network or through a local dial-in line, but the majority of users are located at other institutions in Mississippi. All but three of the universities supported by MCSR are connected to the Internet network through the Mississippi Higher Education Research Network (MISNET) as shown in Figure 1. The most active users are located at Jackson State University, Mississippi State University, the University of Mississippi, and the University of Southern Mississippi as shown in Figure 2.

The level of local network development varies from institution to institution. The Mississippi State University campus network is quite developed while several smaller universities only support one or two networked labs. A handful of Mississippi high schools have Internet access, typically through a

MISNET Topology



Note: Both conventional and/or SLIP dialup connections are available on most campuses allowing MISNET/Internet connections for authorized users.

Figure 1. The Mississippi Higher Education Research Network (MISNET)

SLIP connection to a nearby university. The lack of convenient network access is a primary hindrance for many MCSR users.

Problem Statement

Given these constraints - a user base that is diverse in terms of experience level and one that includes both students and veteran researchers (with the students greatly outnumbering the researchers), users with and without network access distributed across the state, and limited staffing and funding levels - what is the best means for providing a needed, valuable service that makes effective use of available resources? More specifically,

(1) What techniques can be used to manage this user base so that students are accommodated but not allowed to interfere with production work?

and

(2) What sorts of services can be reasonably provided to best meet the needs of the user base without overextending the staff or budget?

This paper describes the methods that have been used by MCSR staff to address these two issues.

Managing the User Base

Carefully considered policies designed to make the most efficient and proper use of computer and network resources are employed in managing the user base. First, Cray X-MP and Cray Y-MP computing resources are allocated so that research-

ers are favored over instructional users. This hierarchy of users is carried out through the application of operating system utilities. Second, expectations are clarified with the purpose of guiding users, particularly students, toward a successful path as they utilize MCSR systems.

Establishing a User Hierarchy

Without imposing some sort of hierarchy on accounts, instructional users would very quickly monopolize system resources due to their sheer numbers. Typically, these users need relatively small amounts of CPU time, memory, and disk space. Moreover, instructional accounts exist for brief, well-defined periods, normally until a semester closes or until the user graduates. A hierarchy is imposed by defining four categories of users:

Funded Research - Faculty, staff, graduate students conducting research associated with one or more funded grants.

Pending Funded Research - Faculty, staff, graduate students conducting research associated with one or more pending grants.

Research - Graduate students doing thesis or dissertation work, faculty and staff conducting open-ended research.

Instructional - High school students, undergraduates, class accounts, faculty and staff who are using MCSR systems for instructional rather than research purposes.

Users are, by default, assigned to the Instructional category. Faculty and staff members apply annually to be placed in the Research, Pending Funded Research, and Funded Research categories.

Implementing the Hierarchy through UNICOS Tools

Category assignment affects users in several ways including through disk space quotas, total CPU time allocated, and relative "share" of the system, with the most stringent limits being placed on instructional users. A priority system is established by employing various UNICOS tools including the Fair Share Scheduler, the Network Queuing System (NQS), user database (udb) limits, and disk space quotas.

Resource groups corresponding to the four categories of users are defined through the Fair Share Scheduler. Figure 3 shows how CPU time is distributed among the resource groups on the Cray Y-MP system. The Fair Share Scheduler has been effective in controlling how CPU time is allocated among users; it is the primary means for insuring that instructional users do not disrupt production work.

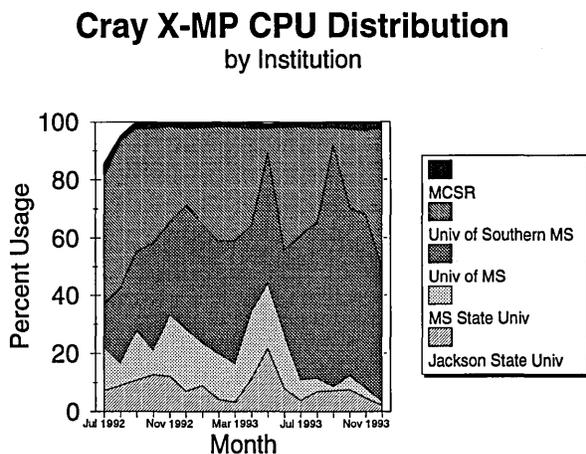


Figure 2. Cray X-MP CPU distribution by institution: July 1992 - December 1993

Resource Group Allocation Cray Y-MP

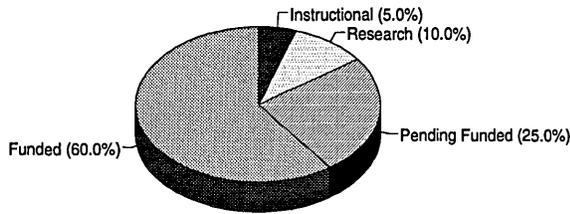


Figure 3. Cray Y-MP Fair Share Resource Groups

A separate NQS queue for instructional users is another means for insuring that these users do not disrupt production work. Without separate queues, instructional jobs can prevent research related jobs from moving to the running state within a queue, frustrating any attempts by the Fair Share Scheduler to grant preference to research related jobs.

User data base limits play a role in managing accounts. Instructional users are limited to five hours of Cray Y-MP CPU time for the lifetime of the account. This limit is enforced through the `udb cpuquota` field. The interactive process CPU, memory, and file limits are applied to all users to encourage the execution of jobs under NQS. Each semester, as students in C programming classes learn about *fork* and *exec* system calls, the interactive and batch job process limits perform a particularly useful function in controlling "runaway" programs. UNICOS disk quotas are enforced on all user accounts to conserve much sought-after disk space.

The Fair Share Scheduler, NQS, `udb` limits, and disk quotas permit control over accounts that is typically not available on UNIX-based systems. Once configured, these tools provide an orderly, automated method for directing resource allocation, with reasonable administrative overhead. Instructional accounts are easily accommodated and, at the same time, are prevented from causing a degradation in system performance.

Guiding Student Users Toward Appropriate Activities

With each passing semester, improper and unauthorized student use has become more of a problem. Unacceptable activities have ranged from students "snooping" in professors' accounts (the students, in some cases, are far more adept at maneuvering through UNIX than are their professors) to breaking into systems - fortunately, not MCSR Cray Research systems - by exploiting obscure operating system holes. Other activities have included those that are not particularly mali-

cious but certainly unsuitable for a research facility where computing resources are limited and very much in demand. An example might be students playing computer games across the network, using network bandwidth, valuable CPU time and memory.

The cost has been highest in terms of the staff time and effort required to identify unacceptable activities and to pursue the responsible persons. Moreover, this situation conflicts with the original purpose for giving students access to supercomputing systems. Only a few students have caused significant problems, yet many students seem to lack an appreciation for the seriousness of computer crime and an awareness of computer ethics.

An "Appropriate Use Policy" was introduced to address these problems. The policy explicitly defines appropriate and inappropriate use of MCSR computing systems. Users read and sign a copy of the policy when they apply for an account on MCSR computer systems. Individuals found to be in violation of the policy lose access to MCSR systems. Portions of the Mississippi Code regarding computer crime are printed periodically in the MCSR technical newsletter. Greater emphasis has been placed on UNIX file and directory permissions and on guidelines for choosing secure passwords in MCSR documentation. Password aging is used on all MCSR systems, requiring users to change their passwords every three months.

Since taking these steps, there have been fewer incidences of students misusing MCSR computing facilities. In the future, it may be necessary to utilize part or all of the UNICOS Multilevel Security system (MLS) to create a more secure environment for researchers. For now, these strategies - controlling resource allocation and educating users about computer ethics and account security - are sufficient.

Understanding User Needs

The Mississippi Supercomputer Users Advisory Group provides MCSR with program and policy advice to ensure a productive research environment for all state users. The advisory group consists of faculty and staff representing each university served by MCSR. MCSR staff members work with representatives to develop policies governing the allocation of supercomputer resources and to define and prioritize user needs. This relationship is an important means for understanding user needs and evaluating existing services.

Serving Users

A varied approach to user education is required to adequately address the needs of all users. Given that individuals learn in different ways, it is advantageous to offer information in mul-

multiple formats. Given the disparity in experience levels among users, it is necessary to provide assistance covering a wide range of topics. MCSR offers many services to its users (summarized in Table 1), including on-line aids, technical guides and newsletters, seminars, and personal consultation. The services offered by MCSR complement on-line vendor documentation (e.g., UNICOS docview facility) and tutorials.

Publications

Manuals and technical newsletters are fundamental to educating users in making best use of available resources. MCSR offers a user's guide for each major system or category of systems that it supports. These guides provide the site-specific information needed to work on the system. They highlight operating system features and tools, provide many examples, and direct the user to more detailed and comprehensive on-line documentation.

The MCSR document, *The Rookie's Guide to UNIX*, targets the subset of users who have little or no prior experience with the UNIX operating system. It gives beginning level UNIX training in a tutorial style. Topics covered include logging in, getting help, file system basics, the vi editor, basic commands, communicating with other users, networking tools, and shell programming. This guide is essential due to the large number of novice users.

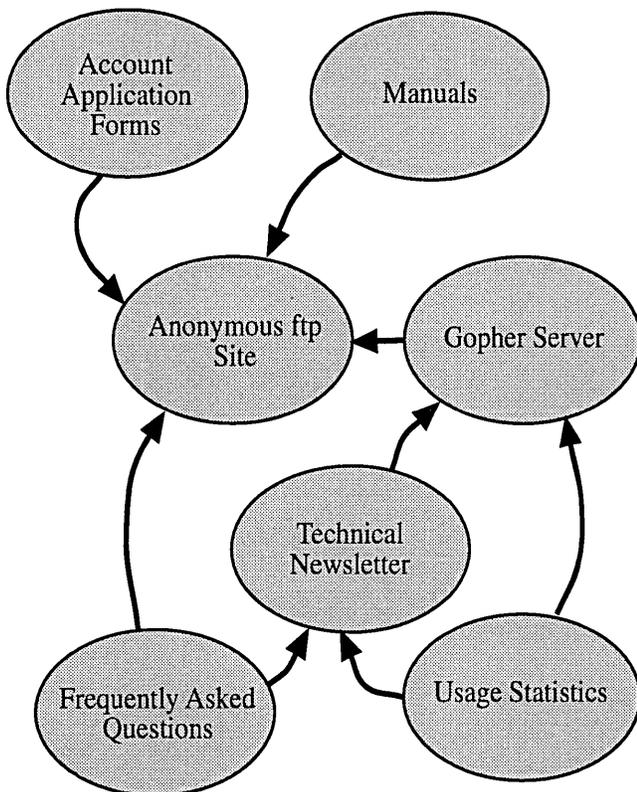


Figure 4. Information flow and reuse

1. Welcome to the MCSR Gopher Service ...
2. Frequently Asked Questions at MCSR.
3. General Information about MCSR/
4. How do I get an account?/
5. Issues of the Super Bullet/
6. MCSR Staff.
7. Mississippi Supercomputer Users Advisory Group Info/
8. News, Seminars, & Announcements/
9. Obtaining User Documentation/
10. Statewide Information Servers (MISNet)/
11. Worldwide Information Servers (Internet)/

Figure 5. Top level menu for the MCSR gopher server

A separate guide, *MCSR Services*, gives an overview of the services provided by MCSR. It reviews the MCSR computing and network facilities, lists contacts, describes available publications and programs, and provides account application information. The MCSR technical newsletter, *The Super Bullet*, provides timely supercomputing news and tips, features noteworthy operating system utilities, gives answers to frequently asked questions, and reports usage statistics.

On-line Services

On-line services play an increasingly important role in educating users. Through on-line services, information is available immediately, and users may retrieve it at their convenience. The active role shifts from the consultant to the user, freeing the consultant to handle more complex questions. On-line services work well in the MCSR environment, because they are universally available to all who have access to the computer systems, and they aid in extending a limited consulting staff.

MCSR supports an anonymous ftp site that contains ASCII and postscript versions of MCSR manuals and account application forms, example files, and a frequently asked questions (FAQ) file. This year, MCSR started a gopher server. The server provides much of the information that is available through *MCSR Services*, as well as other useful items such as advisory group meeting minutes, access to on-line manuals, and staff listings. Documentation efforts are maximized by making text available in multiple formats as shown in Figure 4.

When possible, MCSR employs on-line vendor tutorials such as the IMSL "Interactive Documentation Facility" and the Cray Research on-line course, "Introduction to UNICOS for Application Programmers."

	Service	Description	Audience	Availability	Advantages	Disadvantages
MCSR Publications	<i>Rookie's Guide to UNIX</i>	<i>Manual: Hard copy & On-line, Tutorial for beginning UNIX users</i>	<i>MCSR users who are new to UNIX</i>	<i>Hard copy: by request,</i>	<i>Hard copy: contains text and graphics images for an enhanced presentation, some users prefer to read hard copy as opposed to on-screen guides</i>	<i>Hard copy: Costly to produce and distribute, difficult to keep up-to-date</i>
	<i>User's Guide to the MCSR [Cray Research, SGI Challenge L, Vislab] System(s)</i>	<i>Manual: Hard copy & On-line, Provides site-specific information for the specified system, highlights operating system features and tools, lists applications, offers examples</i>	<i>MCSR users who have accounts on the specified system</i>	<i>On-line (through anonymous ftp and gopher): to all users who have network or dial-in access to MCSR systems</i>	<i>On-line: Immediately available</i>	<i>On-line: Users can peruse text only while in line mode, unavailable when host machine is down</i>
	<i>MCSR Services</i>	<i>Manual: Hard copy & On-line, Provides overview of computing platforms, services offered by MCSR</i>	<i>Current and potential MCSR users</i>			
	<i>Super Bullet</i>	<i>Technical Newsletter: Hard copy & On-line, Provides timely news and tips</i>	<i>All MCSR users</i>			
MCSR Seminars	<i>Application Programming on the MCSR [Cray Research, SGI Challenge L, Vislab] System(s)</i>	<i>Seminar: Provides information about developing code, running programs, accessing applications on the specified system</i>	<i>MCSR users who have accounts on the specified system</i>	<i>To UM users, To other sites by request</i>	<i>Personal, interactive, can easily customize to target specific needs</i>	<i>Reaches only a small subset of users, costly to send instructor to remote site, poses scheduling problems</i>
	<i>Beginning UNIX</i>	<i>Seminar: Assists beginning-level UNIX users</i>	<i>MCSR users who are new to UNIX</i>			
	<i>Intermediate UNIX</i>	<i>Seminar: Assists intermediate-level UNIX users</i>	<i>MCSR users who have mastered beginning UNIX topics</i>			
	<i>Cray FORTRAN Optimization Techniques</i>	<i>Seminar: Provides information about Cray FORTRAN extensions and optimization techniques</i>	<i>FORTRAN programmers working on Cray Research systems</i>			
	<i>Using the [Abaqus, Patran, MPGS, ...] Application</i>	<i>Seminar: Provides information about using the specified application</i>	<i>MCSR users who are working with the specified application</i>			
Consultation	<i>In Person</i>	<i>Consultation: Provides "face-to-face" assistance covering a range of topics</i>	<i>All MCSR users</i>	<i>To users on the UM campus</i>	<i>Fewest communication barriers, can focus directly on the problem</i>	<i>Unavailable to off-campus users, Restricted to MCSR office hours</i>
	<i>By Phone</i>	<i>Consultation: Provides assistance over the telephone covering a range of topics</i>	<i>All MCSR users</i>	<i>To all MCSR users</i>	<i>Can provide personal attention, focus directly on the problem</i>	<i>Restricted to MCSR office hours, some communication barriers</i>
	<i>Electronic Mail (assist)</i>	<i>Consultation: Provides assistance through electronic mail covering a range of topics</i>	<i>All MCSR users</i>	<i>To all MCSR users who have network or dial-in access</i>	<i>Can focus directly on the problem, include session transcripts, offer after-hour support</i>	<i>Response is "asynchronous" - may be slightly delayed</i>
Other	<i>MCSR gopher Server</i>	<i>On-line, menu driven tool for distributing information, including overview of computing facilities & support, account application procedures, and technical manuals</i>	<i>Current and potential MCSR users</i>	<i>To all MCSR users who have network or dial-in access</i>	<i>Content modifications are easily handled, immediately available</i>	<i>Unavailable when the host machine is down</i>
	<i>"Frequently Asked Questions" File</i>	<i>(On-line) file containing answers to frequently asked questions, available through gopher and anonymous ftp</i>	<i>All MCSR users</i>	<i>To all MCSR users who have network or dial-in access</i>	<i>Answers basic questions, immediately available</i>	<i>Unavailable when the host machine is down</i>

Table 1. Summary of Services offered by MCSR

Interactive Assistance

Personal consultation (in person and by phone) is provided by the three MCSR consultants, each focusing on a particular area. Often, the questions addressed in one-on-one sessions are beyond the beginning level. Faculty, staff, and graduate students are targeted with this service; however, individual assistance is occasionally required for users at all levels.

An on-line consulting service, *assist*, is available to handle all types of inquiries. Users send an electronic mail message to *assist* on any of the MCSR platforms, and the message is forwarded to a consultant on call after a copy is saved in a database. If necessary, the consultant reassigns the request to another staff member who is better equipped to handle it. Questions submitted after normal working hours or during holidays are forwarded to an operator who decides if the question needs immediate attention. Both instructional users and researchers make frequent use of *assist*.

MCSR offers a full set of technical seminars covering beginning, intermediate, and advanced topics. The seminar series is repeated each semester on the UM campus and by request at remote sites. Because seminars target only a small subset of users, they are viewed as secondary to other means for disseminating information.

Other Special Programs

Frequently, MCSR is host to regional high school groups for special sessions on high performance computing. Depending on the audience, some sessions are introductory in nature, while others are more technical. Students from the Mississippi

School for Mathematics and Science visit annually for a day-long program on supercomputing. In the following weeks, they use MCSR facilities from their school to complete research projects. Often these users have little experience with computers of any kind before working on MCSR systems.

Outlook

Lack of convenient network access remains a primary hindrance to using MCSR facilities for several smaller institutions and most high schools. MCSR consultants will continue to educate users at such sites concerning the benefits of Internet access. An auxiliary role will be to facilitate the expansion of MISNET through additional sites, by making users aware of potential grants.

MCSR publications will remain at the same level with respect to content and number of documents offered; however, efforts will be made to continue to enhance the presentation and organization of publications. The seminar offering will remain at the same level, with the exception of a new series covering visualization techniques.

The most significant changes will occur in the area of network-based and on-line services. As funding permits, suitable on-line vendor tutorials will be utilized. Consultants will continue to expand the existing anonymous ftp site and gopher server. An upcoming project is to offer site-specific information through multimedia tools such as NCSA Mosaic. MCSR consultants will capitalize on network-based and on-line services to target the majority of users with the widest range of material.

APPLICATIONS OF MULTIMEDIA TECHNOLOGY FOR USER TECHNICAL SUPPORT

Jeffery A. Kuehn

National Center for Atmospheric Research
Scientific Computing Division
Boulder, Colorado

Abstract

Supercomputing center technical support groups currently face problems reaching a growing and geographically dispersed user population for purposes of training, providing assistance, and distributing time critical information on changes within the center. The purpose of this presentation is to examine the potential applications of several multimedia tools, including Internet multicasts and Mosaic, for technical support. The paper and talk will explore both the current and future uses for these new tools. The hardware and software requirements and costs associated with these projects will be addressed. In addition, the recent success of a NCAR pilot project will be discussed in detail.

0 INTRODUCTION

Budget cuts, rising support costs, growing and geographically dispersed user communities, and an influx of less sophisticated users are driving user support to a critical junction. In the face of this adversity, how will we continue to provide high-quality support to our user communities? Old methods of one-on-one assistance and hardcopy documentation have been supplemented with online documentation, but the questions still remain: Where does a new user start learning? Where do they find the documentation? Where is the answer to *this* problem to be found in the documentation? Classes can be taught but when the users are geographically dispersed, the travel costs are prohibitive. One answer to this dilemma may lie in the strategic application of the latest wave of multimedia tools. These tools, however, are not a panacea for all of the problems, and in fact pose several new issues unique to their realm. The Scientific Computing Division at the National Center for Atmospheric Research (NCAR) has examined several such tools, and this paper is an attempt to summarize our experience with the two which appear most promising for user support.

1 BACKGROUND

1.1 Internet Multicast

An Internet Multicast is not really a single tool, but rather a broadcast across the network by a combination of tools used together in concert: a slow-scan network video tool, a network audio tool, and a network session tool. The individual audio and video tools are interesting and can be used for a variety of applications. The network audio tool allows you to use your workstation like a speaker phone on a party line, carrying meetings, conversations, and Radio Free VAT (a network music program) between workstations on the network. The network video tool transmits and receives slow-scan video across the network allowing a user at one workstation to transmit images to users at other workstations. Common uses for the network video tool include transmitting a video image of yourself sitting in front of your workstation or transmitting an image of the view from your office window. The former image lets everyone on the network (including your supervisor) know that you are in fact using your workstation (presumably for something the company considers productive). While the later image is presumably presented as a service to those of us on the network who have been denied an

office with a window. It is, however, the combination of audio and video through a session tool from which the multicast draws its name, and it is this capability which is most interesting for application to user support.

The network session tool does not add to the capabilities of the network video and audio tools, but rather serves as a user interface for combining the video and audio tools into a single broadcast signal. Thus to receive a multicast, you would choose a session using the session tool, which would initiate an audio tool session and a video tool session, allowing you to hear the audio portion of the broadcast and see the video portion of the broadcast simultaneously. Transmitting a session works in a similar fashion. The session tool is used to select/setup a session and then start an audio and video tool which will send signals. A single session can carry multiple audio and video signals, allowing a user tapping into the session to pick and choose which and how many of the audio and video links to display or listen to independently. For instance, five users carrying on a session may decide that each one wishes to hear the audio from all of the others, but that only two of the others are transmitting a video image that is worth seeing.

These tools are available for Sun SPARC, DEC, and SGI architectures. The DEC and SGI kernels will support multicast by default, but the Sun kernel will require modification to the default kernel and a copy of the multicast routing daemon. The support personnel at NCAR strongly recommend that kernel updates and modifications be tracked carefully on all three platforms.

1.2 Mosaic

Mosaic is a tool developed at the National Center for Supercomputing Applications (NCSA) for browsing information on a network. The information can take one of several forms: hypertext documents, other documents, pictures, audio bites, and animation sequences. The hypertext documents are the central "glue" which link all of the information together. These documents contain coded embedded references to files existing somewhere on the worldwide network. The reference coding instructs Mosaic as to the method of retrieval and display of the selected file. Thus, a user usually starts out with Mosaic from a "home page" hypertext document which contains links to other documents (hypertext, ASCII, PostScript), image files, sound files, and animation files on the network. Currently, a great wealth of information is available, but it is organized mostly on a site by site

basis, and the author is unaware of any concerted effort to provide a more network-wide structure at the time of this writing.

2 NCAR PILOT PROJECT EXPERIENCES

2.1 Multicasting User Training Classes

NCAR's Scientific Computing Division has multicast two of its four user training classes to date: the Cray Fortran/C Optimization class, and the NCAR Graphics training class. In both cases, the motivation for attempting a multicast came from one of our user sites at which there were several individuals interested in attending the class in question, but because of tight budgets, funding for travel to Boulder could not be obtained. The decision to try the multicast was made as a compromise solution, allowing the students to view the class and have two-way audio communication with the instructor in Boulder, while sparing the travel budgets. Class materials were mailed to the remote students ahead of time, allowing them to follow the materials with the people in Boulder. The teaching materials provided in both cases were an integrated set of lecture notes and slides in a left-page/right-page format which provided the information from the overhead slide on one page and a textual discussion of the slide on the opposing page.

2.1.1 Software, Hardware, and Staff Requirements

There are several start-up costs associated with producing multicast training classes. The software required to produce a multicast is freely available via anonymous FTP (see **SOFTWARE** for information on how to obtain the software). The costs associated with this software are mainly those related to the staff time required to get it working correctly. Additionally, each workstation will require a kernel configured to handle multicasts, and the multicast routing daemon must be run on exactly one host on the local subnetwork segment.

The hardware costs are a different story. The largest single piece of equipment required is a low-end workstation dedicated to handling the transmission. Additionally, low-end workstations are required for the viewers. NCAR used a vanilla Sun SPARCstation IPX for one class and a SGI Indy for the other as the classroom transmission platform. The cost of such machines runs between \$5,000 and \$10,000 depending

on the configuration. The systems which will be transmitting video signals, require a video capture card which costs between \$500 and \$1,000 if not supplied with the workstation. A high quality camera (costing approximately \$1,000) is highly recommended for transmitting the images of the instructor and their A/V material. An inexpensive CCD video camera (approximately \$100 and often supplied with workstations today) can be used for the students' images. Additionally, microphones will be needed for both the instructor and the remote students. The inexpensive microphones commonly supplied with most workstations will suffice for the students, but it is recommended that the instructor use a high-quality wireless microphone (approximately \$250).

Finally, regarding staff time, we found that setting up and testing the multicast link required about four hours with at least one person on each end of the connection. During the class and in addition to the instructor, a camera person is required. This person handles the camera work for the instructor and must be competent in setting up and managing the link which may occasionally need to be restarted. Given this, there is little economic incentive to set up a multicast for only one or two remote students; however, other issues may outweigh the economic factors. This overhead could obviously be figured into any cost associated with the class for the remote students.

2.1.2 Results

Overall, both of NCAR's multicast classes were highly successful. The remote students claimed that the multicast was "almost as good as being there", while the local students reported that the multicast did not noticeably disrupt their learning process. The course evaluation ratings from both the local and the remote groups were comparable. Later interactions with local and remote students of the the optimization class suggested that both groups had acquired a similar grasp of the material. In all respects, the multicast worked well in producing a learning environment for remote students similar to that available to those in the local classroom.

All of this success did not come without a price. The multicast caused noticeable and sometimes maddening slow downs on the computing division's network. The available network bandwidth allowed transmissions of a clear audio signal and the video signal was able to send between one half frame per

second and five frames per second at the cost of a heavy impact on other network traffic. This imposed restrictions on both the instructor and the camera operator which required changes in presentation and camera technique.

The instructor needed to refrain from gesturing and shift their style more towards "posing". Specifically, because of the low frame rate, one could not move continuously, as this would have two effects: motions would appear disjointed and nonsensical and the network traffic would climb intolerably. Thus, a more effective technique would have the instructor point to material on the projection screen, and hold that "pointing pose" while describing the material. Pacing and rapid swapping of slides were two other areas which caused problems, and both need to be avoided. It turns out that it is actually rather difficult to alter such gestures made in an unconscious fashion, and that some instructors have much more difficulty making this adjustment than others. When the instructor breaks for questions, they need to be conscientious about repeating local questions so that they may be heard by the remote students. Also, as it may have been necessary to mute the remote microphones to reduce feedback on the audio link, the instructor should make a point of allocating time for the remote students to pose questions.

Similar adjustments were required of the camera operator. Specifically, slow panning and zooming of the camera came across very poorly on the slow scan video link and swamped the network in the process. Thus, the camera person was better off making a fast pan or zoom adjustment, and then leaving the camera stationary on the new subject. However, much of this depends on the instructor and cannot be controlled by the camera person. If possible, it is recommended that the instructor and camera operator collaborate on the presentation.

The last of the problems were related to the stability of the links. It was possible that either the audio (most likely) or the video link might fail at some point during the presentation or that the audio signal might break up badly enough that it was unintelligible. The SGI platform had more trouble keeping the link up and running because of software problems. The Sun platform had more difficulty with audio break-up because of the slower CPU. While neither platform was perfect, both served adequately.

2.2 Online Documentation with Mosaic

Since the release of NCSA's Mosaic, sites world wide have begun displaying their interest by developing the hypertext documents which form the links on which Mosaic's access to network information thrives. NCAR has developed a great deal of information regarding their mission and the part in that mission played by each of the NCAR divisions. Additionally, portions of the institutional plans for the future are available. The information can be referenced from any workstation (running X-Windows), PC (running Microsoft's Windows software), or Macintosh which is connected to the Internet and runs the Mosaic software.

2.2.1 Software, Hardware, and Staff Requirements

NCSA's license agreement for Mosaic is very generous and, as mentioned above, it will run on a wide variety of platforms provided that a network connection is available (see **SOFTWARE** for information on obtaining Mosaic). Thus, the primary costs involved in using Mosaic are for the support staff time. The Scientific Computing Division at NCAR has currently dedicated one full time person to developing Mosaic material. Additionally, others are involved in projects to move current documentation over to such online formats.

2.2.2 Results

This project is a work in progress, but the headway made over the last few months is very impressive. Already we have a great deal of material on NCAR in general, and the computing division has a very detailed description of the work done by each of its individual groups. The documents contain static images, movies, music, speeches, and many other bits of information to flesh out the work we do at NCAR. Other divisions are beginning to follow suit and produce material describing their own functions. Individuals are also "getting into the act", and the author himself has just recently learned Hyper Text Mark-up Language (HTML) and written his own home document with hypertext links to information on the network which he accesses frequently. Much of the information developed over the last few months has been of an advertising or overview nature to demonstrate the capabilities of Mosaic, but we are now beginning to move toward making more technical

information available. It should be noted, however, that as a site moves toward becoming a hub for Mosaic users, it would be wise to move the Mosaic information services onto their own machine to reduce the impact of network activity on internal functions.

3 FUTURE DIRECTIONS

3.1 Multicasts

There are several possible future uses for multicasts in user support at NCAR. Training via multicast has proven so successful that the author hopes to see it extended to the other classes taught by the computing division, and possibly to classes taught by other divisions within our organization. However, the possibilities do not end there.

Many advisory committees and panels for the computing division must operate without direct input from that portion of our user base which is not local to the Boulder area. Frequently, remote users must be represented by a member of our staff who has been designated as the advocate for a particular group. While this has worked well in the past, it may soon be possible to allow these groups a more direct method of representation via multicast.

The computing division's consulting office currently operates in a help-desk mode. Whoever is on duty answers all of the walk-in, telephone, and electronic mail questions coming into the consulting office. The others are handling either special appointments made by local users or catching up from their last turn on the duty shift. Multicasts offer two possibilities here. First, a multicast link could serve as an additional medium for handling user questions, over and above the standard walk-in, phone, and e-mail. Secondly, it could serve to extend the capabilities of special appointments to remote users.

3.2 Mosaic

The future of Mosaic at NCAR can be expected to cover the two possibilities in user documentation: that which is relatively static and that which is more dynamic in nature. Static documentation might include things such as documents describing site specific features and commands, descriptions of how to use a service, and frequently asked question lists. The dynamic documentation, however, will be much

more interesting and include items such as our daily bulletin and change control system.

The daily bulletin is published every morning before 9 A.M. It describes the current state of the computing facility and includes short term announcements of upcoming changes. In addition to our current scheme of a local command which will page the current bulletin on any terminal, it would be relatively straightforward to make the information available via Mosaic as well. Furthermore, our current text format could eventually be enhanced to include audio reports of status and changes, video clips of significant events (such as the installation of new machines), and graphic images relevant to the daily announcements.

The change control system at NCAR is currently set up via e-mail notices with a browser based on the day the message was sent. The system works, but Mosaic's capability for including links presents a better model for representing the interconnected nature of the computing facility and how changes in one area may affect another. The advantage to the current system is simplicity; a programmer making a change fills out a template describing the change and the effective date, then mails the completed template to the mailing list. To make any real improvement over this simplicity and take advantage of Mosaic's hypertext capabilities will require a system significantly more sophisticated than the one currently in place.

4 SUMMARY

The tools discussed in this paper provide a cost effective answer to many of the questions posed in the introduction. Though users are geographically dispersed, they usually have some level of network access. Thus, the multimedia tools based on the network can support their needs regardless of their location. Because multicasts can be used to provide training without the associated travel costs, budgets can be saved for other critical needs. Because Mosaic provides entry points into web-like information structures, the question of the starting point becomes less important. Users can start at a point covering their immediate needs and branch out to other areas of interest as time permits. Furthermore, both multicast and Mosaic avail themselves to the possibilities of stretching our current capabilities beyond our facilities by exploiting the conferencing capabilities of multicasts and the audio/video capabilities of Mosaic. However, it should be remembered that the tools are

heavily dependent on the accessibility of high bandwidth networks, though our infrastructure is quickly growing to fulfill this need. The final word becomes a question of support. What difference exists between supercomputing centers today other than differences in support?

5 SOFTWARE

1. SD: The network session directory tool was written by Van Jacobson at Lawrence Berkeley Laboratory. Binaries are available via anonymous FTP at ftp.ee.lbl.gov in the directory /conferencing.
2. VAT: The network audio conferencing tool was written by Van Jacobson and Steve McCanne at Lawrence Berkeley Laboratory. Binaries are available via anonymous FTP at ftp.ee.lbl.gov in the directory /conferencing.
3. NV: The network video tool was written by Ron Frederick at the Xerox Palo Alto Research Center. Source and binaries are available via anonymous FTP at parcfp.xerox.com in the directory /net-research.
4. MROUTED and SunOS 4.1.x mods: The multicast routing daemon and the kernel mods to SunOS was written by Steve Deering at Stanford University. Source and binaries are available via anonymous FTP at gregorio.stanford.edu in the directory /vmtp-ip.
5. MOSAIC: The Internet information browser was written by Eric Bina and Marc Andreessen (X-Windows), Chris Wilson and Jon Mittelhauser (Microsoft Windows), and Aleksandar Totic, Thomas Redman, Kim Stephenson, and Mike McCool (Macintosh) at NCSA/University of Illinois at Urbana-Champaign and is, in part, based on code developed by Tim Berners-Lee at CERN for CERN's World Wide Web project. Source and binaries are available via anonymous FTP at ftp.ncsa.uiuc.edu in the directory /Mosaic.

REFERENCES

1. Paul Hyder, NCAR/SCD, personal communications, May 1993 through March 1994.
2. Greg McArthur, NCAR/SCD, personal communications, August 1993 through March 1994.

INTEGRATED PERFORMANCE SUPPORT: CRAY RESEARCH'S ONLINE INFORMATION STRATEGY

Marjorie L. Kyriopoulos

Cray Research, Inc., Eagan, Minnesota, U.S.A.

"Both the technologies and the time are right. Service industries are on the rise, and performance support appropriately focuses on serving the customers."

Esther Dyson
Software Industry Expert

Abstract

Cray Research is planning to implement the following three critical components of an Integrated Performance Support System (IPSS) this year:

- Online documentation (CrayDoc)
- Computer-based training (CrayTutor)
- Online help and messages

These components provide users with immediate, on-the-job access to information, assistance, and learning opportunities. This paper describes the benefits of integrated performance support systems for Data Center staff, User Services staff, and end users. Current plans for delivering the IPSS components and progress toward their integration also are discussed in this paper.

Cray Research is interested in customer feedback on these user support products and in future directions required by our customers.

Introduction

Customers tell us that they want to be able to find the information they need quickly. And they want the information online. Most Cray Research customers like Docview, but many want their online documentation installed on a workstation instead of their Cray machine. They want to be able to navigate through the information quickly and

easily. In addition, many customers have indicated that they want our information tagged with Standard Generalized Markup Language (SGML) to make it easier for them to add their own local procedures.

Users need the flexibility of being able to learn anywhere at any time in order to meet the demands of today's work environment. Scheduled training classes are often offered at the wrong time or are too costly for customers who receive little or no training credit with the purchase of a Cray Research system or product.

Cray Research and, in particular, Software Information Services (SIS), has been soliciting customer feedback to help us develop our IPSS strategy. Although we have heard various concerns, some of the common issues are:

- The error message doesn't tell me what to do next.
- I can't find the right information.
- Schedules prevent me from sending my operators to class.
- Why do I need to know UNIX to run my application?

All of these concerns have one thing in common: our customers are not getting the information they need when they need it.

The Solution: Integrated Performance Support Systems

An Integrated Performance Support System (IPSS) integrates information, assistance, and learning opportunities, making them available to the user on demand in the workplace at the moment of need. Integration of the components is planned to occur along a continuum, from components such as online documentation, which are loosely integrated, to a fully integrated system that is transparent to the user.

The goal of an IPSS is to enable user performance while drastically reducing or eliminating the training, documentation, and support required for the product.

Benefits: Reduced Customer Costs and Information Overload

Reduced Costs

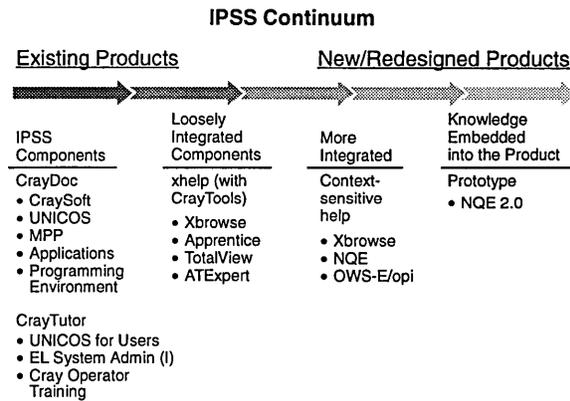
Our customers know that the cost of hardware and software includes the cost of user support and training. With an increasing number of tasks to perform and increasingly complex software products, users no longer have as much time to learn how the software works. They need products that require them to spend less time and money on training.

Reduced Information Overload

Cray Research systems currently require users to read volumes of information in paper manuals. Finding information in these volumes is time-consuming, difficult, and more complex than it needs to be for today's supercomputer user. Users need smaller units of information, provided in context with the current task, at the moment of need.

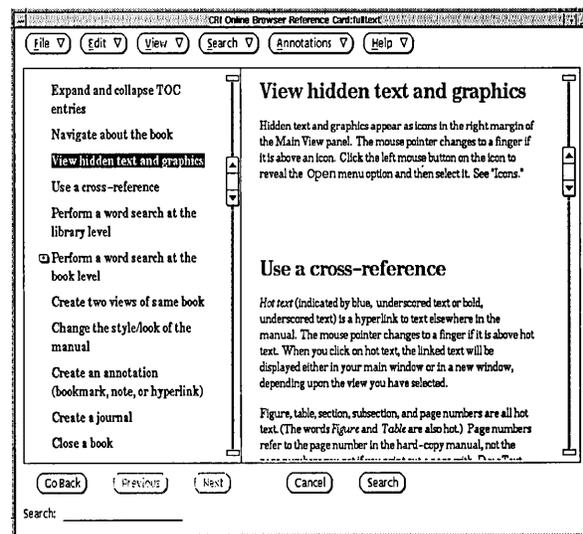
Current Plans

Our current plans are to continue developing IPSS components for existing products, moving along a continuum, from loosely integrated to fully integrated performance support.



IPSS Components

CrayDoc Online Documentation



Based upon feedback from customers, Software Information Services selected a tool to be used for delivery of online documentation. First we developed a list of customer requirements. Then, we developed a list of criteria by which we evaluated various tools. The following requirements are examples of the criteria we established.

Features:

- Hyperlinks across a library set
- Electronic notes
- Electronic bookmarks
- Ability to handle text, tables, and graphics
- Support for multiple languages
- Ability to handle multiple and large volumes
- SGML compliant

Delivery Options:

- CD-ROM
- Release tape

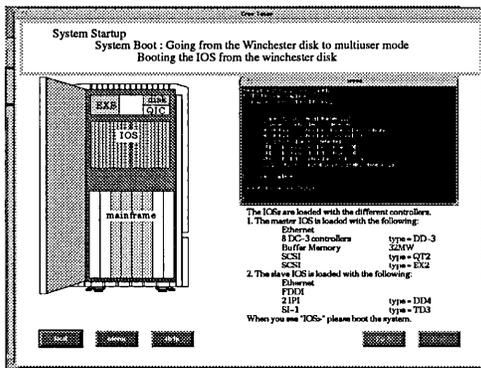
Portability:

- Sun, SGI, HP, Cray Research SuperServers, DEC, IBM, Macintosh, PC, Cray Research

We evaluated six tools against our criteria and selected a tool called *DynaText*, which we call CrayDoc, for delivery of Cray Research documents online. CrayDoc runs on various platforms, including the Cray SuperServer. It does not currently run on the Cray Research platform. The tool was thoroughly tested and prototyped last year and has been demonstrated to customers, who are all pleased with its features and capabilities.

The first product released with CrayDoc online documentation was CraySoft's NQE 1.0. For this product, documentation was delivered on a CD-ROM along with the product. We plan to deliver documentation online, with CrayDoc, for all CraySoft products, UNICOS 8.0.3., CRAY T3D 1.1, the Cray Programming Environment 1.1., and UniChem 2.2.

CrayTutor Computer-based Training



Customers also provided us with priorities for developing computer-based training (CBT) courses. Customer concerns include:

- Cost of training (time and money)
- High turnover rate of operators
- Inability to find information at the time of need
- Training that fits the needs of the student

- Need for an easy-to-use interface to the Cray System
- Need for training anywhere, anytime
- Lights-dim operating environment
- Lights-out operating environment

Last year, we released the Cray Operator Training courseware in response to some of these concerns. The course is intended for Cray system operators who are new to Cray systems, but is designed so that even experienced operators can learn from it.

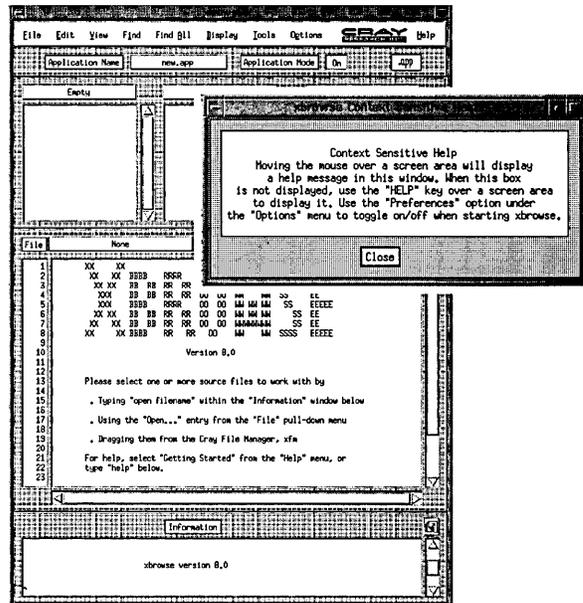
The benefit of using CBT is that the training is self-paced and available at the time of need.

Future CBT courses are being developed using a new, feature-rich tool called IconAuthor, which we call CrayTutor, for delivery of our CBT. CrayTutor courseware runs on UNIX workstations and uses color, graphics, animation, and sound. Courses being developed include:

- CRAY EL System Administration Training, Phase I
- Cray Operator Training for UNICOS 8.0
- UNICOS for Users-Basics (UNICOS 8.0)

Other courses and specific topics are being considered and prioritized according to customer needs.

Online Help and Messages



Online help and messages provide users with the most integrated form of assistance, information, and learning opportunities.

With online support embedded in the product, users can get quick answers to their questions. This improves customer satisfaction and allows Cray Research support personnel and user support services to focus on difficult problems instead of responding to a large number of simple requests.

Better access to information and the reduced amount of information required to perform the task enables users to perform their jobs without having to read through massive volumes of documentation.

One of the initial products with integrated online and context-sensitive help is the Program Browser (xbrowse), a Fortran and C source-code browser released last fall on Cray Research and workstation platforms.

Moving information into the product reduces the distance users must travel to get the information they need. Online help capabilities make information a mouse-click away. For example, through the xbrowse Help menu, users can select the Context Sensitive option to open a pop-up window that displays information specific to a procedure or window area. As the user moves through various windows, context-sensitive help information is readily available. In addition, xbrowse enables users to access man pages directly from the main xbrowse window by selecting the Man page option in the Help menu. Users never

need to leave their xbrowse session to find information about their tools and related topics.

Online help is also being developed for TotalView, an advanced symbolic debugger designed for Fortran and C programs with multiple parallel processes. By the end of 1994, many tools previously released with the UNICOS operating system will be unbundled and redesigned to include context-sensitive online help and will be released as part of Cray's programming environments.

Another product with context-sensitive online help is the Cray operator interface (opi). Help is provided in the form of information about operator tasks and functions. opi provides two windows that are used for standard output and error information.

Integrated Performance Support System Prototype

On the fully integrated end of the IPSS continuum, we are developing a prototype of integrated performance support for the Network Queuing Environment (NQE) 2.0 release. This prototype will demonstrate the integration of the IPSS components discussed earlier.

The goal of this project is to show the benefits of the fully integrated system by building integrated information, assistance, and learning opportunities into the product for one or more user-level tasks. By measuring user performance before and after the implementation of this prototype, we plan to gather quantitative measurements of user-performance improvements.

CRAY, UniChem, and UNICOS are federally registered trademarks and CrayDoc, CRAY EL, CraySoft, CrayTutor, CRAY T3D, Docview, and Network Queuing Environment are trademarks of Cray Research, Inc.

DEC is a trademark of Digital Equipment Corporation. *DynaText* is a trademark of Electronic Book Technologies, Inc. HP is a trademark of the Hewlett-Packard Company. IBM is a trademark and PC is a product of International Business Machines Corporation. Macintosh is a trademark of Apple Computer, Inc. SGI is a trademark of Silicon Graphics, Inc. Sun is a trademark of Sun Microsystems, Inc. SUPERSERVER is a trademark of Cray Research Superservers, Inc. TotalView is a trademark of Bolt Beranek and Newman Inc. UNIX is a trademark of UNIX System Laboratories, Inc.

NEW AND IMPROVED METHODS OF FINDING INFORMATION VIA CRINFORM

Patricia A. Tovo

Cray Research, Inc.
Eagan, MN

The CRInform (Cray Inform) program is an on-line menu-driven information and problem-reporting service for Cray Research customers. As more information has become available via CRInform, it has become increasingly difficult for the user to find new items or to search for particular types of information. Two new methods of finding information are being introduced in CRInform 3.0. The first method makes use of a text searching capability that provides very efficient searching through large collections of text. The second method provides faster and easier access to new items in CRInform and replaces the current site status file mechanism.

An overview of the major components of CRInform 3.0, new features in CRInform 3.0 including the details of the new and improved methods of finding information, CRInform 2.1 usage statistics, and future plans for the CRInform program are described below.

CRInform 3.0 Overview

CRInform 3.0 is scheduled for release on March 28, 1994. The major differences between CRInform 3.0 and CRInform 2.1 are:

- new user interface for menus
- replacement of the Site Status File
- additional information in email notifications
- full-text searching across documents

Before describing these and other changes, an overview of the CRInform 3.0 documentation and system is presented below.

CRInform 3.0 Documentation

The CRInform 3.0 release includes the following documentation:

- the CRInform User's Guide (SG-2125 3.0)
- Differences Between CRInform 3.0 and CRInform 2.1
- CRInform Program Flyer (MCSF-3230393)

The CRInform on-line help files and the on-line version of the CRInform User's Guide have been updated to reflect the CRInform 3.0 release.

CRInform 3.0 System

The CRInform 3.0 system consists of five major components:

- technical assistance and problem reporting
- CRInform/Software Problem Report (SPR) database
- service and marketing information repository
- software, publications, and training catalogs
- customer bulletin board

These five components will be described below.

Technical Assistance and Problem Reporting

The technical assistance and problem reporting component allows customers to request assistance and report problems in the customer's native language. The request is entered as a Request for Technical Assistance (RTA) that is forwarded to the customer's regional support center. An RTA can be any request or question, or it can be a software problem report. Through this mechanism, the customer can request that the RTA be converted into a Cray Research Software Problem Report (SPR). If this occurs, the customer will be notified of the SPR number in the RTA's resolution. When an RTA is submitted, it is assigned a number immediately for future reference. When an RTA is closed, the resolution field is updated by the customer's regional support center. The status of a customer's RTAs can be queried using the Status of RTA query screen, and new information can be added to an RTA that hasn't been closed.

CRInform/Software Problem Report (SPR) Database

The CRInform/SPR database contains software problem reports for released CRI products. Customer SPRs are included for all customers that are participating in the CRInform Program, i.e., for customers that have signed the CRInform Program Agreement. In addition, CRI internal SPRs are included for released products. All SPRs in the CRInform/SPR database are accessible to all users with the site information removed from the SPRs. As of the CRInform 3.0 release, 205 customers are participating, and 85% of the SPRs are available.

The CRInform/SPR database can be queried using the Query SPR Database screen. Using this mechanism, the user can

quickly view the SPRs that originated at his/her site, and/or view SPRs from other customer sites in order to determine if a particular problem has already been reported, and if so, if and how it has been solved.

The customer can add information to an SPR that hasn't been closed. This information is added to the CRInform database (as well as the Cray Research internal SPR database), and is immediately forwarded to the customer service analyst handling the SPR. This allows for a dialog between the customer and the Cray Research analyst.

Information Repository

CRInform includes a repository of service and marketing information. The service and marketing information includes:

- the Cray Research Service Bulletin (CRSB)
- software field notices (SFNs)
- software release documents
- software problem fix information
- company and product announcements

Catalogs

CRInform contains on-line versions of the following Cray Research catalogs:

- Cray Research software catalog
- Directory of Application Software
- User Publications Catalog
- software training catalog

The CRInform user can order from the software and publications catalogs and communicate with the regional training registrar.

Customer Bulletin Board

The objective of the CRInform 3.0 customer bulletin board is to provide communication among Cray Research customers. The Bulletin Board System (BBS) is based upon Usenet, a network message-sharing system that exchanges messages in a standard format. Messages are arranged into topical categories called newsgroups. The CRInform newsgroups are only accessible by CRInform users, that is, Cray Research customers who have signed the CRInform Program Agreement, and Cray Research employees who have requested access to the newsgroups. Messages sent to the unicos-1 email alias are being archived in a CRInform newsgroup, `cray.crinform.unicos-1`.

New Features in CRInform 3.0

Seven features in CRInform 3.0 will be discussed in the following section. Five of these features will become available on March 28, 1994, and the other two features have already been

made available in CRInform 2.1. All of these features were chosen based upon customer requests, and most of them deal with the need for finding and navigating through information quickly and easily.

The five new features are:

- new user interface for menus
- Site-related news
- additional information in email notifications
- full-text searching
- new CRInform machine

The other two features that will be discussed are:

- customer mailing list maintenance
- unicos-1 news group

New User Interface for Menus

CRInform 3.0 offers an enhanced user interface for its browsing menus. Prior to CRInform 3.0, the protocol for menu item selection has been select-by-letter only. The new protocol is similar to that of popular mail and news reader programs like elm and tin. The primary feature of the new protocol is that there is always a highlighted menu item (either inverse video or underlining). In addition, menus are not limited to one page as they were in the previous release.

Site-related News

The Main Menu option Site-Related News replaces the Site Status File. Site-Related News provides immediate access to actual documents as opposed to just the notification of the existence of new or updated documents as in the Site Status File mechanism.

The types of events logged have not changed:

- changes in software problem reports (SPRs) or requests for technical assistance (RTAs) activity
- new orderable software
- new Cray Research Service Bulletins
- new or updated software field notices
- new software release documents
- new software problem fix information
- new marketing information
- new CRInform program information

In earlier versions of CRInform, the content of email notifications was identical to what was written to the Site Status File. This is no longer true for Site-related News. For example, if an SPR is listed under the Site-related News option, you can

select and view it on the spot as opposed to having to return to the Main Menu and select another option to view the SPR. You can randomly read items, save items to an external file, and delete items.

Additional Information in Email Notifications

The format of email notifications will be essentially the same as in CRInform 2.1. The logged events are the same as those in the Site-related News option described above. However, to provide quick access to software field notices (SFNs) and software problem reports (SPRs), CRInform now appends the actual text of new or updated SFNs and SPRs relevant to your site.

Full-text Searching

CRInform 3.0 provides full-text searches across documents and document collections. The full-text searching is provided both within a single category of information (for instance, Marketing Information) or across multiple categories. A new menu item has been added to the Main Menu for searching across multiple categories of information.

The following types of searches are supported:

- boolean searches
- phrase searches and wildcards
- relevancy ranking
- a thesaurus function

The results list of a search is a list of documents each of which you can search or page through individually with the search terms highlighted. If you perform multiple searches over a given document collection, CRInform saves the results of each search and lists them so that you can re-examine them or use them as the base for a subsequent search.

New CRInform Machine

The CRInform machine has been upgraded from a Sequent S81 running DYNIX to a Sun S1000 running Solaris 2.3. Solaris 2.3 is a System V Unix system.

Customer Mailing List Maintenance

The CRInform 3.0 user can maintain their site's mailing list for the following Cray Research communiques:

- Cray Research Service Bulletins (CRSBs)
- Software Field Notices (SFNs)

These list maintenance features are provided through the Utilities option from the Main Menu. The user can add, delete, or otherwise alter the entries from the lists.

unicos-1 News Group

The CRInform Bulletin Board now includes messages from the email alias unicos-1, which is maintained by the Cray User Group (CUG). You can access the CRInform Bulletin Board from the Main Menu or by using the news reader tin or rn from the UNIX prompt.

CRInform 2.1 Usage Statistics

Usage patterns of CRInform 2.1 are very similar to those described in the Fall, 1992 CUG Proceedings for CRInform 1.0. Fifty percent of all usage involves browsing service related information including 34 percent querying the SPR database, and 16 percent accessing service related textual information including the Cray Research Service Bulletin, field notices, release documentation, and software problem fix information.

Seventeen percent involves browsing and ordering from the software, publications, and training catalogs (9%-software, 5%-publications, 3%-training).

Eight percent involves reading the bulletin board. Note that most of this usage is read-only. The only on-going source of information being posted to the bulletin board is the unicos-1 mailing list.

Seven percent of the usage involves reading the marketing information, i.e., product and company announcements.

Six percent of the application usage is in the area of technical assistance, i.e., submitting RTAs and obtaining the status and resolution of RTAs. 4 percent is in the area of mailing list maintenance, and the remaining 8 percent includes file transfer via electronic mail (3%), viewing CRInform News (3%), and viewing the site status file (2%).

Future Plans

Cray Research is planning a CRInform 3.1 minor release in September of 1994, and a CRInform 4.0 major release in March of 1995. We plan on extending the user interface concepts used in the CRInform 3.0 browsing menus to the data input menus including the menus used to query the SPR database. In addition, we plan on extending the full-text searching capability to include the SPR database.

We need customer input to help determine the requirements and priorities for these releases. Input can be submitted via the CRInform utility for sending comments to the CRInform administrator or given directly to a Cray Research service representative.

To obtain access to CRInform, Cray Research customers must submit a signed CRInform Program Agreement form to their Cray Research service representative.

Online Documentation: New Issues Require New Processes

Juliana Rew

Documentation Group, User Services Section
Scientific Computing Division
National Center for Atmospheric Research
P.O. Box 3000
Boulder, Colorado 80307-3000

ABSTRACT

While acknowledging that hardcopy documentation still has an important place, this presentation discusses the following:

- The advantages of using online documentation in light of new advances in information technology on the Internet
- The problems that documentation providers and maintainers face in having to maintain multiple versions of the same document, each tailored to certain online delivery systems
- The methodology of "process reengineering" and identification of areas where future improvements in online documentation may be possible through designing new processes

Introduction

The business community has given wide credence to a new management technique called "process reengineering." Some performance experts say most work processes can be broken down into five basic, elemental steps (see Figure 1). Process reengineering is aimed at maximizing the first step (real work), while minimizing or eliminating the other steps in an effort to increase efficiency.

What does process reengineering have to offer to computing technology? In this article, we will show how one particular area of information technology—online documentation—may naturally take advantage of potential efficiencies to be gained from process reengineering. While acknowledging that hardcopy documentation still has an important place, we will (1) discuss the advantages of using online documentation in light of new advances in information technology on the Internet (including timeliness and availability, easy-to-use browsers, choice of searching mechanisms, training and documentation, consulting, printing on demand, and use of standards) and (2) identify areas where future improvements may be possible through process reengineering.

Potential Process Improvements Offered by Information Technology

Information technology refers to computers and automation. We believe information technology can play a key role in reengineering processes to eliminate and minimize waste.

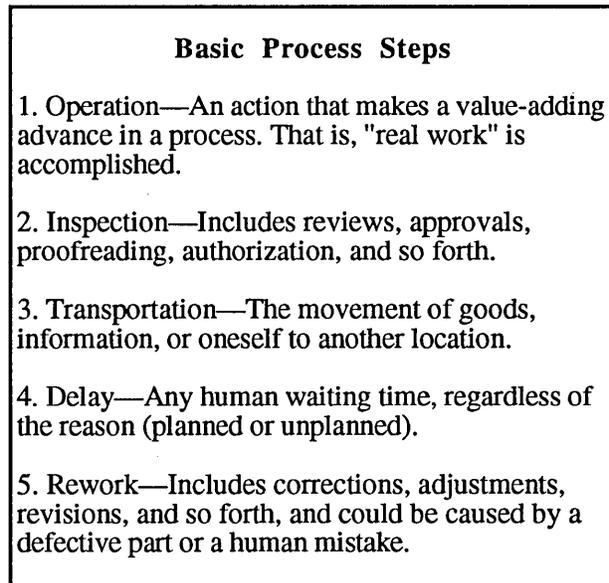


Fig. 1. Five basic process steps.

To use the jargon of the performance improvement field, if you use a FAX machine or e-mail instead of regular mail, you are using a "transportation minimization device" (TMD). If you take notes at a meeting on your laptop computer instead of writing them by hand and transcribing them later, you are using a "rework minimization device" (RMD), because you are not reentering information, and you are eliminating the potential for the introduction of new typing errors.

You get the idea. Now, say you use your laptop to read a manual online. You've got yourself both a handy "transportation elimination device" (TED) and a "delay minimization device" (DMD), because you have avoided the trip to the documentation office and (or) the time it takes to mail the hardcopy document to you.

There are process efficiencies to be gained for the documentation maintainers too. By observing the rule of collecting data only once, at its source, we can make the computer serve as a "data source entry device" (DSED). Then we can devise efficient ways to reuse the data.

Online Documentation Finally Comes of Age

While some forms of online documentation, such as UNIX man pages, have been a modest success, most online documentation has been slow to gain popularity. (Acceptance, yes, but not popularity.) Some commonly cited reasons are that it is harder to read much text on a screen, and most nonproprietary online documentation systems have been either ugly (as with ASCII text) or unsearchable (as with PostScript documents). Hardcopy is a time-honored, portable, reliable user interface. While eliminating hardcopy may save the documentation maintainers the effort of printing and distribution, many readers are not terribly impressed with such advantages. However, recent advances in accessibility and convenience mean that online documentation is finally beginning to measure up to its potential.

Advantages of Online Documentation on the Internet

Timeliness and availability. The growth of the Internet offers the potential to make information available almost instantaneously to more people who otherwise would not have had access to it, including high-level managers and ordinary workers. This enabling technology can help everyone make better decisions in their jobs. Online documentation is unbeatable in the area of timeliness.

Easy-to-use browsers. Systems such as NCSA Mosaic and the University of Minnesota's Gopher (see the article "Internet safari: Exploring the world of online information" in this issue) have made it much easier to read online documentation, and multimedia capabilities make it possible to browse documents containing pictures, sound, and animation across the Internet.

Choice of searching mechanisms. Users now have a variety of strategies for "discovering" information. Gopher excels at keyword searching through its Veronica utility, while the World Wide Web offers the ability to "jump" to information via hypertext links. Archie lets you search for files on anonymous File Transfer Protocol (FTP) servers around the world. The Wide Area Information Servers (WAIS) give you full-text retrieval of the information in hundreds of databases. The goal of letting the customer control/activate the process as much as possible is finally being attained.

Training and documentation. Users are helping other users train themselves to make the most of the new online information discovery services. For example, Richard J. Smith (Carnegie Library of Pittsburgh) and Jim Gerland (State University of New York at Buffalo) recently offered a "Navigating the Internet—Let's Go Gopherin" course to 17,000 people via e-mail. (See "E-mail course teaches Gopher search tricks," in this issue.) Documentation for online systems such as Gopher and Mosaic is itself online. Lunchtime seminars on networking are becoming popular at universities and companies. These changes break the old rule that only experts can successfully perform complex information searches.

Consulting. The placement of all documentation online makes it easier for you to rapidly look up simple questions for yourself. This has the added advantage of freeing our consultants to serve you better on complex questions.

Printing on demand. Systems such as Gopher and Mosaic allow you to save and print off passages locally. SCD also plans to enable users to print large PostScript manuals economically on our high-speed Xerox 4050 laser printers.

Use of standards. The new Internet tools use a few, commonly used standards. This makes them more accessible to users with machines ranging from PCs and Macintoshes to high-performance mainframes. Most of the tools have dumb terminal interfaces as well as windowing interfaces. Multiple-protocol servers such as GN may offer a single interface to both Mosaic and Gopher protocols.

Notice of New Information

An important service is to notify readers when new information is available and to guide them to what's new. The Mosaic browser has begun to address that question by offering a "bulletin board" where information providers on the World Wide

Web can announce new products and servers. A similar facility is needed for individual information servers. It would be even better if new documents automatically announced themselves!

New Issues Require New Processes

With the large increase in the number of possible avenues for online information dissemination, documentation providers and maintainers face the hazards of having to maintain multiple versions of the same document, each tailored to certain online delivery systems.

An obvious solution appears to be to select the enabling technology of a "shared database" of documents, selecting one application such as FrameMaker for authoring the source documents and using filters, templates, and macros to generate the additional versions needed. Automation is called for here, both to avoid the problem of versions getting "out of synch" and to avoid the "rework" of maintaining several versions manually.

SCD sponsored a project recently in which senior computer science students from the University of Colorado created a document database manager that would catalog source documents and provide online ASCII translations on the fly using built-in filters. We found that translation on the fly effectively dealt with the requirement that each filtered version must simultaneously reflect any changes. Future versions of Mosaic may provide "hooks" to allow us to insert on-the-fly translators.

A "reengineered solution" to this question might require discarding a proprietary authoring application in favor of a more universally applicable text markup language such as the Standard Generalized Markup Language (SGML). SGML was developed to provide device independence (rather than usability across a selected set of common platforms as with FrameMaker). Thus, if your authoring platform is not around in 10 years, your document should still be readable by future generations of machines. Encouragingly, the World Wide Web is moving in the direction of SGML by using Hypertext Markup Language (HTML), which is a simple subset of SGML. Less encouragingly, WYSIWYG applications such as FrameMaker are not yet fully SGML compliant.

Other avenues for reengineering might include information mapping and radical downsizing of documents to facilitate reading in hypertext "chunks." A well-done, concise "Frequently

Asked Questions" (FAQ) list is potentially more useful than a whole manual.

Icebergs

However desirable, *automation can only be successful for processes that are stable.* FrameMaker recently released a new upgrade. How will the current filters to ASCII and HTML interact with it? How will we deal with "new, improved" filters? Likewise, PostScript is an evolving entity. New online documentation products, such as the Mosaic browser, do not yet deal with the problem of finding the differences from previous versions. None of the tools are much help in making sure that multiple representations of the same information are kept in synchronization. Automated conversion scripts require frequent maintenance when both the source and target forms are evolving rapidly. It's like trying to ride two icebergs floating in different directions.

It is also not clear yet whether process reengineering has anything to offer to the process of creating information. In the past, it has always required human brainpower to organize the information and to express it in natural language. Paradigms such as hypertext differ radically from databases and seem to defy any but simple automation attempts. For example, our hardcopy summer issue of *SCD Computing News* contained 27 articles. By contrast, the Mosaic (hypertext) version consists of 107 files, including pictures and cross references, that users can jump to. Even if they are "synchronized," are these two versions really the "same" document?

Another question is how to monitor the status of frequently changing documents. If we make several isolated changes over time, each change may seem simple and straightforward in its limited context, but the resulting document as whole may become sufficiently different that at some point it requires total renovation.

Finally, the "payoff" for using SGML/HTML may be long in coming. Coding in SGML represents a large investment of time, and is best suited to documents that will have a long shelf life.

The Future: Keeping a Step Ahead

The rise of online documentation will bring with it many other issues, including authenticity and accuracy of information, research privacy and patents, and copyrights. We feel the potential advantages to be provided by online documentation

make it worth the effort of reexamining the processes we use, in spite of the challenges listed above. As enabling technologies are identified, we can then readily take advantage of them.

We look forward to the opportunity offered by partnering process reengineering and information technology. We will seek out innovative ways to deliver up-to-date information to our users that is more usable and attractive yet easier to provide and maintain. We may have to break a few old rules along the way. As Michael Hammer, process reengineering guru and former MIT professor, has said, "The real power of technology is not that it can make the old processes work better, but that it enables organizations to break old rules and create new ways of working—that is, to reengineer."

An online version of this paper can be viewed on the WWW. Its URL is:

<http://www.ucar.edu/docs/cug94.rew.html>

MetaCenter Computational Science Bibliographic Information System: a White Paper

by Mary Campana, consultant to SDSC, and Stephanie Sides, SDSC

This paper describes the first phase of developing a full-featured one-stop research information service described at the Cray User Group meeting March 17, 1994.

Over the last eight years, the four national supercomputer centers¹ and some 10,000 researchers that make use of the centers have produced a unique collection of written, numeric, and graphical information related to advances in computational science and the underlying computer (and computer-related) technology. This information includes journal and conference publications, technical reports, user documentation, datasets, hand-written computer programs, still images, and animations. The information includes records of citations of materials as well as the physical materials themselves.

A survey of this information was made in January 1994. A snapshot of this survey reveals the following

- Over 14,000 journal articles, research reports, etc.
- 3,000 online documents
- 500 documents containing workshop notes or tutorial information
- 6.2 TBytes of data
- 1,400 still images on film
- 400 digital animations and analog videotapes
- 6,000 titles of hardcopy materials related to supercomputing, visualization, networking, etc., held by the centers' libraries

Although many of the journal and conference publications can be located through indexes prepared for a particular scientific discipline (e.g., physics, chemistry, and atmospheric sciences), indexing has generally not been done using computational terms. For example, it is not possible to search information by hardware platform, applications package, or algorithm used as part of the research project. Some materials, such as the reports, visualizations, and other "internal" materials, are simply not indexed anywhere.

¹The Cornell Theory Center (CTC), the National Center for Supercomputing Applications (NCSA), the Pittsburgh Supercomputing Center (PSC), and the San Diego Supercomputer Center (SDSC).

This material holds unique value for its potential to fuel the scientific process and serve as a historical archive. But in its current form, it remains unusable because it has not been

- Consolidated as a single collection
- Indexed comprehensively to make it easily searchable from a variety of perspectives (e.g., scientific, computational)
- Made available electronically to the Internet at large

This problem is compounded by the lack of sophisticated search tools on the Internet that would provide effective mechanisms to access and search this information. WAIS, veronica, gopher, and archie² are good tools for locating resources, but they lack the ability to perform sophisticated text retrieval searches within a collection of resources. Additionally, they lack important capabilities that would make relevant information much easier to find that include the following:

- Syndetic structure to enable a user to move from what s/he asks for to what s/he needs at a specific source of information
- Keyword indexing based on the terms people are most likely to use
- Cross references
- In-depth search and retrieval mechanisms
- Boolean and wildcard searching capabilities
- Standardized terms
- Logical and useful filenames

Therefore, the centers propose to develop an integrated, online computational science information system containing bibliographic information about work produced by the centers' aggregate users and staff. The proposed system will include:

- An efficient, cost-effective method of collecting the information produced by the four centers so that it will be up-to-the-moment in currency.

²NCSA Mosaic and PSC GDOC have much greater capabilities than these tools but do not presently include indexing.

- In-depth, precise indexing based on a thesaurus of standardized terminology, which will permit accurate and comprehensive retrieval of information primarily from a computational perspective.
- A powerful search engine that will permit natural-language query, relevance ranking, and retrieval of text through a syndetic structure.³ This engine will not only permit the system to be searched easily and thoroughly, it will be a tool usable throughout the Internet, greatly improving Internet users' ability to search for and find relevant information in other contexts.
- A standard interface usable by non-computer professionals on the Internet.

This project will serve as a testbed for a more comprehensive information system which will include datasets, still scientific images, digital animations, computer programs, and other non-bibliographic materials. The information system will also expand beyond the materials of the MetaCenter to incorporate computational science information from high-performance computing centers throughout the nation and eventually the world.

This project will be facilitated greatly by taking place in the context of the MetaCenter, a collaboration of the four national supercomputer centers formed in 1992 to leverage their diverse resources and expertise so as to advance computational technology and promote scientific progress. Significantly, an important part of this collaboration is the MetaCenter's involvement in advancing the National Information Infrastructure.

In 1992 the San Diego Supercomputer Center wrote a proposal to Cray Research, Inc., and obtained \$25,000 to assess the options for creating a computational science information system. The library at SDSC and Datasearch, Inc., of Ithaca, NY, which works with the Cornell Theory Center, have inventoried the various collections of materials at the centers. During 1994 Datasearch is conducting a feasibility study to recommend the design of this system and how to implement it. A final report is expected late this year recommending a suitable hardware platform, a teaming arrangement with a selected software vendor, procedures for consolidating and maintaining the data, and potential sources of funding to implement the proposed system. The Cray funding is not adequate to pursue the project beyond this feasibility phase.

³Serendipitously, concurrent with the centers' efforts to produce this online system, private industry has developed several search tools that may be useful for this project. For many years the library and information science community and related industries have been at the forefront in developing precise, user-oriented search engines. Dialog, Mead, Datastar, Westlaw, Personal Library Software, and Conquest are some of the companies whose research and development efforts in this area are used daily by the library and information provider community. This community has watched the growth of the Internet and tools such as gopher and WAIS, and has participated in discussions and projects attempting to team the search-and-retrieval expertise developed by libraries with emerging networking technologies. The MetaCenter's computational science information system is in a unique position to leverage the strengths of both the computer science and the library science disciplines simultaneously to produce the first such prototype system.

The centers expect industry, foundations, and government to be interested in this project and are seeking cost sharing from all three. All major supercomputer vendors are expected to be interested in seeing this project move forward as this system will highlight the work accomplished on their platforms and encourage demand for their products. Several foundations have shown interest in the advancement of scientific information systems. And we expect the government to be interested because of this project's potential in making a significant contribution to Internet resources by providing a unique source of information about computational science.

The Internet has been described as "a library with all the books thrown on the floor in the dark." The MetaCenter computational science information system can provide both the light and the catalog for that library.

Electronic Publishing: From high-quality publications to online documentation

Christine Guzy, Production Coordinator

*Documentation Group, User Services,
Scientific Computing Division
National Center for Atmospheric Research*

Abstract

This presentation will cover electronic publication of printed and online documentation required by users of a high-performance computing center. It will emphasize the transition from one format to the other while minimizing maintenance of many versions in different formats, such as ascii, word-processing and text-layout software, and formats recognized by online browsers. Research and development techniques and tools will be discussed. Printed documentation and brochures as well as their NCSA Mosaic online counterparts will be highlighted.

Introduction

We documentation specialists are all adjusting to the challenges of new and fast-changing technology in presenting information and documentation to our Cray users. While there are a lot of advantages to this technology, there are also frustrations.

This paper is about some of the advantages that electronic publishing technology has given us, including higher resolution hardcopy publications, increased efficiency, and online access.

An overview of some of the filtering tools that we use to convert file formats to make them compatible for importing illustrations into text files will be presented. There is also an overview on filters for converting various file formats into ASCII and HTML.

We currently use Sun workstations and Macintoshes. The software we use is FrameMaker (on both platforms) for most of our documentation. We use Microsoft Word and PageMaker for the newsletter. We also use Adobe Illustrator and Adobe PhotoShop.

Advantages to electronic publishing

By outputting our documents directly to film at high resolutions, we are able to save a generation in quality. In the past, the print contractor made the negatives from our paper output.

By ordering photo scans directly from negatives we save the cost and step of printing photographs. These scans can then be used in our hardcopy as well as online publications.

In the case of our Newsletter, the hardcopy contains black-and-white photos while the online version can display photos in color.

By importing color illustrations electronically, an entire page can be output as one color separation. Typically the cost of a color separation is relative to the size of the separation, plus the stripping costs. However, in my experience, the cost is no more than traditional ways even though in effect you're buying an 8 x 10 separation.

Scanning and placing photographs and artwork into the files electronically is more efficient and accurate than indicating crop marks and placement to print contractors. Electronically placed illustrations can be proofed along with their captions and in relation to the text. This eliminates the introduction of errors by print contractors, such as switched or inverted photos and illustrations.

Templates make short work of formatting

Templates automate the tedious task of formatting text. To format text you simply click on the format you want for a particular level of heading or body copy. For example, if you want body text, you place your cursor in the text you want formatted, go to the menu containing the name of the style, and click on it. Voila! Consistent formatting. This is vital to good documentation and especially when filtering one file format to another.

Our FrameMaker, Microsoft Word, and PageMaker software templates have been standardized to make it easier for us to filter files from one format to another. We try to keep the style names consistent even though the formatting may not look the same from one document to another.

Obtaining filtering tools

Programs are necessary to filter incompatible file formats for importability. Also, getting all the files into electronic format is just the first step in putting information online. For instance, we convert FrameMaker mif (Maker Interchange Format) files to html (HyperText Markup Language) for viewing on NCSA Mosaic.

There are an infinite number of filters available on the Internet. While these filters may not solve all of your filtering needs, "new and improved" versions crop up frequently,

so check often. Also, be aware that software upgrades can become incompatible with filters that were previously successful and that can send you back into search mode.

Some ways to find filters are:

- Use Veronica to search Gopher server menus
- Use Archie to search anonymous FTP (File Transfer Protocol) archives
- Use WAIS (Wide Area Information Servers) to search indexed databases

A compilation of html text filers can be viewed with NCSA's Mosaic. See the following URL (Uniform Research Locator) on the WWW (World-Wide Web) in hypertext:

<http://info.cern.ch/hypertext/WWW/Tools/Overview.html>

It is important to use caution when moving formats from one platform to another. Not all file formats are ascii, like PostScript and EPS. Formats such as TIFF and EPS files with preview images are binary and hex files. The easiest way to avoid corrupting files is to assume the files are binary and move them accordingly. It is best to use FTP in binary mode.

Using filters

Some ways to get formatted ascii text. (By formatted I mean with some tabs between items in tables and indents for headings and such):

- Microsoft Word — simply save as "text with formatting"
- FrameMaker — use miftortf, open in Microsoft Word, save as "text with formatting"
- NCSA Mosaic, save as "formatted text"

Some filters that we find useful:

- **ps2epsi** — PostScript to Encapsulated PostScript (EPS) with optional preview Image (EPSI) for importing into FrameMaker and Macintosh programs.
- **xv** — interactive image display for the X Window System, for saving screen dumps in formats such as tiff (for importing into text files) and gif for use with html.
- **fm2html** — FrameMaker mif to html
- **Adobe Photoshop** — for saving pict files to tiff, eps, gif and many other formats. Also good for enhancing, cropping, and rotating photos and images.
- **ncgm2fmeps** — Converts NCAR Graphics into epsi for importing into FrameMaker

NCAR Graphics is a software package developed by SCD's Scientific Visualization Group. The standard output format of the graphics package is a cgm. This filter was developed so we could import examples into the documentation for the software package.

Conclusion

Streamlining and automating production tasks, and keeping up with technology in the field of electronic publishing pays off in the long run. Or as Juli Rew mentions in her presentation, "process engineering" and working "smarter not harder" really increases efficiency.

Our online user documents now include, in addition to text, graphics, illustrations, and photographs. In addition to our hardcopy documents, we now have our SCD User-Docs available via anonymous FTP and Gopher, as well as html-coded files for viewing with NCSA Mosaic.

The quality of a publication is what attracts the reader to take a better look at the contents of a document. Our documents are designed to assist our users in finding the information necessary to perform their computing tasks. The easier we make it for them to access that information, the less the phones ring in the consulting office.

For more information on:

- Using Archie, see "Consult Archie, the FTP archive guru, to find files" in the March 1992 issue of *SCD Computing News*.
- Using Internet tools, see the "Special issue: Online information tools," Jan./Feb. 1994 issue of *SCD Computing News*.

JOINT SESSIONS

MSS/Operating Systems

WORKLOAD METRICS FOR THE NCAR MASS STORAGE SYSTEM

J. L. Sloan
National Center for Atmospheric Research
Scientific Computing Division
Boulder, Colorado

Abstract

The NCAR Mass Storage System, MSS-III, generates ten megabytes a day of transaction log containing a wealth of information about its workload. Traditional metrics such as overall mean are useful, but omit information regarding temporality, locality, and burstiness. NCAR has begun to examine metrics usually applied to virtual memories and hardware caches to more precisely characterize the MSS-III workload. These metrics are generated by trace-driven programs which calculate the working set of the workload, and which simulate portions of MSS-III as caches. The cache simulation portion of the project is still being validated.

Introduction

About a year ago, Dr. Bill Buzbee, director of NCAR's Scientific Computing Division, asked: if funding were available to expand NCAR's mass storage system, how could it best be spent? This was a perfectly reasonable question. Answering it with confidence turned out to be more difficult than we expected. This paper describes how we have applied tools and generated performance metrics traditionally used in areas outside of mass storage.

NCAR's 38 terabyte mass storage system, MSS-III, moves an average of about five terabytes of data per month (eight terabytes peak) responding to user requests. An equal amount of data is moved within the storage hierarchy as part of cache management (Harano [1]). The MSS-III caches consist of a 1.2 terabyte robotic tape library (which we will refer to as the tape cache), and a 120 gigabyte disk farm (the disk cache). These caches are fed from a manual archive of 112,000 tape cartridges. Where files are cached is based on their size relative to an MSS-III file threshold parameter. Currently, files smaller than 30 megabytes go to the disk cache, larger files to the tape cache. Clients served by MSS-III include a CRAY Y-MP8/864, a CRAY Y-MP2/216, a Cray-3, a TMC CM-5, an IBM SP-1, and dozens of smaller file and compute servers. The primary method of connecting to the MSS is via HIPPI channels through a series of cascaded channel switches.

In the execution of its duties, MSS-III generates transaction logging records to the tune of about 10 megabytes a day. These transaction logs can be mined for information about what the MSS is doing, when it is doing it, and who it is doing it to and for.

Distributions

We began mining the transaction logs and tabulating typical statistics such as the number of files read, megabytes moved, and effective transfer rates, by MSS device, client system, and user. This allows us to generate spiffy charts like *Figure 1*, which shows the average transfer rates in kilobytes per second between MSS-III and different types of client systems for files of different sizes. However, we convinced ourselves that while these types of statistics are useful, they do not tell the entire story.

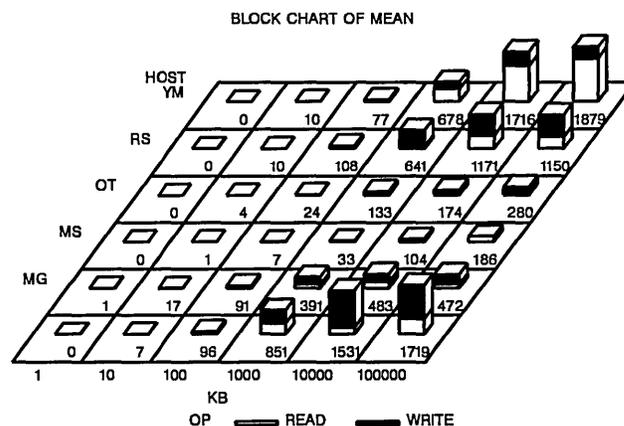


Figure 1. Average transfer rates in KB/S

For example, a statistic that is frequently bandied about is that the average size of a file accessed from MSS-III is about 28 megabytes. However, the frequency distribution for accesses by file size for the 1.2 million files in the MSS is not normal; the distribution is strongly biased toward small files. Given any MSS file access, there is nearly a 50% probability that it is for a file smaller than one megabyte, and almost an 80% probability that it is for a file smaller than 10 megabytes. This is illustrated in *Figure 2*, which shows the distribution of file requests by file size in quanta of one megabyte.

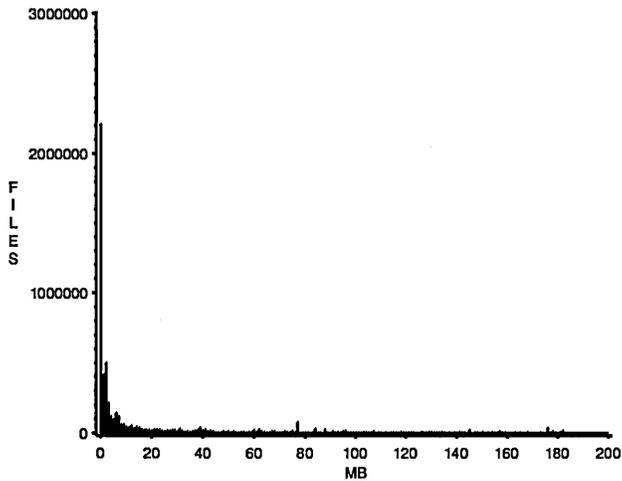


Figure 2. Distribution of files by file size

Even so, given any byte accessed from the MSS, there is less than a 1% probability that it is from a file less than one megabyte in size. Highest probabilities are for files in the neighborhoods of 80, 145, and 180 megabytes. This is probably an artifact of the file usage patterns of the climate models run at NCAR. This illustrates the importance of examining statistics and distributions derived not only from how files are used, but how the data is used; the distributions are frequently quite different. *Figure 3* shows the distribution of data requests in megabytes by file size.

Running averages

Although distributions are important, they are inadequate. Consider these three cases:

- a) one one-megabyte file is read 100 times;
- b) 100 one-megabyte files are each read one time; and
- c) one 100-megabyte file is read one time.

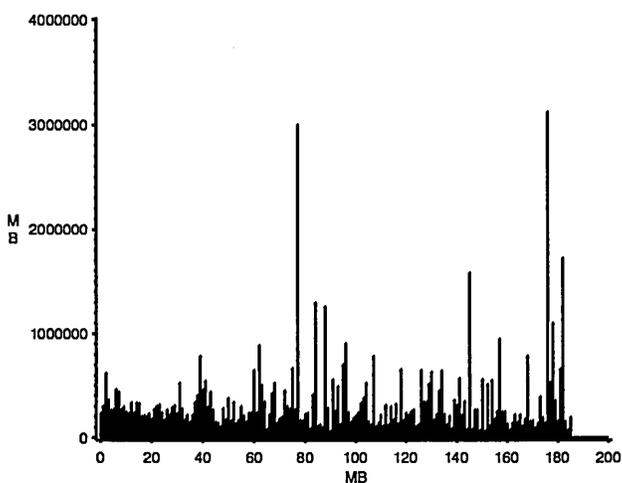


Figure 3. Distribution of data by file size

Cases (a) and (b) are identical in terms of number of files transferred, and all three cases are identical in terms of number of bytes transferred. Yet it is easy to see that the impact that each of these cases could have on the MSS might be as much as two orders of magnitude apart. For example in (a), the file might be cached immediately, resulting in a single manual tape mount, while (b) would result in 100 manual tape mounts. Fixed over head per file transfer would be incurred 100 times for (a) and (b), but only once for (c). (a) only takes up one megabyte in a cache, while (b) and (c) each take up 100 megabytes. There is a temporal quality lost in the compilation of distributions; it matters not only that a file was read, but when and in what context it was read. To recover this information, we had to turn to trace-driven analysis of our transaction log data.

Trace-driven analysis is an attempt to show how different aspects of the MSS workload change over time. The analysis is driven by the time stamped transaction log records. This method was inspired by earlier work characterizing the UNIX 4.2BSD file system (Ousterhout [2]) and supercomputer file access (Miller [3], [4], and [5]).

We used running averages to try to get a handle on how the workload changed over time. *Figure 4* shows how the running average of the number of megabytes moved per hour between client systems and the MSS tape cache varies from December 1992 to November 1993. Note how the running average fluctuates until enough time has elapsed for the graph to settle down and converge to a long-term average.

Linear regression

Using the running average for trend analysis is troublesome. The longer a span of time the analysis covers, the greater a change is required to perturb the graph. Over very long runs (a year or more), significant workload peaks visible to the users

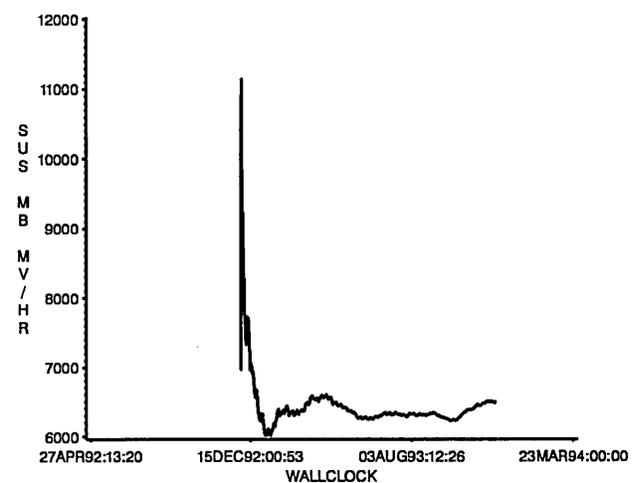


Figure 4. Running average of megabytes moved per hour

may not be visible in the running average. On the other hand, very long runs were necessary for the averages to converge on a result we believed. The running average did not adequately capture the bursty nature of the MSS workload, and it was hard to extract any clear trend from the resulting graph.

Unfortunately, the burstiness evident in the raw data makes any trend analysis difficult. Regression analysis usually results in confidence limits that do not inspire confidence, or the variance is so large as to make the regression line incredible. *Figure 5* shows a linear regression with a 95% confidence, plus a scatterplot of the raw data, of the number of bytes moved between client systems and the MSS-III tape cache for the December-November time frame.

Working sets

Intimidated by the bursty nature of the MSS workload and disheartened by the relatively poor results provided by either running averages or linear regression for trend analysis, we began to shop around for better metrics. The fact that the MSS-III caches operate a bit like virtual memory inspired us to wonder if working sets (Denning [6]) might be applicable. A working set is defined as the amount of unique data referenced within a given time window. Our time window would be our target residency periods for cached files (currently five days for the disk cache and 30 days for the tape cache). The working set of an MSS cache would tell us how large a cache we would need to achieve a target residency.

Figure 6 shows how the 30-day working set of the MSS-III tape cache changed during the December-November time period. We knew from our distribution statistics that we were meeting our desired residency time for files in the disk cache, but were falling short for files in the tape cache. Determining the working sets of both caches told us why. We found that although the requests for files from the MSS were biased towards small files that would be kept in the disk cache, the

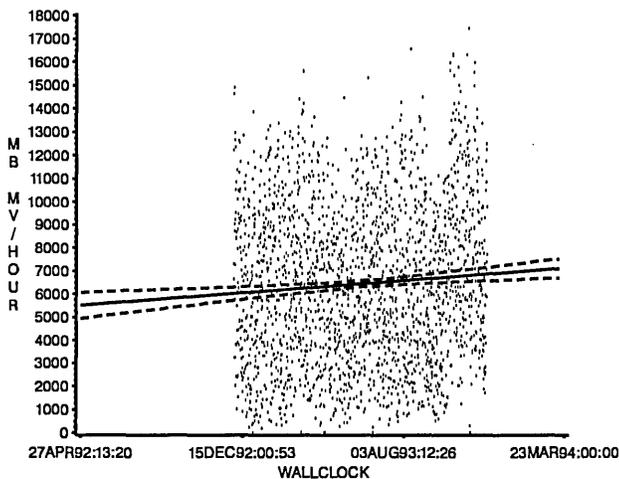


Figure 5. Linear regression of megabytes moved per hour

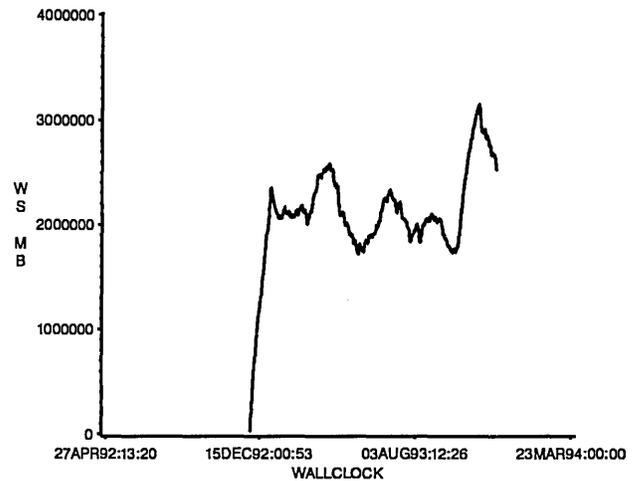


Figure 6. 30-day working set of the tape cache

five-day working set of the small file workload fit comfortably on our disk farm. Unfortunately, the working set of the tape cache was about 200% (peak around 300%) of the actual tape cache size. We are responding to this by upgrading our tape cache with double-density drives and extended-length tapes. When complete, the tape cache will approximately quadruple in size.

Temporal locality

Working sets give us the big picture as to the overall locality of the workload within a specified time window, but they do not give us a distribution of how locality varies by file size. To determine the working set of an MSS cache, we had to maintain state about every file written to or read from MSS-III. This made it easy to generate a distribution by file size of the probability of a file being read after it had been read or written previously, and if read, the average time interval between the successive requests. To get a rough metric that combined both the probability of reuse and the interarrival time, we plotted the former divided by the latter versus the file size in megabytes. High probability of reuse is good, but only if the interarrival time is fairly short; if files of size 10 megabytes have a 99% probability of being reused, but requests for those files occur 30 days apart, caching all 10 megabyte files for 5 days would not be useful.

Figure 7 shows the simple locality metric for files grouped according to the log of their size in kilobytes. Highest locality occurs for files of ones of megabytes in size. Taking the broad range of file sizes in MSS-III into account, from kilobytes to hundreds of megabytes, in general smaller files exhibit higher locality (that is, a higher probability of reuse).

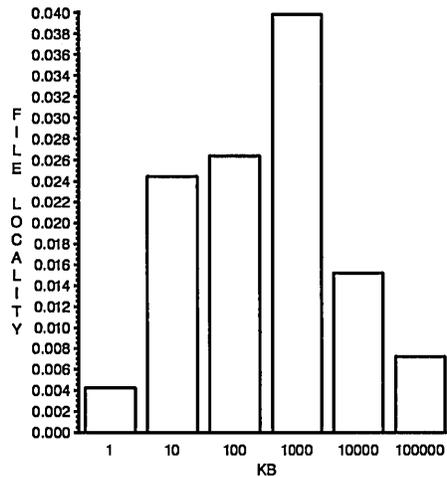


Figure 7. Locality by log of file size in kilobytes

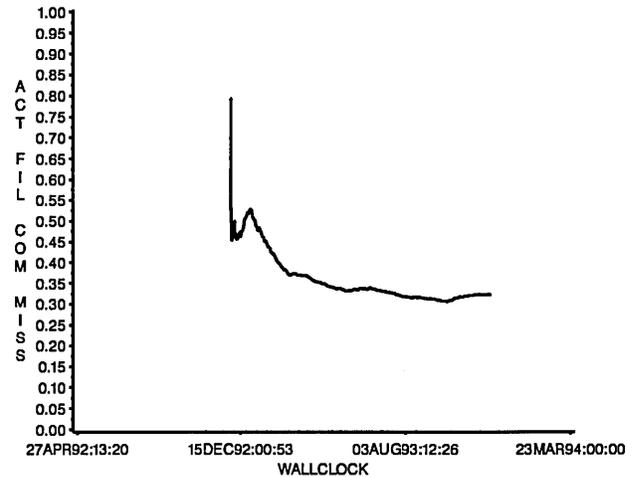


Figure 8. Actual compulsory miss ratio for tape cache

Cache performance

The simple locality metric is useful for comparing the relative locality of files, but it does not provide a useful unit with which to make decisions about MSS configuration or policy. A (perhaps obvious) insight was to measure the effectiveness of the MSS-III caches using traditional cache performance metrics such as the hit, capacity miss, and compulsory miss ratios. The hit ratio would tell us the probability of finding a needed file in the cache. The capacity miss ratio would tell us the probability of a cached file being dropped from the cache then re-referenced at a later time, and so is a measure of adequacy of the cache size. The compulsory miss ratio would tell us the probability of a file being referenced that had never been in the cache before, and so is a measure of the temporal locality of the given workload. These three probabilities should add up to 100%, since a file is either in the cache, was in the cache, or has never been in the cache. Since we were already keeping state about each file read from or written to the MSS, adding the ability to track the actual MSS caching of the file was easily done.

Figure 8 shows how the actual compulsory miss ratio for the 1.2 terabyte tape cache changed over the December-November time period. Overall, during this time, the tape cache had an actual 57% file hit ratio, a 10% capacity file miss ratio, and a 32% compulsory file miss ratio. The compulsory miss ratio is especially troubling since it is a function of low temporal locality in the workload, and cannot be easily addressed.

Cache simulation

Working sets and cache metrics provide lots of good information about how adequate the caches are sized, and how effective they are. But they do not provide any predictive capability, for example, what would happen if we increased the size of the tape cache. Intuitively it would seem to be a good thing, but the known compulsory miss ratio made us wary of the likelihood of diminishing returns.

By using the transaction logs as traces for a cache simulation, we can model portions of the MSS for a specific input workload. Some differences from the usual hardware cache simulation are necessary, for example cache lines in MSS-III are variable length (the size of individual files), and when a cache line is dirtied, the entire cache line rather than a single cache word is replaced.

Figure 9 shows how the predicted compulsory miss ratio for an upgraded 4.8 terabyte tape cache changes over the December-November time period.

There are a number of caveats we discovered in simulating the MSS-III caches. Many results are simply artifacts of the input workload. For example, our cache model never predicts more aggregate I/O bandwidth between the MSS and its clients than actually exists on the computer room floor. This is because the

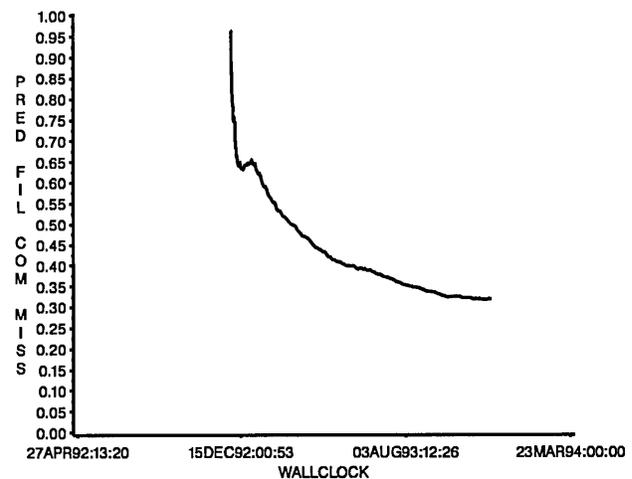


Figure 9. Predicted compulsory miss ratio for tape cache

traces with the associated inter-arrival time of requests which drive the simulation are derived from the actual workload. Also, an enormous amount of work must be done to validate the simulation, otherwise we would have no confidence in its predictions. Other problems with trace-driven simulations can be found in the literature (Jain [7]).

Conclusions

It may look as if we want to discredit virtually every computationally simple metric that might be commonly used to measure MSS performance. In practice, we generate all of these metrics, and they all have their place. However, the sheer volume of MSS transactions and their bursty nature forces us to step back and try to characterize the overall workload in a manner which will give us understandable metrics in usable units which we can apply to configuration, design and policy decisions. We still do not believe that we have a good handle on a metric to measure the burstiness of the workload, and are currently interested in methods applied to characterizing network traffic (Leland [8]).

This has been a challenging, and incomplete, task. In fact, the situation may be worse than we portray: at NCAR, the performance of our production supercomputers and our mass storage system is so closely coupled that it is difficult to understand one without analyzing the other. Work is underway to model the supercomputer workload at NCAR (Colarelli [9]). There is likely to be an advantage in coordinating the two efforts.

More detailed information about workload characterization and performance analysis of the NCAR Mass Storage System III can be found in a related paper, Sloan [10].

Acknowledgments

The National Center for Atmospheric Research is operated by the University Corporation for Atmospheric Research under the sponsorship of the National Science Foundation. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author and do not necessarily represent the views of the National Science Foundation.

The author would like to express his appreciation for the ongoing support of his management for this project. Specifically, thanks go to Gene Harano, Bernie O'Lear, and Bill Buzbee, all of NCAR.

References

1. Harano, E., "Data Handling in the NCAR MSS", National Center for Atmospheric Research, July 1993
2. Ousterhout, J. *et al.*, "A Trace-Driven Analysis of the UNIX 4.2 BSD File System", *Proc. of the Tenth ACM Symp. on Operating Systems Principles*, December 1985
3. Miller, E., "Input/Output Behavior of Supercomputer Applications", UCB/CSD 91/616, University of California, January 1991
4. Miller, E. *et al.*, "Analyzing the I/O Behavior of Supercomputer Applications", *Proc. Eleventh IEEE Symp. on Mass Storage Systems*, Monterey CA, 1991
5. Miller, E. *et al.*, "An Analysis of File Migration in a Unix Supercomputing Environment", extended abstract, U. of California, Berkeley, 1992
6. Denning, P., "The Working Set Model for Program Behavior", *Proceedings of the ACM Symp. on Operating Systems Principles*, October 1967
7. Jain, R., *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, New York, 1991
8. Leland, W. *et al.*, "On the Self-Similar Nature of Ethernet Traffic", *ACM SIGCOMM '93 Proc.*, San Francisco CA, 1993
9. Colarelli, D., "Modeling the Network Queuing System", National Center for Atmospheric Research, November 1993
10. Sloan, J., "Flying with Instruments: Characterizing the NCAR MSS-III Workload", to appear in *Proc. Thirteenth IEEE Symp. on Mass Storage Systems*, Annecy, France, June 1994

SCIENTIFIC DATA STORAGE SOLUTIONS: MEETING THE HIGH-PERFORMANCE CHALLENGE

Daniel Krantz, Lynn Jones, Lynn Kluegel, Cheryl Ramsey, and William Collins
Los Alamos National Laboratory
Los Alamos, New Mexico

The Los Alamos High-Performance Data System (HPDS) has been developed to meet data storage and data access requirements of Grand Challenge and National Security problems running in a high-performance computing environment. HPDS is a fourth-generation data storage system in which storage devices are directly connected to a network, data is transferred directly between client machines and storage devices, and software distributed on workstations provides system management and control capabilities. Essential to the success of HPDS is the ability to effectively use HIPPI networks and HIPPI-attached storage devices for high-speed data transfer. This paper focuses on the performance of the HPDS storage systems in a Cray Supercomputer environment.

INTRODUCTION

The Los Alamos Computing, Information, and Communications Division has developed HPDS to meet the high-end data storage and data access requirements of the Los Alamos computing network. Major computational resources include seven CRAY computers that have a total of 39 CPUs, two fully configured Thinking Machines CM-200s, a 1K Thinking Machines CM-5, and thousands of workstations. A network based on the 100-megabyte-per-second HIPPI (High-Performance Parallel Interface) technology [1] interconnects HPDS, computational machines, and visualization systems.

HPDS Requirements

The principal clients of HPDS are Grand Challenge and National Security problems running on massively parallel machines and large-memory supercomputers. These problems generate gigabytes to terabytes of output that are stored on HPDS as sequential files, and data is often appended to the end of files. This output is accessed by post-processing systems that perform calculations such as time and space averaging and store the processed data as megabyte- to gigabyte-size files. Visualization systems access the output and post-processed data randomly or sequentially at some stride. To meet these requirements, HPDS must be able to store petabytes of data, transfer files at tens of megabytes per second, handle terabyte-size files, and provide selective access to the data (i.e., transfer partial files). Metadata and scientific data management systems and methods are being investigated as possible means for facilitating data storage and access. These

systems will provide users with a convenient and powerful interface to HPDS.

A Fourth-Generation Data Storage System Model

Traditional network data storage systems use a "front-end" computer that provides network connectivity for storage devices along with storage management, device management, and data transfer capabilities [2, 3, 4]. This approach can result in a solid, flexible system; however, transferring data through a front-end machine can make the data storage system very expensive and limit the performance. To meet high-end data storage and data handling requirements with a traditional system, a large mainframe or small supercomputer must be used for the front-end.

An alternate method is to directly attach storage devices to the network and transfer data directly between storage devices and client machines. Higher data transfer rates and reduced hardware costs are realized by this method, which allows more powerful data storage systems to be implemented to meet the high-end requirements. The term "fourth-generation" has been given to this type of data storage system in which data is transferred directly between storage devices and client machines, and workstation-class machines are used for control and management. HPDS at Los Alamos is a fourth-generation data storage system; NCAR [5] and the National Storage Laboratory at Livermore [6] are also developing fourth-generation systems. Major differences between HPDS and other fourth-generation systems being developed are: HPDS uses a peer-to-peer protocol for the HIPPI data transfer; and the HIPPI connection is used exclusively for data packets.

The recent availability of HIPPI network technology and HIPPI-attached storage devices has generated widespread interest in this new generation of data storage systems, especially for high-end applications [7, 8]. HIPPI network switching components are commercially available, most supercomputers have HIPPI interfaces, and scientific workstations are beginning to have HIPPI interfaces (e.g., IBM RS/6000). HIPPI-attached RAID-3 and RAID-5 disk arrays are now available from Maximum Strategy and IBM. Maximum Strategy has modified its HIPPI controller to work with the Ampex/E-Systems DD-2 helical-scan tape recorder. Other companies (e.g., Sony, and Broadcast

Television Systems) are pursuing HIPPI-attached DD-1 helical-scan tape systems.

HPDS IMPLEMENTATION

The current production version of HPDS has been operating for nine months in the Los Alamos High Performance Computing Research Center (HPCRC). During this time, substantial work has been done to enhance overall reliability, performance, and usability.

Major components of HPDS implementation are shown in Figure 1. The DD-2 tape storage system is currently under development, and plans are being made to implement the Gateway system. All other HPDS subsystems are fully operational. Each HPDS component is designed to run on a separate UNIX workstation-class machine. In practice, however, several components are run on the same

workstation. All components communicate using fixed-format XDR messages delivered over a UNIX stream socket.

The current implementation does not provide any automatic data migration and caching. Clients are required to explicitly specify the storage system and to move files (or parts of files) between storage systems. Besides the need to provide a capability to meet immediate requirements, there is the need to gain experience with the storage and access of gigabyte- and terabyte-size files before implementing migration and caching strategies.

The IEEE Mass Storage Reference Model [9] was used for system design, and the Reference Model terminology is used in the remainder of this document.

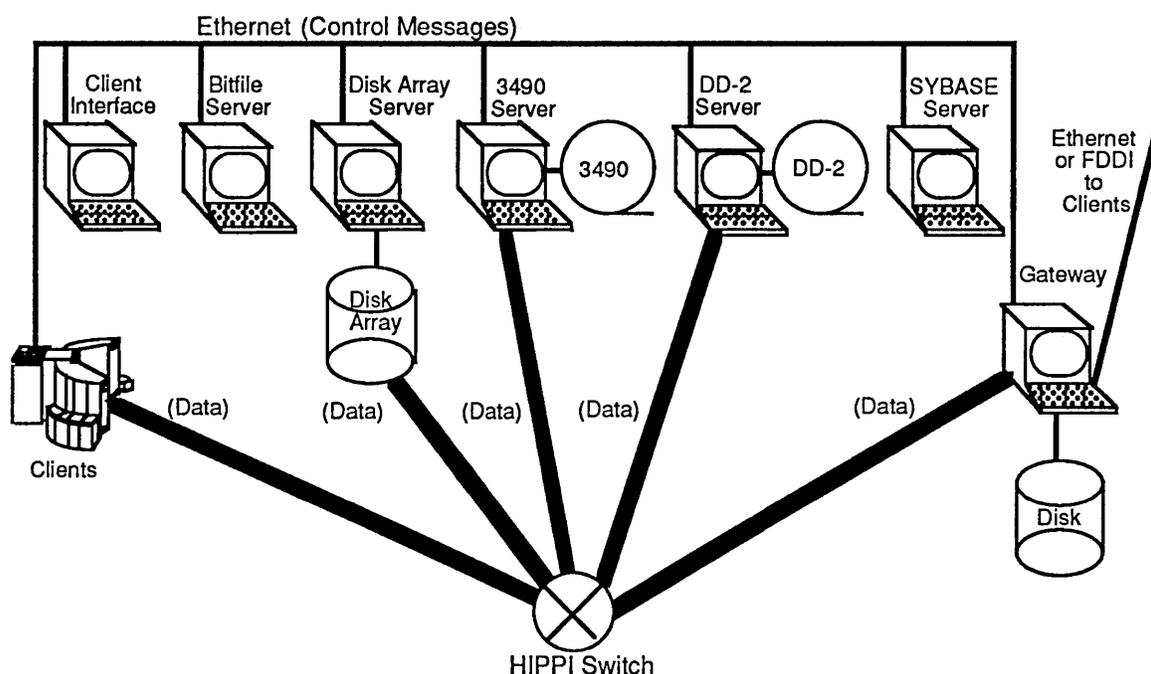


Figure 1. HPDS implementation.

Client Interface/Name Server

Currently, the client interface and the name server are integrated in a single subsystem. The initial client interface is called Data Transfer Interface (DTI). DTI is based on the Internet File Transport Protocol (FTP) code, which provides a familiar data transfer interface and offers a way of controlling client access. The file transfer commands and UNIX directory commands use FTP syntax. DTI differs from FTP in that partial file transfers and file appends are allowed. DTI also has commands to send data from HPDS to display systems and to transfer

data between HPDS storage systems. From the client requests, DTI generates requests for the name server and the bitfile server.

The name server maps the client UNIX file system path name to a bitfile_id. The current name server was implemented using a UNIX file system in which each validated user has a home directory. DTI commands that list files and manage directories are handled by the name server within its file system. The name server associates

the `bitfile_id` with the UNIX file name by writing the `bitfile_id` as data in the named file.

An intelligent user interface has been developed to create a UNIX-like environment with both interactive capabilities and C- and Fortran-callable routines. The UNIX commands and routines are translated into requests that are then processed by DTI. The high-level interface provides caching, batch programming, metadata management, and file association through chunking (i.e., family of files).

Bitfile Server

The bitfile server processes requests from the client interface to create and delete bitfiles, to query and modify bitfile attributes, and to store and retrieve bitfile data. The HPDS implementation uses a single bitfile server to manage multiple storage servers, which enables the bitfile server to be a location server for storage servers. The bitfile server determines which storage system to use for data retrieval and data storage. The design allows for data to be directly stored and retrieved from any of the storage systems and allows for files and partial files to be cached on the disk array storage system.

Disk Array Storage System

The disk array storage system is implemented using an IBM 9570 HIPPI-attached RAID-5 disk array and an IBM RS/6000 workstation. The 9570 has a maximum data transfer rate of 60 megabytes per second and a claimed storage capacity of 58.1 gigabytes (expandable to 232 gigabytes). The RS/6000 workstation is used to implement the disk array server, which maps logical storage to physical storage and provides commands to allocate and deallocate space and to store and retrieve data. The disk array server also includes the device driver for the disk array and implementation of the data transfer protocol (data mover).

The bitfile server sends requests to store and retrieve data to the disk array server, which establishes the data transfer protocol with the client machine and issues read and write commands to the disk array through an Ethernet connection. Data is transferred directly between the disk array and the client machine via a HIPPI connection; no data is transferred through the workstation. Disk files may be randomly read, written, and updated as long as no holes (areas without valid data) are created.

The disk array storage system is used to provide speed matching between fast client machines and slower tape storage systems, sustained data transfer with streaming tape systems, random access to data by caching files and

partial files, and high-speed data transfer to visualization systems such as frame buffers.

3490 Tape Storage System

The 3490 tape storage system is implemented using an IBM 3490 tape subsystem, which consists of two tape controllers and four tape drives, and an IBM RS/6000 workstation. The 3490 tape subsystem has a storage capacity of 400 megabytes per cartridge (800 megabytes for double-length 3490E cartridges) and a data transfer rate of three megabytes per second.

The RS/6000 workstation is used to connect the 3490 tape subsystem to the HIPPI network and to implement the 3490 server, which maps logical storage to physical storage and provides commands to allocate and deallocate space and to store and retrieve data. The 3490 server also includes the device driver for the 3490 tape subsystem and implementation of the data transfer protocol (data mover). The bitfile server sends requests to store and retrieve data to the 3490 server, which establishes the data transfer protocol with the client machine and issues read and write commands to the 3490 tape subsystem. All data passes through the workstation.

Tape files may be randomly read and may be appended but may not be updated. Tape files may be directly stored and retrieved without being cached on disk.

DD-2 Tape Storage System

A tape storage system based on the Ampex DST-600 helical-scan drive is under development. The DST-600 has a storage capacity of 25 gigabytes per cartridge and a data transfer rate of 15 megabytes per second. As with the 3490 tape system, an IBM RS/6000 workstation is used to connect the DST-600 to the HIPPI network and to provide storage management, device driver, and data transfer capabilities. All data passes through the workstation. The DST-600 is a streaming device and will be used to hold large files (greater than a gigabyte), while the 3490 system will then be used for smaller files.

SYBASE Server

The SYBASE Relational Database Management System (RDBMS) is installed on a workstation and is used by the bitfile server and storage servers to store and access system tables. Use of an RDBMS for this function offers several advantages. A large part of any data storage system is the software needed to insure the integrity of the system tables when hardware failures and system crashes occur. A commercial RDBMS provides capabilities to backup, journal, and restore data tables, to maintain consistency across updates to multiple tables, and to lock and unlock tables. The SYBASE SQL interface offers all

HPDS system components, including system management (operator interface, etc.), a powerful means of accessing system tables.

HIPPI Network Connection

The IBM RS/6000 workstation provides a good way to connect storage systems to the HIPPI network. The RS/6000 can be configured with two 80-megabyte-per-second micro-channels and up to one-gigabyte memory, has interfaces to a variety of high-performance storage systems, and has a HIPPI interface that does outboard protocol processing for TCP/IP, IPI-3, and HIPPI-Framing Protocol (FP). A throughput of over 60 megabytes per second has been demonstrated for an RS/6000-970 using the HIPPI IPI-3 protocol. Data was read into an application program on a HIPPI interface connected to one micro-channel and then output on a HIPPI interface connected to the other micro-channel. An RS/6000-970 should be able to support simultaneous data transfers for three Ampex DST-600 tape drives (15 megabytes per second) or eight IBM 3490 tape drives (3 to 8 megabytes per second).

HIPPI Data Transfer

Reliable, high-speed data transfer in a HIPPI network is important for HPDS, computational machines, and visualization systems. A data transfer protocol for the Los Alamos HIPPI network must provide routing, flow control, reliable data delivery, process identification, and peer-to-peer connectivity. The protocol must not substantially degrade the performance of the physical HIPPI network and must be supported by the client computers, storage systems, and other components of the network. HIPPI Data Transfer (HDT) was developed by Los Alamos to meet these requirements[10].

HDT is based on the separation of control and data, where control uses TCP socket connections for message communication (currently over Ethernet), and data packets are transmitted over a HIPPI connection. This separation of control and data allows for reliable delivery of control messages, at the same time allowing large packets of data (megabytes) to be transferred over the HIPPI with minimum overhead. The protocol provides flow control, block-level retransmission, and timeouts.

HDT is implemented as a library of C routines that can be loaded and called by application programs. The source or destination of an HDT transfer can be memory, disk, or any storage to which the application has access. HDT can also be implemented as a standalone utility program. HDT allows the starting data transfer address and the length of the data transfer to be specified, which can result

in partial file transfers. Data may also be appended to the end of a file.

HDT consists of system-independent routines and system-dependent routines. System-independent routines implement control protocol using TCP sockets for communication. System-dependent routines interface to the HIPPI driver and with local storage (e.g., disk or memory) to read and write data packets. The protocol is based on receiving side routines requesting data packets from sending side routines.

The most difficult HIPPI data transfer challenge was to match the HIPPI implementations of the various systems: the RS/6000 supports TCP/IP, IPI-3, and HIPPI-FP protocols; the 9570 disk array supports IPI-3 protocol, use of header-less data packets (data packets with no header and no protocol), and use of alternate-header data packets; the Cray UNICOS machines support TCP/IP and IPI-3 protocols, use of header-less data packets, and use of HIPPI-FP; and the Thinking Machines CM-5 and CM-200 only support header-less data packets.

Presently, all HIPPI data transfers between Cray clients and HPDS storage systems use HIPPI-FP while header-less data packets are transferred between the Connection Machines and the HPDS disk array. The separation of control and data, where all control is done using TCP sockets over Ethernet, allows header-less data packets to be transmitted over the HIPPI network. Use of header-less data packets means that the destination must be able to determine from the control path what each data packet is as it arrives, which prohibits multiplexing of data transfers and could cause data integrity problems. The internal HIPPI interface for the Connection Machine (available by mid-1994) will be able to support data packets using HIPPI-FP; then, all HIPPI data transfers will use HIPPI-FP.

HPDS PERFORMANCE

Theoretical performance statistics are rarely achieved in real-world systems due to high-overhead costs and unpredictable operating environments. The following paragraphs present actual HPDS storage system performance numbers gathered in a true production environment from a variety of HPDS clients.

Major HPDS client architectures consist of Cray supercomputers and massively parallel Connection Machines from Thinking Machines Corporation. Performance statistics were gathered for a CRAY X-MP, CRAY Y-MP, CRAY M98 and CM-5 using DTI. The 1K CM-5 is supported by a multi-gigabyte Scalable Disk

Array (SDA) and an external HIPPI connection. Table 1 shows the different Cray configurations.

Disk Array Storage System Performance

Although the disk array has a potential capacity of 58.1 gigabytes, the HPDS implementation can only use 48.06 gigabytes. Of the 20 available disks in the array, 16 are used for data, 2 for parity, and 2 for hot spares. This configuration increases the reliability and availability, and it avoids the penalty of having to read parity data before each write of new data. The disk array has a potential transfer rate of 60 megabytes per second, but several

factors limit this number: type of transaction, transfer block size, file size, and HPDS client architecture. Experience has shown that writing data to the disk array has a slower transfer rate than reading data from the disk array. Therefore, all performance numbers shown in the following figures will differentiate between these transfers.

Transfer block size has a significant impact on disk array performance. Figure 2 shows curves derived from a HIPPI tester that wrote data to and retrieved data from the disk array in raw mode, bypassing all HPDS protocols and layers of overhead.

Machine	UNICOS Version	Number of CPUs	Memory Size and Type	IOS Model - Disk Type	ldcache Size and Location
CRAY X-MP	7.0.5	4	16-megaword CMOS	C - DD49	384-megabyte SSD
CRAY Y-MP	6.1	8	64-megaword CMOS	D - DD42	366-megabyte SSD
CRAY M98	7.C.3	8	16-gigabyte DRAM	E - DA62 with 4-way stripe	1.75-gigabyte main memory

Table 1. Cray Client Configurations.

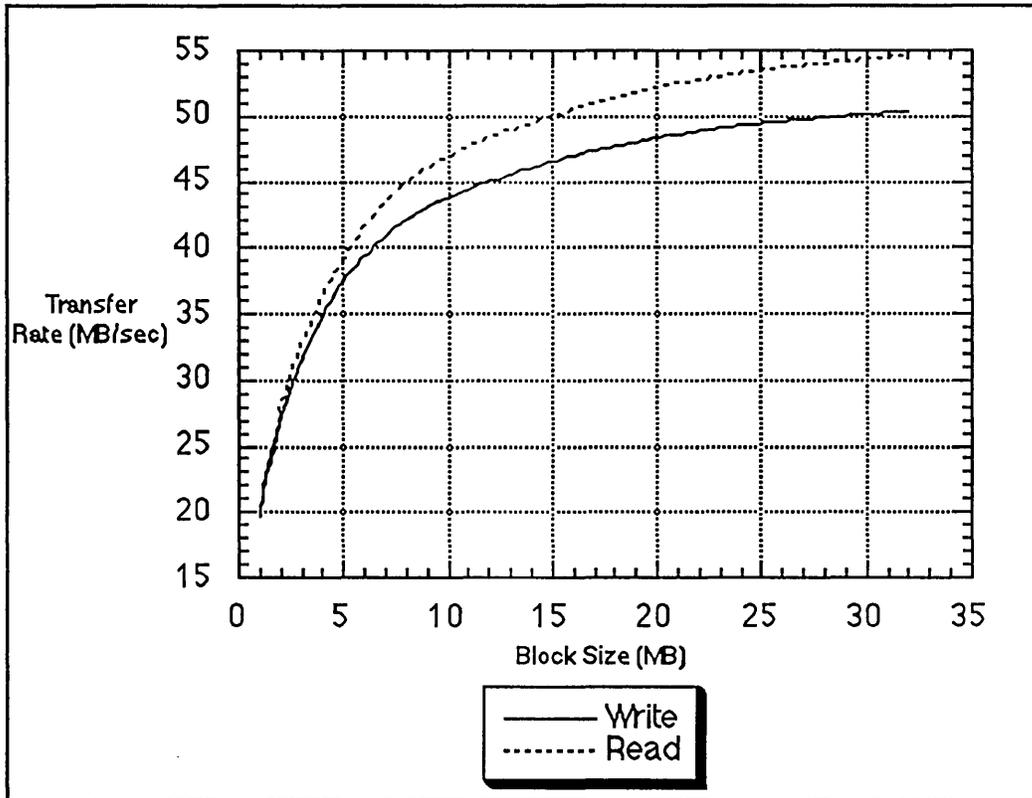


Figure 2. Raw transfers using a HIPPI tester without HPDS overhead.

As the transfer block size increases, actual data transfer rates approach the theoretical limit of 60 megabytes per second. Although it appears that 32 megabytes should be chosen as the transfer block size, HPDS operates with a 4-megabyte block size because of HPDS client constraints.

Multiple transfers between the disk array and HPDS clients were conducted with files of random data ranging in size from 2 megabytes to 1.5 gigabytes to examine HPDS disk array performance from a practical standpoint. Each machine was in production use, allowing contention for the HIPPI channel, ldcache, memory, and CPU. The CRAY M98 was also run under dedicated system time (DST) to demonstrate full HPDS capability without user contention on the HPDS client. Figures 3 and 4 show transfer rates for the range of file sizes under a constant transfer block size of 4 megabytes.

Transfer rates for the CRAY X-MP and CRAY Y-MP drop drastically between 200 and 400 megabytes, which is due,

in large part, to the ldcache, which falls within this range. Transfers less than the size of the ldcache will achieve SSD or main memory speeds. Once the file size exceeds the ldcache, transfer rates become limited by the Cray disks. The CM-5 has a low, but consistent, transfer rate because of the CM-5 external HIPPI interface. The CRAY M98 is able to sustain high transfer rates throughout the range of file sizes for two reasons: the disks can sustain transfer rates of 128 megabytes per second, and the main memory ldcache is greater than the largest file in the test suite.

These performance statistics were gathered while HPDS maintained data transfer block sizes of 4 megabytes. To demonstrate the drop in performance with smaller block sizes, the same suite of tests was repeated from the CRAY M98 for block sizes of 1, 2, 4 and 32 megabytes. Figures 5 and 6 show the results.

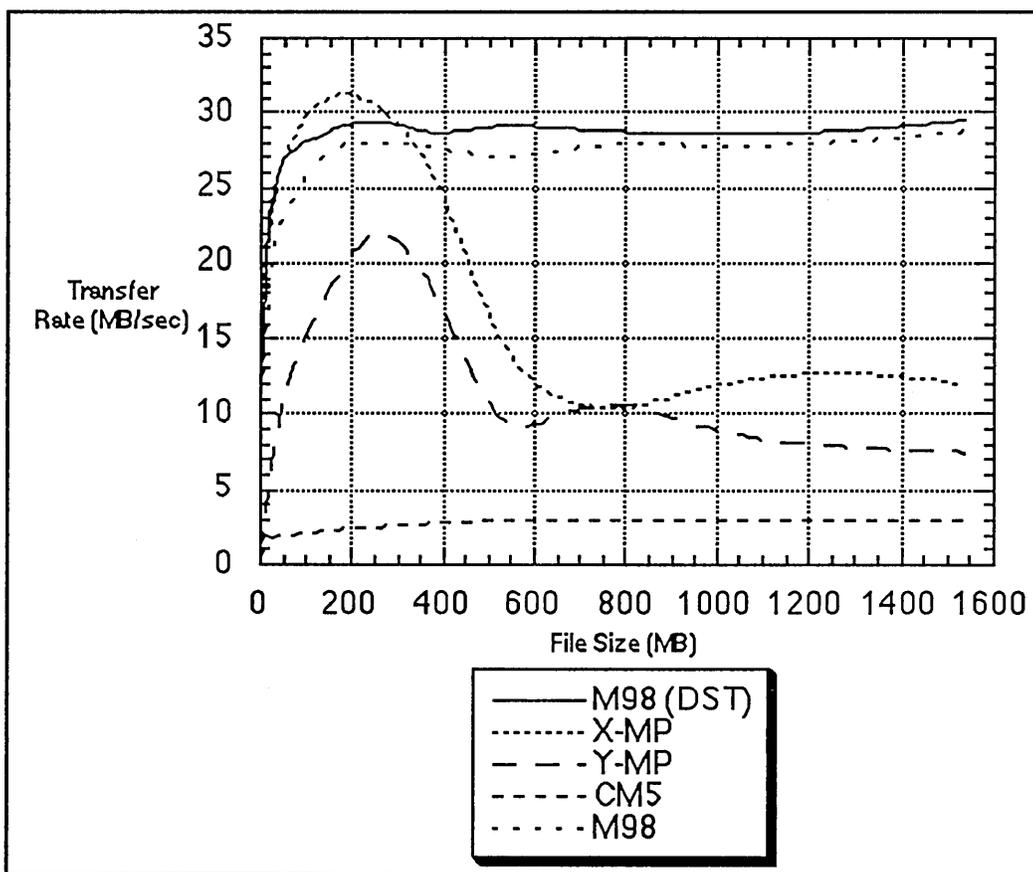


Figure 3. Transfer rates for reading data from the disk array onto the HPDS client.

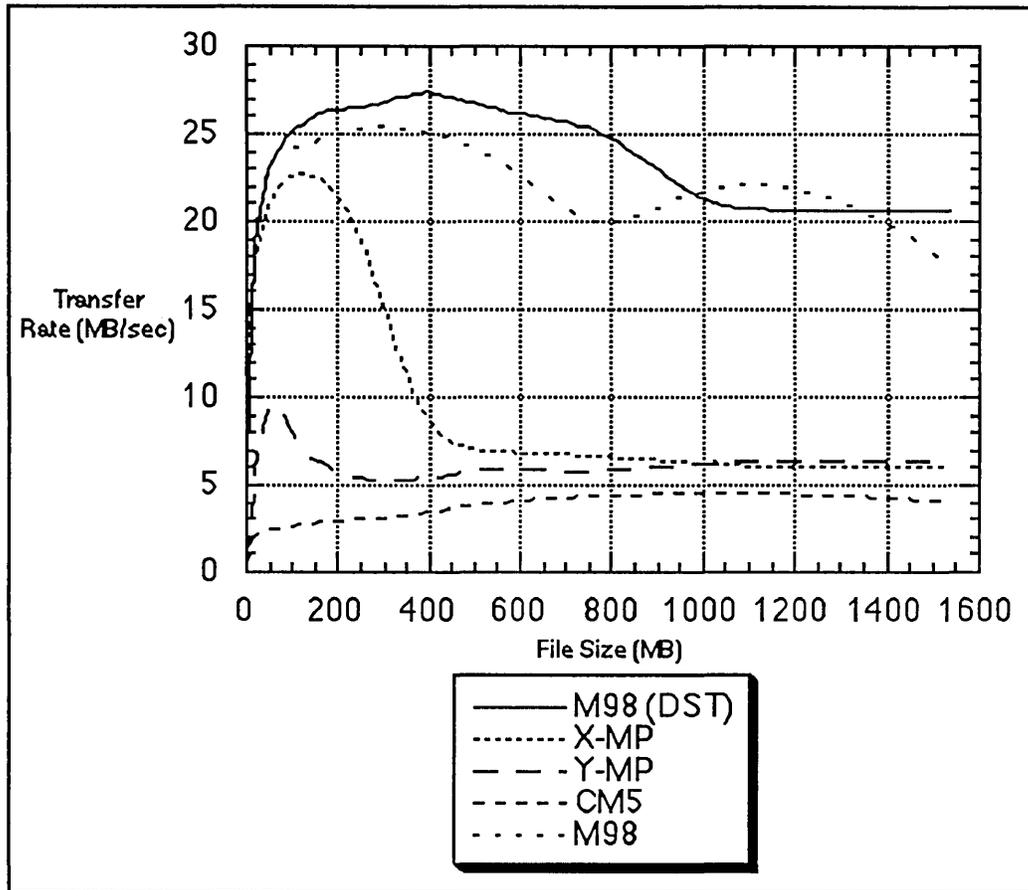


Figure 4. Transfer rates for writing data from the HPDS client to the Disk Array.

Clearly, larger transfer blocks yield better performance statistics, but other HPDS client architectures must be taken into consideration. The Cray M98 performs well with 32 megabyte blocks because it is a large machine with very low user demands. On heavily used systems, the 32-megabyte transfer blocks introduce a significant delay in CPU scheduling, lowering transfer rates below the 4 megabyte block performance levels. To accommodate the wide spectrum of machines, the transfer block size is currently maintained at 4 megabytes.

3490 Tape Storage System Performance

The 3490 tape storage system is currently connected to an RS/6000 using a parallel channel interface, which limits the data transfer between the 3490 tape system and the workstation to 2 megabytes per second. Tests similar to the disk array performance analysis were conducted on the 3490 tape system. In almost all cases, the transfer rates averaged 1.5 megabytes per second. As would be expected, the transfer rates were slightly lower for the larger files because of multiple tape mounts. Improved transfer rates are anticipated for the ESCON interface, a DMA device with an expected transfer rate of 17

megabytes per second. With a 2 to 1 compression ratio, we expect to achieve 6 megabytes per second transfers to individual tape units. HPDS user data achieves an average compression ratio of 1.9 to 1, and visualization data can sustain a compression ratio of nearly 3 to 1.

DD-2 Tape Storage System Performance

Preliminary tests of the DD-2 tape storage system, which is under development, indicate the obvious: the theoretical transfer rate of 15 megabytes per second will not be achieved. Again, the transfer block size plays a key role in determining the transfer rates. The Ampex DST-600 supports transfer block sizes up to 1 megabyte, and when this maximum is used, data can be moved between RS/6000 memory and DD-2 helical scan tape at 10 to 12 megabytes per second. Transfer block sizes less than 1 megabyte drastically affect this rate and will not be used by HPDS. A production DD-2 tape storage capability should be available by mid-1994.

FUTURE WORK

Plans for enhancing HPDS include: acquiring a large-capacity, high-performance automated tape storage system based on advanced linear technology or helical-scan technology to replace the existing 3490 tape storage system as the principal storage system; implementing migration programs to automatically move files down in the storage hierarchy when they are idle and to cache files in better-performing storage systems to improve data access; implementing a Gateway System to provide clients with standard FTP and NFS access to HPDS data using Ethernet and FDDI connections (HPDS data will be cached on the Gateway system); implementing additional scientific data management systems to provide users with more intelligent access to their data; and supporting the CRAY T3D computer as a client.

SUMMARY

Data storage and data access requirements of a high-performance computing environment can be met in a cost-effective manner by using a fourth-generation data storage system in which storage devices are directly connected to a HIPPI network and are managed by workstation-class machines. Vendor and user organizations, including Los Alamos, are now building these fourth-generation data storage systems using a variety of approaches and software.

HPDS is specifically aimed at providing Grand Challenge and National Security computational problems with the ability to store and retrieve gigabyte to terabyte amounts of data promptly. Transfer block sizes, transaction type, and client architecture significantly affect HPDS performance. This analysis demonstrates the crucial need for efficient client HIPPI connectivity.

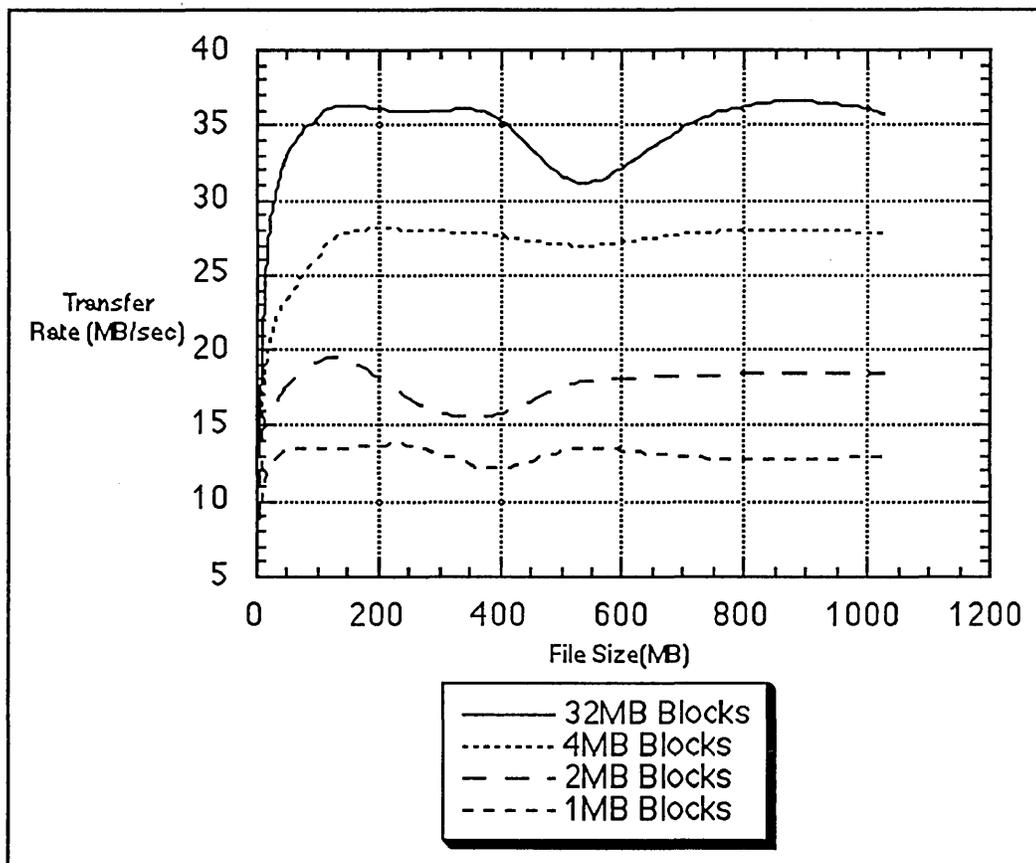


Figure 5. Transfer rates for reading data from the disk array to the Cray M98.

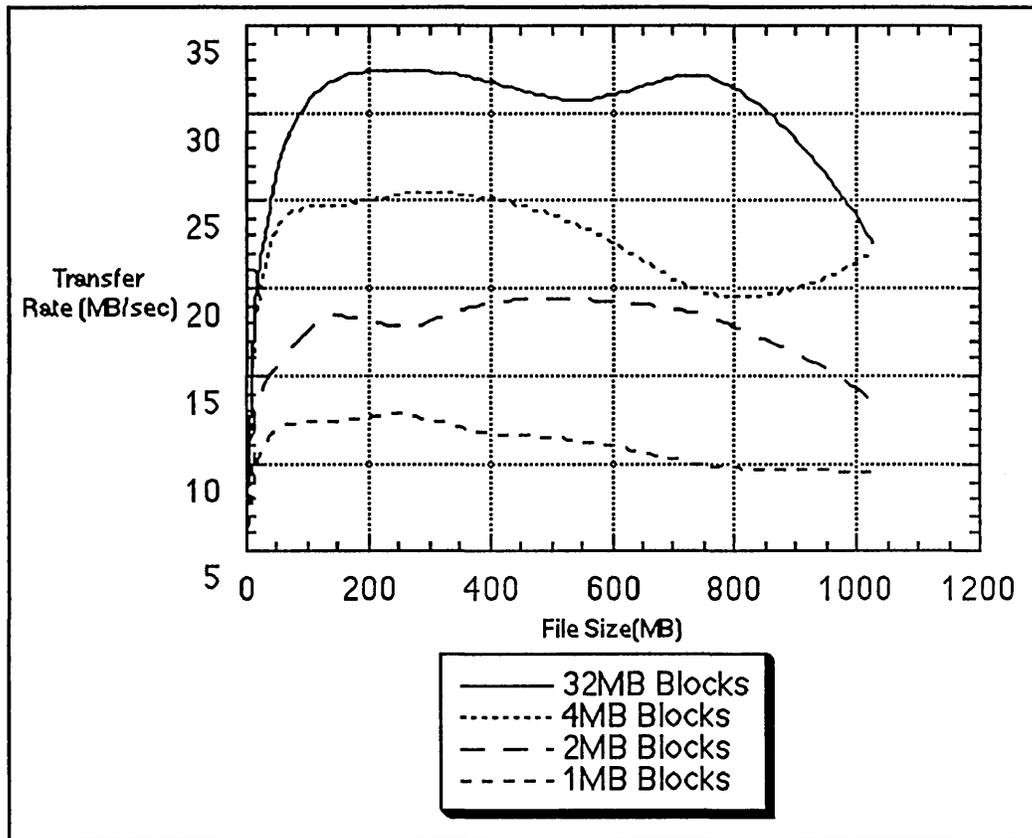


Figure 6. Transfer rates for writing from the Cray M98 to the disk array.

REFERENCES

1. "High-Performance Parallel Interface Mechanical, Electrical, and Signaling Protocol Specification (HIPPI-PH)," ANSI X3.183-1991.
2. Hogan, C., et al., "The Livermore Distributed Storage System: Requirements and Overview," Digest of Papers, Tenth IEEE Symposium on Mass Storage Systems, May 1990, pp. 6-17.
3. Tweten, D., "Hiding Mass Storage Under UNIX: NASA's MSS-11 Architecture," Digest of Papers, Tenth IEEE Symposium on Mass Storage Systems, May 1990, pp. 140-145.
4. Peterson, A., "E-Systems Modular Automated Storage System (EMASS) Software Functionality," Digest of Papers, Eleventh IEEE Symposium on Mass Storage Systems, October 1991, pp. 73-76.
5. Sloan, V., et al., "MASSIVE: The Mass Storage System IV Enterprise," Twelfth IEEE Symposium on Mass Storage Systems, April 1993.
6. Coyne, R., Hulen, H., and Watson, R., "Storage Systems for National Information Assets," Proceedings of Supercomputing 92, November 1992, pp. 626-635.
7. Tolmie, D., "Local Area Gigabit Networking," Digest of Papers, Eleventh IEEE Symposium on Mass Storage Systems, October 1991, pp. 11-16.
8. Collins, B., "High-Performance Data Systems," Digest of Papers, Eleventh IEEE Symposium on Mass Storage Systems, October 1991, pp. 25-26.
9. Coleman, S., and Miller, S., "Mass Storage Reference Model: Version 4," Goddard Conference on Mass Storage Systems and Technologies, September 1992, pp. 1-76.
10. Collins, W., et al., "Los Alamos HPDS: High-Speed Data Transfer," Twelfth IEEE Symposium on Mass Storage Systems, April 1993, pp. 111-118.

TRADEMARKS

Connection Machine, CM-200, and CM-5 are registered trademarks of Thinking Machines Corporation.

CRAY, Y-MP, and UNICOS are registered trademarks of Cray Research, Inc.

Ethernet is a registered trademark of Xerox Corporation.

IBM and RS/6000 are registered trademarks of International Business Machines Corporation.

NFS is a registered trademark of Sun Microsystems, Inc.

SYBASE is a registered trademark of Sybase, Inc.

UNIX is a registered trademark of AT&T.

COPYRIGHT

Copyright, 1993, The Regents of the University of California. This document was produced under a U.S. Government contract (W-7405-ENG-36) by the Los Alamos National Laboratory, which is operated by the University of California for the U.S. Department of Energy. The U.S. Government is licensed to use, reproduce, and distribute this document. Permission is granted to the public to copy and use this document without charge, provided that this notice and any statement of authorship are reproduced on all copies. Neither the Government nor the University makes any warranty, express or implied, or assumes any liability or responsibility for the use of this document.

All Los Alamos computers, computing systems, and their associated communications systems are to be used only for official business. The Computing and Communications Division and the Operational Security and Safeguards Division have the responsibility and the authority to periodically audit users' files.

BEYOND A TERABYTE FILESYSTEM

Alan K. Powers

Sterling Software

Numerical Aerodynamic Simulation Facility

NASA Ames Division

Moffett Field, CA USA

powers@nas.nasa.gov

Abstract

The Numerical Aerodynamics Simulation Facility's (NAS) CRAY C916/1024 accesses a "virtual" on-line filesystem, which is expanding beyond a terabyte of information. This paper will present the evolution of the Data Migration Facility (DMF) at NAS and some options for fine tuning DMF to stretch the on-line disk capacity and explore the transitions to STK 4490 devices.

NAS Mission

The National Aeronautics and Space Administration (NASA) created the Numerical Aerodynamic Simulation (NAS) Program to focus resources on solving critical problems in aerospace, space technology, and related applications by utilizing the power of the most advanced supercomputers available. 1

The mission of the NAS Program is to ensure continuing leadership in Computational Fluid Dynamics (CFD) and related computational aerospace disciplines by:

- acting as a pathfinder in advanced, large-scale computational capability through systematic incorporation of state-of-the-art improvements in computer hardware and software technologies;
- providing a national computational capability, available to NASA, DoD, industry, other government agencies, and universities, as a necessary element in ensuring continuing leadership in computational fluid dynamics and related computational aerospace disciplines;
- creating a strong research tool for the NASA Office of Aeronautics.

NAS C90 Configuration

The NAS Program is currently administering a CRAY C916/1024 (C90) that accesses a "virtual" on-line filesystem. The C90 contains 16 CPUs and 1024 MW of main memory, one solid-state storage device (SSD), and five I/O

clusters. Three communication channel adapters connect the Operator's Work Station (OWS) and two FDDI rings. Four HiPPI channel adapters connect to an UltraNet hub, two Maximum Strategy Raids, and a HiPPI "switched network." Four tape channel adapters connect four cross-coupled StorageTek (STK) 4490 control units with a total of 16 STK 4490s tape transports inside two Library Storage Modules (LSM, aka SILO), and eight IBM 3490E manual tape drives.

The C90 is using a beta version UNICOS (8.0.1) and DMF (2.0.4), and manages 315,000 files using 130 GB of on-line DD42 and DD60 disk and 1.1 TB of off-line tape usage and 5.5 TB silo capacity. The total disk capacity is 263 GB for various system and test filesystems and two temporary filesystems, /big (18 GB using 1.75 GB of ldcache), and /fast (5.5 GB of SSD) and five home filesystems (25 GB total).

NAS Process

Most everyone has heard the saying "Pascal assumes the programmer doesn't know what he is doing, C assumes the programmer does know what he is doing". Ada assumes the programmer doesn't know what he is doing, but if he thinks he does he needs to get approval. As in most organizations assume a change needs to be approved, therefore change is a slowly evolving process.

Originally NAS used DMF with a Cray 2 (navier) and a CRAY YMP (reynolds) to transfer files reliably to the mass

storage system (prandtl). This was done primarily because network transfers used a slow unreliable network connecting to an unstable machine. By using DMF within NQS jobs, job failures decreased, but prandtl was still causing major problems due to long periods of down time.

The NAS model on the Cray systems was to have three different types of user disk storage: home, scratch, and *TMPDIR*. The home filesystem was the login directory to hold user source code and support files for the application. The scratch filesystems were short term storage (3-4 days) to hold job output that would be needed for a subsequent job. *TMPDIRs* would be used for NQS jobs to create intermediate files that would be removed at the end of the session.

With this model both the home and scratch space were over subscribed by 200% and 400% respectively. This had not previously been a problem, because all 1000+ customers were on the system at varying times throughout the operational year and did not overlap enough to cause a space problem.

It was thought that the users would run their NQS jobs in the *TMPDIR* filesystem then copy the 'needed' output files to the scratch area, because these were the fastest disks using *ldcache*. Files needed for the long term were transferred (*rcp/ftp/dmput/dmget*) to prandtl. This model sounds great, but in reality customers could not count on prandtl being up to get the data back, so the users started hoarding the scratch space to run their NQS jobs. Disk space was like gold, those

who had it ruled. Those who didn't had problems with jobs aborting due to a lack of scratch space to save job results.

When NQS jobs abort, the Cray cpu cycles are wasted and no useful research is accomplished. If just five percent of the time on the YMP was lost this constituted 3500 cpu hours and if the problem continued with the C90 (arriving in March, 1993), it would be wasting 7000 hours. It would be like wasting one full year of computing time for a Cray 2.

Different alternatives were investigated to help resolve the problem of jobs aborting due to the lack of scratch space. The best alternative at the time was to attach STK silos directly to the YMP and build a filesystem for users that DMF could migrate. In the summer of 1992, NAS decided the YMP was only for computation and it was expected the problems would be solved "soon" because of the system upgrade on prandtl.

"The system will always be defended by those countless people who have enough intellect to defend but not quite enough to innovate. Politically, change forced by a crisis is much more acceptable because it is obvious that something must be done - and surviving a crisis is achievement enough." 2

In January 1993 the crisis occurred. After the upgraded mass storage server had been up and running for a little over a month, it had a major operating system failure and was down for several days. At this time NAS decided they

TABLE 1. Daily tape pool (msp) report

DATE	New Tapes	Put Files	Total Put GB	Average Put GB	Get Files	Total Get GB	Average Get GB
930912	97	40640	21.83	0.00054	70	0.908	0.01297
931101	155	1414	31.39	0.02220	7124	36.967	0.00519
931103	263	3250	50.60	0.01557	6151	37.330	0.00607
931205	379	8056	78.78	0.00978	54	0.431	0.00798
940201	27	377	11.27	0.02988	633	3.100	0.00490
940202	29	904	12.53	0.01386	1779	10.412	0.00585
940203	32	1148	13.59	0.01184	515	2.739	0.00532
940204	34	931	13.66	0.01468	434	8.848	0.02039
940205	30	1411	12.96	0.00918	162	1.968	0.01215
940206	26	726	10.71	0.01475	484	2.395	0.00495
940207	11	694	5.05	0.00727	834	5.802	0.00696

had to attach the silos to the YMP because the Cray 2 was leaving in February. In middle of January, a massive effort commenced: tape channels were installed and connection to the silos. DMF and silo software were installed and tested. By February the silos were fully attached and DMF in production. In the middle of February about 70 gigabytes of DD42s were added to the YMP for DMF migration for the purpose of being the "interim" mass storage server.

" Success is when opportunity meets preparedness."

Confucius paraphrased

DMF worked well for the YMP customers and DMF storage net growth was over 200 gigabytes a month. During this time several tools (**silorelist**, **silostat**, **siloload**, **silobject**, **silorecycle**, **dmf_stats**) were developed to help manage the DMF tape pools. The report in Figure 1, showing the current status of a DMF tape pool, was created by **silostat**. The report in Table 1, showing the daily usage of the tape pool, was created by **dmf_stats**.

When the C90 finally arrived in the middle of March 1993 (into production in April) the filesystems were configured the same as previous Crays with home, scratch and **TMP-DIR** filesystems. Although the scratch space was doubled for the C90 and DMF on the YMP was reliable, most users still hoarded their scratch space. After several months this became a serious problem again. Also NAS researchers needed to run jobs producing substantial output files (10 GB and greater), but there was not enough free space.

The NAS model needed to change. The easiest solution was to reduce the users' disk quota, but the customers needed the existing disk quota to complete their projects, but not all the time. NAS wanted the new mass storage system to replace DMF on the YMP, but it was not ready when the YMP was to be decommissioned in October of 1993.

FIGURE 1. A snapshot of the status of a msp.

```
carf1 status:
tapes to be loaded into silo = 2767
empty tapes to be loaded into silo = 2767
partial tapes to be loaded into silo = 0
empty silo tapes = 2444 (2097 available for use)
used silo tapes = 2486
empty db tapes = 5231 (2097 available for use)
used db tapes = 2486
total db usage = 921.041401 GB
last tape in db = N68299
tapes to merge = 2 ( 0.309855 GB) in 1 merge operations
Tapes to be ejected from lsms: 2431
Tapes to be ejected from lsm0: 1116
```

Tapes to be ejected from lsm1: 1315

In August 1993 DMF and the migrated filesystem were moved to the C90 and configured. DMF was to be read-only on the YMP, read/write on the C90, and the silos were to be shared between the two machines. A modification was made to **dmpu** to allow migration of files on a specified filesystem. By this time DMF was managing 1.3 terabytes of data.

In late September 1993 the C90 memory was upgraded to 1024 megawords. Extra swap disk space was needed and the only place to take space was from scratch. The NAS model changed to combine all the scratch space into migratable space (130 GB). This typically provided 30 GB of free space. Researchers could now create the large output files they needed, and jobs would not abort due to full filesystems.

This created a different set of problems to solve. Before the change, DMF was migrating 200-500 files a day, and afterwards, 2000-6000 files a day for a total of 30 GB of new data per day. This created a lot of tape shuffling in and out of the silos.

To help manage the growth, NAS wanted to enforce a 10 GB limit for on-line and off-line space, and a maximum of 750 inodes per user. This limit was to go into effect by the middle of November. In response customers tried to bring their files back at once, but this created large delays and clogged the I/O paths to the migratable filesystem.

DMF modifications as shown in Figures 3 and 4 were added so only a small number of files could be migrated during the day. This helped avoid "put" storms, but users wanted small files and the files they were using to stay on-line. A DMF attribute file was created to keep files less than a day old or less than 100 KB on-line, and the DMF configuration file parameter **MIN_DM_SIZE** was changed from 4 kilobytes to 100 KB.

The reason 100 KB cut-off was chosen is shown in Figure 2. These files represented 65 percent of the total number of DMFfiles and less than 1 percent of the total data. Retaining these files on-line also reduced the number of files in the DMF database. In a few weeks, DMF data went from 1.7 TB and 450,000 files to 0.8 TB and 300,000 files.

"Success covers a multitude of blunders"

George Bernard Shaw

FIGURE 2. File Size Report

SIZE	FILES	%	MBYTES	%
0K	4671	1	0	0
0-4K	86481	26	119	0
4-50K	116861	35	1486	0
50-100K	12179	3	870	0
100-500K	29602	9	7145	0
500K-1M	16378	4	12119	1
1-50M	57840	17	409621	42
50-500M	4186	1	445118	46

In the middle of November, the new mass storage server came into production. The C90 migratable filesystem switched from an interim mass storage server to short term storage for jobs. Files needing long term could be transferred to the mass storage server. The file age report (Table 2), shows the data is spread out fairly evenly over time. In the future this report may be used to justify moving the older data off the C90 to the mass storage server

FIGURE 3. New options for *dmmctl*, *dmhit*, *dmfree*

-b	migrate new files only i.e. no dual state files
-m filecount	migrate filecount number of files
-i blockp	migrate a percentage of the total migratable blocks

FIGURE 4. Keep files less than 1 day and less than 100KB on-line.

Attribute	Startpoint	Constant	Variable
access	0	-1	0
access	1	1	.01
access	2	10	.1
access	3	100	1.
access	4	1000	1.
size	0	-2000	0
size	25	0	0

In December 1993, the tape transport and control units were upgraded to support the new STK 4490s. While benchmarking the STK 4490s, several one GB files were written to a 3490E tape using a blocksize of 64 KBytes; the transfer rate was about 3.5 MB/s. Compare this to 2.6 MB/s for the previous STK 4480s. Using the *ebmxmon* on the OWS, the best transfer rate to date for the STK 4490 is 3.8 MB/s. The upgrade to STK 4490 has provided 30-40 percent increase in transfer rates. Unfortunately, the current version of DMF and DMF 2.1 assumes the tape to be fixed length, neither version can take advantage of the data compression.

Dump was used to compare the tape usage and time between the IBM 3480s and 3490Es. The 3490E are functionally equivalent to the STK 4490s. The size of the filesystem to be dumped was 18.5 Gbytes, at 93% capacity. The actual data dumped was 17.2 Gbytes. Figure 5 shows an 85% (6.5 to 1) decrease in the number of tapes needed for both 3480 and 3490E tapes using the new IBM 3490E drives, and 66% (3 to 1) decrease in time. If DMF could take advantage of data compression, it would probably increase the 3490E tape capacity from 1000 MB to over 1500 MB.

FIGURE 5. Transfer time and usage comparison for drives and tapes

IBM	3490E tapes	3480 tapes	3480 tapes & drive
Tapes	12 tapes	20 tapes	75-80 tapes
Time	1 hr 40 min	2 hr 56 min	4-5 hours

In January 1994, the C90 starting to run out of processes because zombie processes were not being removed. It was determined that *init* could not completely sync the disk in a timely matter and this caused a backlog of disk request on the migrated disk. Also during this time, several customer jobs, using the *assign -s u*, degraded I/O on all jobs. Several customers were requested to change the *assign* statement. Then the migrated disk were reconfigured with the first couple of DD60s to handle all the inode lookup on the migrated filesystem. After this was done the zombie problem went a way.

Lessons Learned

Try to have a VSN naming convention for the msp tape pool (0-499 primary tapes 500-999 secondary tapes).

Have a backup recovery plan in place, and know how long the recovery will take to do before there is a failure. It may not be best

TABLE 2. File age report

DAYS	TOTAL		RESTORED				
	FILES	%	MBYTES	%	FILES	%FIL	%TOT
0-7	32586	9	152215	15	31937	98	41
7-30	45255	13	176693	18	13091	28	16
30-60	24881	7	94812	9	4034	16	5
60-90	18223	5	65285	6	2620	14	3
90-120	20820	6	63453	6	2403	11	3
120-180	44898	13	137038	14	5182	11	6
180-240	32981	10	81790	8	4589	13	5
240-300	23566	7	76576	7	2254	9	2
300-365	19170	5	54290	5	2119	11	2
1-2yr	36700	11	48479	5	6034	16	7
2-3yr	16916	5	8847	0	2017	11	2
3-4yr	6990	2	627	0	678	9	0
>4yr	5300	1	369	0	281	5	0
TOTAL	328286		960473		77239		

Files accessed in the last day 10007 megabytes 85599.5

do a full filesystem restore. If inodes on the failed disk are not used, then the disk can be replaced and just the files on the disk can be restored. **3**

Avoid mixing disk types and disk allocation units. Cray has this working now, but it is one less potential problem to worry about.

Develop a policy for how long and how much the data should be kept on-line per user. The NAS goal was to keep all the primary tapes inside the silo.

Only use ldcache with the write-through option on migrated and DMF filesystems.

Use the fastest disk possible for the DMF databases.

Make sure the tape daemon has all the bug fixes installed. When the tape daemon had problems it appeared to the customers there were problems with DMF.

Make sure there is hardware maintenance on the SUN workstation that controls the silos. This is a single point of failure. If this machine is down then DMF is down.

DMF system load is not an impact on workload.

Outstanding Problems

Dumps on the migrated filesystems take too long. This could be resolved if most of the files were migrated and brought back every night. Some testing to do this has already been done, but has not been put into production. Also with DMF 2.1, using the command **dmfill** could resolve the problem.

The directory lost+found is limited to 4 kilobytes. This is a big problem on a large filesystem. When the system crashes and **fsck** is fixing the filesystem the lost+found directory fills up quickly. The system then has to be brought down, and the directory moved and recreated, before starting the next **fsck**. This is something that may need to be resolved by CRAY.

The **/.dmpre** area grew to 20,000 files, which the **dmdaemon** searches for a migrated file (**rm**, **dmput**, **dmget**, etc.). It was taking over 40 seconds to do an **ls > /dev/null**. This causes long delays for on-line migrated files to be used. Cray should not use directories to see if the data blocks are to be migrated. A possible solution is to set flags in the inode to see if the file's data should be migrated. Some of this is done now, but not all of it.

Archiving sometimes causes a large number of **dmputs** filling the DMF pipe (`/usr/dm/dmd.req.pipe`), so **dmgets** have to wait until the pipe is empty before the **dmdaemon** is aware to process the **dmget** request. The **dmgets** takes several hours before users are able to use the data even if the data is already on disk in the `/.dmpre` area. It would be great if the **dmgets** had their own pipe queue to the **dmdaemon** and it would round-robin between the primary pipe and the **dmget** pipe.

Future

Future plans are to install the STK clipper doors into the silos, which will need ACSLS V4.0 for the STK SUN server. Currently all DMF tapes are 3480s XL configured for 500 megabyte capacity. These tapes will be replaced with 3490E tapes and configured to at least 1000 megabyte capacity. The tmp filesystem size might have to increase to accommodate the larger tapes size when several tape merges are in progress.

Acknowledgments

Special thanks to Nicholas P. Cardo for his DMF knowledge and support tools and Julia S. Close for her DMF measurement tools.

This work was performed by Sterling Software at Numerical Aerodynamic Simulation Facility (Moffett Field, CA 94035-1000) under NASA Contract NAS2-13619.

All brand and product names are trademarks or registered trademarks of their respective holders.

References

- [1] NAS User Guide Chapter Volume 1, P. 1-4
- [2] Sam A. Falk Molisvich, "System Requirements for Support of Computational Chemistry." Thirty-First Semi-Annual Cray User Group Meeting, Montreux, Switzerland, March 1993, P. 418.
- [3] K.C. Matthews, "UNICOS 6.0 File System Recoverability", Twenty-seventh Semi-Annual Cray User Group Meeting, London Great Britain, April, 1991.

Operations/Environmental MIG

Overview of Projects, Porting, and Performance of Environmental Applications

**by Tony Meys
Earth and Environmental Sciences Group
Applications Department
Cray Research, Inc.**

I. Introduction to Environmental E&ES

The Earth & Environmental Sciences (E&ES) Group is a multidisciplinary group of technical analysts with expertise in geosciences. A segment of this group is a dedicated resource for supporting projects specifically related to environmental science. The core expertise of Environmental E&ES lies in the porting and optimization of global and regional weather models. Additionally, Environmental E&ES is beginning to investigate other applications of Cray supercomputers for the solution of environmental problems. These emerging areas of supercomputing problem solving include air pollution, groundwater studies, hydrology, and coupled modeling problems which explore the complex interaction of air, ocean, land, and biological impact.

E&ES is interested in learning more about unique applications requiring the power of supercomputing which CUG members may be contemplating. The possibility for joint projects between Cray and CUG members exists where novel science pushes the limits of high performance computing, especially where the problem solved has both theoretical and practical application. CUG members are

encouraged to discuss possible projects with E&ES using the contact information provided at the end of this paper.

Environmental E&ES is somewhat unique in comparison to other groups within the Applications Department of Cray. We do not usually support third-party vendor codes (although we are open to opportunities to do this) or Cray owned applications. What we do support are needs within Cray and among our customers which require combined expertise in environmental science and Cray computers. The goal of Environmental E&ES is to assist our customers in getting the best performance possible from their environmental applications.

II. What Kinds of Environmental Applications are Cray Supercomputer Users Running?

The following table summarizes some of the environmental applications which E&ES knows to be running on Cray platforms. It is by no means an exhaustive list and is meant only to be representative of the kinds of environmental problems being solved on Cray machines.

Copyright 1994, Cray Research, Inc., 655E
Lone Oak Park, Eagan, MN 55121, U.S.A.

Application	Comments
CCM2	NCAR Community Climate model
MM4 / MM5	NCAR
ARPS	Center for the Advanced Prediction of Storms (CAPS)
HIRLAM	Regional forecast model (mainly used by European scientists)
NMC Production	10 day GCM; NGM and Eta model for regional U.S. forecasts
ECMWF	10 day forecast model
SKYHI	Global circulation model (e.g., GFDL)
POP	Ocean circulation model (e.g., GFDL, LANL)
Proudman Model	Ocean circulation model (Proudman Oceanographic Institute)
RADM/AQM	Air pollution (SUNY at Albany; U.S. EPA)
UAM	Urban airshed model (used by U.S. EPA)
MODFLOW	Groundwater
BIGFLOW	Groundwater (Southwest Research)

TABLE 1: Environmental Applications which run on CRI platforms.

The remainder of this discussion will focus on several examples of performance tuning of environmental applications for CRI platforms. These examples have been chosen because they illustrate performance considerations on C-90 and T3D architectures, the two most recent additions to CRI's family of products at the mid to high-end of computer performance.

III. C-90 Performance on Global Circulation Models

Since the mid-1980s, CRI vector/parallel machines have been the leading computational engine for the integration of operational and research general circulation models. This trend has continued with the introduction of the C-90 series of machines. The following two examples show some of the performance characteristics E&ES has observed in the process of evaluating customer codes.

CCM2: As part of a recent performance study, Dr. Jim Hack (NCAR Scientist and leader of the Community Climate Model Core Group) ran a high resolution global climate simulation on a dedicated C90/16 supercomputer at CRI's corporate headquarters. The purpose of this experiment was to validate CCM2 performance on CRI's largest PVP configuration. Prior to making this run, CRI field analysts and E&ES consulted with the Core Group on a wide range of performance issues which included I/O optimization, parallelization techniques, system tuning, and providing a large system for an extended (3 day) dedicated session. The following table summarizes computer performance related results of the experiment.

Resolution	Y-MP/8	C-90
T42L18	41	12
T63L24	170	42
T106L30	835	185
Using Isotropic Grid Formulation		
T42L18	31	9
T63L24	123	31
T106L30	584	132
Using Isotropic Fully Semi-Lagrangian Formulation		
T42L18	25	7
T63L24	64	16
T106L30	250	58

**TABLE 2: CCM2
System Turnaround (wallclock seconds per simulated day)**

SKYHI: E&ES recently participated in a performance analysis of a version of SKYHI. The code, which is a global circulation model, was provided to E&ES by the Geophysical Fluid Dynamics Lab (GFDL). Some results are summarized below. Of particular interest in this tuning exercise was the impact of SSD utilization. Even with very large core memories, it still is very important that large environmental models retain an out-of-core solution option. The lower cost of SSD coupled with high communication bandwidth provide the most cost efficient solution for running multiple, large simulations in Y-MP and C-90 environments. This also leaves more core memory available on a shared, production system for other users. The result is increased overall site throughput with only a minimal performance degradation cost resulting from use of the SSD.

Some of the parallel/vector projects the Earth & Environmental Sciences Group of CRI

will be involved in this year will include performance tuning of environmental codes for entry-level supercomputers, porting and optimization of environmental applications which are not now widely used on supercomputers (e.g., in areas of hydrology, groundwater, pollution), and evaluation of CRI's next generation of PVP machines on selected atmospheric and ocean modelling problems.

M/C	PROC	SSD W/OPT	SPEED UP	IN-MEM W/OPT	SPEED UP
YMP	1	1450	1.0	1386	1.0
YMP	8	199	7.3	189	7.3
C90	1	598	1.0 (2.4)	557	1.0 (2.5)
C90	4	157	3.8 (9.2)	147	3.8 (9.4)
C90	8	83	7.2 (17.5)	77	7.2 (18.0)
C90	16	49	12.2 (29.6)	44	12.7 (31.5)

TABLE 3: SKYHI
N90 (100 Steps = 1.67 hour fcst.)

IV. Developing Environmental Codes for the T3D

Optimization techniques for the Cray T3D are significantly different when compared to strategies used for PVP machines. This, of course, is the expected result of a shift to a fast scalar architecture and the larger number of processing elements available for use by an application. It has been the author's experience that compilation of existing Fortran codes for execution on a T3D requires minimal, if any, modification of a code. One caveat to this observation is the need to transform Cray floating point initialization files to IEEE format. Simple conversion programs can be written which utilize UNICOS's "assign" command to make necessary conversions.

For those users which have already developed application codes which take advantage of multi-tasking, the initial PVP decomposition may be of value. Since global weather models have traditionally been decomposed along latitude strips, a substantial amount of work already exists along each of these bands. In a benchmark code adapted by the author, an effective port was made of a single-threaded global circulation model using the traditional method of decomposition by latitudes. This was done for 32 latitude "groups."

Additionally, pairs of processors were experimented with so that teams of two PE's could split spectral transform calculations. Admittedly, this is not the best way to parallelize this kind of a code, since it will not scale beyond the number of latitudes and involves significant inter-PE communication. Still, an important part of the porting "battle" may involve getting an existing application running on the T3D so that the developer can experiment with decomposition methods and begin work on optimization of routines which may not require extensive modification even after a complete domain decomposition.

Ultimately, the most successful T3D ports will require rethinking the parallel design of an application. Examples of codes which have undergone this kind of redesign include POP and SKYHI. POP is currently running on the T3D. A project is now underway at GFDL (Geophysics Fluid Dynamics Lab) to evaluate SKYHI on both C-90 and T3D platforms. E&ES is following both of these projects and providing consultation as requested by the developers and CRI field analysts.

Today's T3D Tuning Techniques:

As with any applications development environment, analysts are always interested

in finding ways to avoid "reinventing the wheel" and evaluating performance. To date, E&ES environmental analysts have found a communications "template," knowledge of single PE optimization techniques, and the performance tool Apprentice to be particularly useful.

A communications "template": Since a major applications development issue when creating new MPP programs is initialization of a communications network among PEs and the availability of higher level communication utilities which perform recurring functions, a communications template is an important tool. E&ES is currently experimenting with the initialization, communication, and "stencil" routines created by the developers of POP. Since these researchers paid particular attention to implementing communication routines in a way which is portable to a variety of MPP machines, their communication modules have provided an excellent starting point for the development and testing of codes. Basically, the communication modules provide PVM initialization, communication among PEs configured as subcomponents of a volume, and an assortment of basic mathematical/communication routines. The performance of some of these routines has been improved by 1) assembly language routines, and 2) use of the CRI get/put communication primitives.

Communication "templates" like those created for POP allow a developer to immediately get a code running and begin to focus on performance optimization of code which will run on each PE. E&ES strongly recommends that T3D analysts develop these kinds of routines, or ask us about the example communications template just discussed.

Single PE optimization techniques: The DEC Alpha is the processing power source of the Cray T3D. Like all other high-performance platforms, a bit of knowledge about the

architecture of the machine can have a substantial impact on programming results.

A few things to keep in mind include:

- data cache use and alignment
- loop unrolling
- data locality
- memory "pages"
- dual instruction issues
- inter-PE communication overhead

Several examples based on solution of partial differential equations found in environmental applications were discussed as part of the CUG presentation accompanying this paper. It is anticipated that some of the optimization considerations may become less important as improved versions of the compiling system for the T3D become available.

Use of Apprentice: Application developers familiar with Perfview and HPM on Cray Research PVP platforms will find Apprentice to be a useful performance analysis tool for providing similar kinds of performance information on the T3D. A brief discussion highlighting some of the features of Apprentice was included in the CUG talk which accompanied this paper.

CRI supercomputer users have begun a number of T3D porting and tuning projects for their environmental applications. The following table summarizes some of these activities, indicating whether the project is in a planning, porting, or performance tuning stage.

Application	Comments
POPS	Ocean model; GFDL, LANL, others; CRI is consulting with Dr. Chris Kerr; code runs on T3D and is being tuned
Proudman Ocean	Ocean model; CRI support of customer from Cray UK; code runs on the T3D
HIRLAM	Regional weather model; CRI support of customer from Cray UK; initial port to T3D has been made, development continues
SKYHI	Global circulation model; E&ES is consulting with Dr. Chris Kerr, GFDL; code has been domain decomposed for MPP and initial T3D port is in progress
ECMWF	10-day forecast model; Cray UK is assisting with porting
CCM2	E&ES and CRI field plan on providing NCAR with assistance in porting/tuning this year
Others	

TABLE 4: Summary of Activities and Status

E&ES anticipates a substantial increase in its support activities of T3D projects during the remainder of this year. CUG members with an interest in tuning and porting codes for Y-MP, C-90, or T3D machines are encouraged to contact E&ES to ask questions and share results.

For more information contact:

Tony Meys
E&ES Group, Applications Department
Cray Research, Inc.
655E Lone Oak Park
Eagan, MN 55121
U.S.A.

phone: (612) 683-3426
e-mail: meys@stratus.cray.com

T3D SN6004 is well, alive and computing

Martine Gigandet, Monique Patron, Francois Robin
Commissariat a l'Energie Atomique
Centre d'Etudes de Limeil-Valenton
94195 Villeneuve-Saint-Georges Cedex - France
gigandet@limeil.cea.fr

March 10, 1994

Abstract

This paper intends to review the different stages we went through, from the day we decided to install an MPP machine at our site with the willingness of making it contribute heavily to our production, to this very day.

Introduction

CRAY T3D SN6004, the third T3D machine delivered to the field, arrived at Limeil on december the 1st, exactly on schedule.

The process of installing an MPP machine at our site had begun about one year earlier...

1 The choice

By the end of 1992, our management decided to replace our CRAY XMP 216 by a more powerful computer with a newer technology and innovative architecture.

As most of large computing centers, we had to face the challenge of providing a stable, continuous production environment and bringing leading edge computing capabilities. We were already offering an heterogeneous computing architecture to our user base : moderately-

parallel scalar (i.e, servers and workstations) and moderately-parallel vector (i.e, traditional supercomputers) connected via a tiered networking structure.

We felt it was the right time to add a third component i.e., massively-parallel for four main reasons :

- There was no other way to get the processing power future supercodes will need.
- The MPP market was mature enough to offer a usable production environment.
- We had acquired some experience to efficiently use MPP systems for our applications by various experiments done on different machines (CM2, CM5, iPSC860, ...).
- New programming techniques had to be taught to more programmers in order to be ready over the next year or two.

The choice was a CRAY T3D which would be delivered by the end of 1993. It was guided by several considerations :

- CRAY's skills in architecture and technology,
- very fast inter-processors communication,

- programming models availability,
- easy integration in existing environment.

At that time, the option was a standalone system integrating CRAY YMP, IOC and MPP elements into a single package. The configuration included an MPP system with 128 Processing Elements and 1024 Mwords of distributed memory, two YMP cpus, 256 Mwords of YMP DRAM memory and two I/O clusters.

2 The M92 delivery

As we needed more computing power before the end of 93, CRI accepted to deliver a CRAY M92 in march 93 in replacement of our XMP-216. The M92 modules would be reinserted in the T3D chassis or removed in case of a new complete T3D delivery at the end of 93.

We knew this M92 had to be operated with UNICOS 7 in order to provide our users with the CRAY T3D emulator so that they would be able to execute and do some performance analysis on some MPP programs before the machine arrival. During february, we made the transition from UNICOS 6 to UNICOS 7 on our XMP and we validated our production under that system. The M92 was started in march and we made an agreement with CRI to get the T3D emulator prerelease.

3 The hardware option changes

Meantime, CRAY informed us that they would not go further with the standalone system option we had chosen, the so called SA128/2-256. They had made the decision to replace the CRAY host YMP M92 by a YMP 4E with 64 Mwords of memory.

As we needed larger global memory on the host, we converted our first option

into a tightly-coupled system option. That meant that by the end of 93, we would keep the M92 as it was and receive one more separate chassis housing the MPP PE's. The MPP I/O cluster would be installed in the M92 chassis. This option opened the possibility to upgrade our configuration up to 512 PE's sometimes.....

Last hardware change concerned the size of distributed memory. The 16 Mbits memory chips allowing the 8 Mwords local memory per PE would not be in bulk production by the end of 93. Consequently, the machine that would be shipped would be equipped with 2 Mwords of local memory per PE. An upgrade was scheduled by the end of 1st quarter 94. We made an agreement with CRAY to divide the acceptance process into two steps corresponding to the two sets of hardware delivery.

4 The acceptance program test suite

At the beginning of october, we started to build an acceptance program test suite. We had to do it only with our own programs because of the small number of applications available in the new MPP arena and those being submitted to licensing.

The program test suite was made of 7 sets of 19 different programs :

- CEA Benchmarks : 8 Fortran programs running in single processor mode including kernels, linear algebra and pieces of production codes.
- Communications kernels : 3 tests (C and Fortran) designed to check and measure communications primitives (send, receive, broadcast) for PVM and "get-put" operations.
- T3D/M92 data exchange test via files with data conversions (IEEE 64 bits, CRAY), PVM between T3D and M92 being not available at that time.

- Neutron transport using Monte-Carlo method : very few communications, the particles being processed independently by each processor.
- Boltzman equation for high altitude flows (rarefied gases) : needed a transition from PVM 2.4 to PVM 3 done via a locally developed library.
- Electromagnetism : 2 C codes using PVM 3, the first one dealing with electromagnetic fields, the second with dielectric constants and 1 Fortran code using "get-put" operations dealing with RCS.
- Linear algebra : 2 Fortran programs solving a linear system by conjugate gradient with diagonal preconditioner and generating heavy data exchange. One uses PVM 3, the other "get-put" operations.

Most of those programs were tested first with the emulator and second on a T3D prototype located in Chippewa by the end of november. From that very first contact with the T3D, we drew the following considerations :

- We had no problems with the compilation operations.
- The user and system cpu times consumed on the YMP host by compiling and linking for T3D were very high (respectively 500 and 29 seconds for a 28000 lines Fortran program instead of 91 and 1 when the same program is compiled for a YMP).
- A few T3D reboots were necessary, some of them being probably due to the prototype state of the machine. We noted that rebooting is easy and quick.
- We effectively used the command "mpstat"

(displays information about state and use of the machine) and the tools "totalview" (debugger) and "apprentice" (optimizer).

- Some programming errors not detected by the emulator had to be corrected.
- Some program modifications were necessary, main concerns being about time measurement accuracy, cache efficiency, PVM environment variables setting.
- All numerical results were as expected.

5 The finish before T3D arrival

During november, we finished the work allowing our M92 to be a T3D host system. It required to upgrade UNICOS to the 7C3 release and the OWS to the 7.0.6 release. T3D software (compilers, loader, libraries, μ -kernel for computing nodes, μ -kernel for I/O nodes, drivers, commands, ...) was installed. A M92 hardware modification (with regard to the LOSP and HISP channels linking T3D and M92) had also to be achieved so that the MPP connect/disconnect operations will never have adverse effects on M92 production.

6 T3D arrival and first contact

On december the 1st, CRAY T3D entered Limeil laboratory. Time was about 12:30 am. First hardware tests were running the evening of that very day. The next day, installation process started on normal conditions. First connection with M92 was realized on december the 4th. Cray people experienced five PE modules failures, one

I/O gateway failure and a weld problem in the cooling system causing a fluorinert leak on the HEU. T3D IOC was missing, but of course we could not intend to use it at that phase of T3D program development. T3D software finished to be installed on december the 8th. The machine was made available for our first testing on december the 13th.

6.1 The system tests

Before validating our program test suite, we wanted to check some system functionalities :

- Reboot MPP and verify it does not impact YMP normal operation.
- Configure administrative pools with different attributes.
- Modify the pools state (available, unavailable) and check there are no bad effects on applications already running.
- Run concurrently multiple applications and verify the partitions are correctly created and returned to the pools.
- Execute the same application in different pools and compare the results.
- Test the NQS support for MPP.
- Test administrative and user commands targeted for the T3D use.

All those minimum system functionalities worked satisfactorily.

6.2 The applications tests

During the final preparation of the acceptance run, we encountered the following problems :

- Our UNICOS system (as probably most of UNICOS systems) does not allow more than 64 opened files per process. Since each MPP application is controlled by one agent process using 9 file descriptors for its own needs, 55 is the maximum number of files one application can have opened at the same time. If, for example, the application is running in 128 PE's and every PE is trying to open one file, it will fail. The issue to allow more than 64 files open (change of UNICOS kernel OPEN-MAX parameter) is whether we need to recompile all the applications on the M92 host or not. CRAY's safe answer is "I'm not sure" !!! We encountered a problem related to this limitation : a local default user environment setting (the "FILEENV" variable) forced all the PE's to open the file describing the "assign" parameters; it caused the program abort for partitions exceeding 32 PE's.
- During high T3D workload, the configuration driver reallocated the PE's to new partitions too rapidly which caused a μ kernel panic.
- Some partitions shapes allocated were not cubic and led to the abort of the corresponding programs, some others (which were valid but wrapped around) induced the programs' blocking up.
- Program blocking occurred too when one PE was sent a great number of messages by the others at the same time.
- After some "normal" program aborts (due to programming or environment setting errors), the T3D had to be rebooted.

7 The first step acceptance

On december 20th, a new version of MAX (1.0.0.3 with some fixes) was installed and our M92 was rebooted with a new MPP configuration driver. These modifications corrected the problems affecting our acceptance program test suite (i.e, bad partitions shapes). So we decided to proceed to the acceptance process. The T3D was configured with one 128 PE's pool.

- First, the acceptance programs were executed successively with different sizes :

script	#PE's	elapsed time
arlene	128,64	11 mn
bench	1	5 mn 30 s
benchk	1	1 mn
bmsolver	4,8,16,32,64,128	7 mn 10 s
boltzmann	4, 8, 16, 32	26 mn 10 s
diagopt	16, 128*3, 64	9 mn 20 s
diagpvm	16, 128	5 mn
io	4, 8, 16, 32	50 s
mctran	128, 64, 32, 16, 8	12 mn
perf	2	15 s
pvmtest	2, 32, 64, 128	1 mn
renorm	16*2, 32*2, 64*3	27 mn

- In addition of executing successively all the acceptance programs, we had planned to load them all in bulk into the machine. Up to 8 programs ran simultaneously. During this phase, while two programs were using respectively 16 and 32 PE's, we noticed that a program requiring 64 PE's was waiting.

All results were correct and we signed on the first step acceptance notification.

8 The first performances results

All those early performance results have to be refined. Specially, we noticed that

changing some T3D software components led to variations in running times ranging from 5 to 20 % for the same application depending on its size.

8.1 Single processor

- Performances depend heavily on cache use.
- 5 to 20 Mflops are currently obtained with Fortran programs.
- Looks like 5 times slower than a YMP-M92 processor in vector mode.

8.2 Network

- With Fortran, data exchange rate between two near-by PE's are :
 - using PVM : 33 MB/s with 20 μ sec latency time.
 - using GET : 35 MB/s with 2 μ sec latency time.

Within a 128 PE's partition, the minimum data transfer rate is 20% lower.

- With assembly language, we verified reading a data (i.e, a cache line) in another PE's memory needs 100 to 150 cycles depending of the two processors' distance.

8.3 Applications

Because the goal of the acceptance was to check that the T3D was working as expected, measuring performances was not our primary concern.

However, it is possible to draw some preliminary conclusions :

- With an optimized program tailored for the T3D, it is possible to get really high performances. For example, an optimized RCS program runs at 6 Gflops for 128 PE's.

- For a communication intensive program, it is often not very difficult to switch from PVM to "get-put" operations (especially if the program is "well" written). The performance improvement obtained can be very interesting. For example, in a conjugate gradient program iterations loop, the communications are located in 2 subroutines (dot product and matrix vector multiplication). Rewriting those subroutines is straightforward and leads to 1 Gflops on 128 PE's instead of 200 Mflops with PVM.
 - For this kind of programs, the performances of the T3D are quite good compared to other MPP machines (SP1 or CM5) because the speed of the network hides the not so good performances of the nodes (due probably for a part to the lack of maturity of the compiler and the cache size and policy).
 - On a simple particle transport program (this kind of codes accounts for a significative amount of CPU time on our vector machines), the T3D with 128 PE's runs 18 times faster than a M92 in monoprocessor mode.
- of our programs could be relatively easily adapted. The main issues when working with CRAFT are the following :
- CRAFT is evolving and a number of restrictions (often made in the interest of runtime performance) are introduced.
 - CRI seems to be short in time. Will all parts of the model be implemented in first release?
 - Some programming choices are difficult to make since we don't know whether automatic compiler optimization will occur or not.
 - Our programming strategy is based on the emulator results. We noticed the emulator has not always the behavior a real T3D would have (for instance, it simulates remote accesses for obviously local data in some pieces of code).
 - Some features planned for CRAFT are not emulated (shared variables pointers).
 - YMP cp time consumed by the emulator is high :

times for a 2200 Fortran lines code with 344 directives are :

YMP-M92	1.33s
emu -L0 -F0 -p4	132s
emu -L0 -F0 -p8	225s
emu -L0 -F0 -p16	426s
emu -L0 -F0 -p32	814s

- What will be the emulator status in the next future?

9 CRAFT

CRAFT, the Cray global-address-space programming model, is not yet released. Still, we have put the emphasis on it since the beginning of our experiments by using the T3D emulator. We are very concerned by the CRAFT features which combine a global address space with data-sharing and work-sharing permitting programmers to use a high-level, implicit method of specifying parallel work (fortran 90 array syntax, for example) or a lower-level, more explicit method. Our experience with CRAFT suggests that many

10 Conclusion

The early presence of a CRAY T3D at our site represents a great challenge for us. From our first contact with the machine, we feel confident for the future : hardware

shows a great level of reliability and software has all the features for usable high sustained performances processing.

10.1 Current work

Until now, only a few programmers are working on T3D (about 10 people). More programmers will be involved when the size of the distributed memory will be upgraded and CRAFT released. Current work deals with :

- Getting a better knowledge of the machine in general.
- Estimating the T3D I/O impact on YMP-M92 when our applications will be in production.
- Building tools to monitor M92 and T3D as a production computing facility.
- Porting on CRAFT system a CFD code with domain decomposition implementing hydrodynamics and particle transport.
- Porting RCS codes needing the development of a complex parallel Cholesky solver not existing yet.
- Porting an Euler explicit hydrodynamics code using PVM.

10.2 Drawbacks and advantages

The two following subsections summarize our early evaluation of the weaknesses and the strengths of the T3D computer system.

- Drawbacks
 - Small memory size (until 2Q94)
 - Difficult memory distribution (system buffers - applications)
 - Hard hierarchical memory (registers, cache, DRAM, remote DRAM) considerations

- PVM only (until 2Q94)
- CRAFT uncertainties
- High compilation and linkage times on YMP-M92

Among all those items, the last one has the worst effect. The M92 is a production machine with a high workload leading to unacceptable elapsed times for our T3D developers.

- Advantages

- High hardware reliability (no failure since installation)
- Few software problems (a non-regression test suite is executed at each system or compiler change)
- Easy integration in existing computing environment
- Adequate applications changes paths
- Satisfactory tools
- Interesting early performances

The expected progresses both of T3D software optimizations and programmers' skills should perceptibly improve the performances. For those reasons, we think the T3D will respond to our needs.

System Administration Tasks and Operational Tools For the Cray T3D System

Susan J. Crawford

*Cray Research, Inc.
Eagan, Minnesota*

This paper provides a conceptual overview of system administrators' responsibilities and the administrative tasks and tools for configuration planning, reconfiguring, and monitoring a Cray T3D system. The paper describes implementation to date and future directions.

March 1994

1.0 Introduction

T3D system administration to date can best be described in three parts. The three areas of interest are:

- Background information about the T3D resources that an administrator must understand and manage
- A description of T3D administration capabilities today
- A view into the areas that will be enhanced during 1994

2.0 Background Information on T3D System Resources

When administrators prepare for the arrival of a Cray T3D system, they must understand the system resources available to the T3D system. There are six basic resources that require some study. Those are:

- PE memory
- Barrier network
- Routing tables
- Redundant nodes
- Cray Y-MP system central memory

- PEs (processing elements)

PE Memory

PE memory is available in two sizes, two million words per PE or eight million words per PE. A goal of the Cray Research system software developers is to allow users access to as much PE local memory as possible. UNICOS MAX supports both memory sizes. It is important that an administrator understand these basic ideas about PE memory but no further administrative control of PE memory is required.

Barrier Network

The barrier network is implemented as a hardware fabric separate from the PE network fabric. The UNICOS MAX OS (operating system) currently allocates two barriers to each application for use by the application. No further control of the barrier network is necessary by the administrator. However, the administrator should study the high level hardware design of the barrier network for two important reasons. First, it is important to understand that both barrier bits and PEs must be available before the OS can allocate resources. Secondly, in the event of a failure in a wire in the barrier network, the administrator should

be aware of the effect this reduction in the total number of barrier wires has on OS resource allocation.

Routing Tables

The administrator should study the default routing tables that are provided with the T3D system to understand how these routes satisfy that installation's application needs. In general, no routing table administration is required.

Redundant Nodes

An administrator can replace a failing node by mapping in a redundant node at the next T3D machine reboot. Until that node is mapped in, the system stays up but runs with fewer available PE resources. On a Cray T3D system, a node is a pair of PEs which share a switch within the Cray T3D system network. The administrator should understand the impact on the resource allocator when a node fails and the procedure that is used to map in a redundant node.

Cray Y-MP System Central Memory

Cray Y-MP system central memory is required by the parts of the UNICOS 8.0 operating system that control, configure, administer and process T3D system application system calls. After an application load is characterized at a T3D system site, the administrator can further tune Y-MP system central memory use. This paper does not address that kind of tuning.

PEs

Within the UNICOS MAX operating system, a resource allocator maintains a list of free/available resources, a list of resources that are in use, and a queue of pending applications that have been validated for access to the T3D system. These pending applications are waiting their turn for resources. By default, this queue is processed in FIFO order. The OS uses resource allocation algorithms to determine if the free PEs and barrier bits are aligned in the hardware such that they can be handed out to meet an application's requirements. These requirements, time limits, number of PEs and memory, are communicated to the OS via environment variables and a.out options. In general, the administrator must realize that:

- When resources are available, the OS resource allocator gives an application a power of two PEs in a rectangular array.
- In some cases even though the correct number of PEs may be available to meet the requirements of a pending application, those PEs may exist in a location in the hardware such that barrier bits are not available. In this case the application remains in the OS application queue.
- Interactive jobs are validated and directly enter the OS application queue. They are not currently coordinated with those applications that are submitted and validated by NQS as they flow through NQS queues to the OS application queue.

The collection of PEs in the machine allow large applications to run where large means great numbers of PEs or quite possibly all PEs in the machine. This PE collection also allows many small applications to run simultaneously when each application is using a small number of PEs. In this paper large jobs will mean applications which request a great number of PEs. Small jobs will mean applications which request a small number of PEs.

To best manage T3D system resources, experienced T3D system administrators have found that the resource of primary concern is the total number of PEs in the T3D system. The PEs must be managed such that the T3D system is as sharable as possible. Another way to view this is to think of T3D system job throughput. Even though administrators cannot directly control the time of day nor the order in which programmers decide to submit batch and interactive jobs, they can set up the PE usage to get the most work from the T3D system.

T3D system administration centers on PE administration and the other five resources discussed in this section of this paper are of more interest to the administrator when:

- an error has occurred within the system
- applications make read/write system calls that cause the high speed channel from the T3D system to the Y-MP system to operate at or near the maximum possible transfer rate

The rest of this paper addresses administration of PEs.

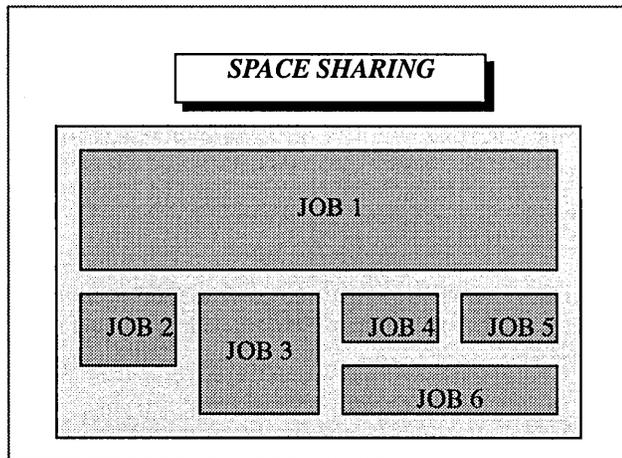
3.0 Background Information on OS Resource Allocation

The OS resource allocator was designed to meet the following goals:

- Allocate PEs to an application at run-time. These allocated PEs are referred to as a T3D system partition.
- Provide T3D system space sharing.
- Allocate a power of two PEs per partition.
- Support one PE applications.
- Gang schedule PEs, barriers and PE memory so that these resources are allocated and available at application start up and held until application completion.
- Allow application access to as much PE memory as possible.
- Allocate both a normal barrier and a eureka barrier to each application.
- Provide a heterogeneous application environment.

Space sharing means that when resources are available more than one partition can be allocated and active at a time. The following diagram illustrates space sharing.

FIGURE 1. T3D System Space Sharing



The UNICOS MAX OS supports the heterogeneous application environment in the following key ways. There is:

- Support for optimal application performance by allowing an application to be split between a Cray parallel

vector processor (PVP) front-end system and the Cray T3D MPP system.

- Support for optimal application performance by providing a high bandwidth, low latency connection between the PVP and MPP.
- Support of a common view or shared view of the file system for the heterogeneous application.
- Support for heterogeneous application system calls by distribution of the OS between the MPP and the PVP.

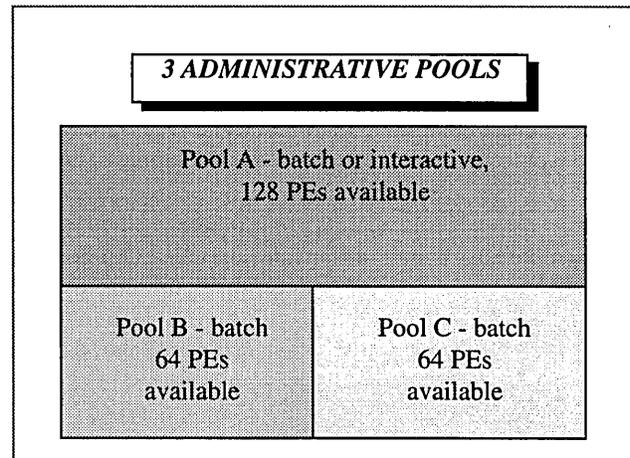
4.0 T3D System Administration Today

The focus for the administrator is T3D system shareability through the administration of PEs. T3D system PEs can be viewed in a hierarchical way:

- Level 1 - the whole machine.
- Level 2 - administrative pools of PEs within the whole machine.
- Level 3 - application partitions within the administrative pools.

The administrator considers dividing the whole machine into administrative pools of PEs. The following diagram illustrates one such administrative pool configuration. (See figure 2.)

FIGURE 2. Administrative Pools



As applications are assigned a partition and downloaded from the Cray PVP front-end machine to the Cray T3D system, sharing will occur. (See figure 1.)

5.0 Administrative Control Today

Management of T3D system PEs is through the use of administrative resource pools. PE resources are controlled on a pool basis. Finer granularity of control is achieved by the administrator through the use of four pool attributes supported by the UNICOS MAX OS today. These attributes are:

- pool type
- pool status
- pool privilege
- pool job flow

Pool type or usage can be batch, interactive or both. Pool status can be available or unavailable. The OS allocates resources only from pools marked as available. An administrator may wish to change pool attributes on demand. The procedure for doing this is to mark the pool as unavailable to allow active partitions to complete normally (or drain from the pool) and to prevent the OS from allocating additional pool resources. Once the pool is drained and quiet, the pool attributes can be changed and the pool once again set to available status.

Pool privileges can be limited by assigning group IDs to a pool. In the future, the OS will also support pool user IDs to further control pool access.

Pool job flow can be managed through the use of:

- the definition of an express job in terms of wall clock time
- the specification of maximum job wait/pend time
- the maximum number of partitions

To keep jobs flowing out of the OS application queue and into a T3D system pool, it is sometimes helpful to define the express job. The administrator could, for example, define express jobs to be those with one minute time limits or less. By doing so this causes the OS to allow a job of one minute or less to be given resources and downloaded to the T3D system even though it may not be at the head of the OS application queue. This would help most when an

administrator expects to see a mix of large/long jobs and small/short jobs waiting for T3D system access.

If express job processing is turned on as described in the paragraph above, maximum job wait could be used to further control the job skipping that will occur. Job skipping will occur if a large job rises to the head of the OS application queue but is then being bypassed for resource allocation due to a steady stream of express jobs space sharing the T3D system. By setting a maximum job wait time limit, the OS resource allocator turns off express job processing once the skipped job has waited the maximum time. The active T3D system applications are allowed to complete normally to sufficiently drain the pool and allow the skipped job to acquire the resources it needs.

The administrator can also specify the maximum number of partitions allowed per pool to further control pool job flow.

6.0 Administrative Monitoring

There are four commands that help the administrator monitor the use of T3D system PEs.

- *'mpping'*
- *'ps -m'*
- *'qstat -m'*
- *'mppstat'*

To ping the PEs on the T3D system, *'mpping'* is used. This command serves as a quick check to see if the PEs are responding. To display T3D application process information, the *'m'* option has been added to *'ps'*. The *'m'* option has been to *'qstat'* so status of batch queues for the T3D system can be retrieved. The *'mppstat'* command displays administrative pool status.

7.0 Future Development Work

Future UNICOS MAX OS enhancements fall in these categories:

- increased system shareability
- ease of use in scheduling
- ease of use in retrieving status

Shareability

There are currently some undesirable interactions between interactive job processing and NQS job processing since the two types of job submissions do not work in concert in the UNICOS MAX system today. There will be enhancements to the OS resource allocator to coordinate these types of job submissions.

There are a few situations that can occur when PEs and barriers are available for a partition but the resources are sufficiently fragmented such that the OS resource allocator cannot assign those resources. Again there will be enhancements to improve upon system shareability.

The third area that is undergoing development is a new feature called job roll. Job roll is often referred to as rolling or rollin/rollout. When a job roll out is requested by the administrator the UNICOS MAX OS will perform this sequence of steps:

- First, the to-be-rolled application is brought to a quiet point.
- Second, sufficient job state is saved so that the job can later be rolled back in to the T3D system.
- Third, the job roll image is moved off of the T3D system to a file.
- Fourth, the resources are returned to the resource allocator's free list.

Job roll is useful when the administrator wishes to temporarily stop a long running job so that those resources can be used by other short jobs.

When the administrator wishes to roll in a roll image, that job is located and placed in the application queue.

Scheduling

T3D system job scheduling can be made easier to understand and control. An administrative interface to scheduling can be provided through the use of a common scheduling command in the future. Also, there are utilities that have been developed for use in testing the UNICOS MAX OS that allow run-time reordering of the OS application queue. These utilities could be released with the software as options to a common scheduling command.

Status

In the future the 'mppstat' command can be enhanced to provide additional terse and verbose forms. This command can also be enhanced to serve as the common collection point for T3D system monitoring information.

CRAY Y-MP and UNICOS are trademarks of Cray Research, Inc.

Performance Evaluation/Applications and Algorithms

I/O Optimisation Techniques Under UNICOS

Neil Storer (*neil.storer@ecmwf.co.uk*)
European Centre For Medium-Range Weather Forecasts
Reading, Berkshire, RG2 9AX
England

1. Abstract

This paper will describe the techniques employed at ECMWF to optimise the performance of jobs which do large amounts of I/O. It will concentrate on the use of Flexible Fortran I/O (FFIO) by Fortran programs, though some mention will be made of techniques available to C programs.

The paper will also describe the I/O performance tools "proctstat", "procview" and "procrpt", which can be used to monitor I/O usage, indicating the files on which to concentrate tuning efforts.

2. Glossary

CACHE This is a buffer area, in main memory, designed as an intermediate repository for data blocks, positioned between the user's arrays and the physical disk. There are two reasons for the cache area. The first is to allow the user's logical I/O requests to be "batched up" into physical I/O requests to the disk. The second is to allow frequently accessed blocks of data to be made available at "memory speeds", without having to perform disk I/O.

CACHE PAGE This is a single cache buffer.

CONTIGUOUS When applied to disk space, this means that the individual sectors which make up the space on the disk are consecutively allocated. For example a 100 sector contiguous file starting at address "A" on the disk will use up all the sectors through to address "A + 99". It is much more efficient to do I/O to a contiguous data set than a non-contiguous one, especially if that file is LDCACHED. "setf" can be used to create a contiguous file, and the "/etc/fck" command can be used to confirm this.

COS This is an acronym for "Cray Operating System", an earlier operating system for Cray computers. It was a Cray proprietary system, not based upon Unix and although still supported by Cray Research, is used at very few sites.

KERNEL This is the name given to the lowest level of UNICOS. Whenever a program requires an operating system service, such as I/O, it makes a system call to the Kernel. This then handles the I/O control and drives the I/O devices.

LDCACHE This is a cache area in the SSD which permits I/O to those filesystems which are ldcached to perform at speeds from 8 to 500 MBytes/sec, depending upon whether or not the data is in an LDCACHE page. LDCACHE is allocated on a per filesystem basis. If a program does I/O to a filesystem,

such as /tmp, which is ldcached, then that data will automatically go via the LDCACHE area for that filesystem. To get the most from an LDCACHED file, I/O should be well-formed (see later) and the record size should be a multiple of the LDCACHE page size. Small record size I/O to fragmented LDCACHED files can be worse than to non-LDCACHED contiguous files.

SSD This an acronym for "Solid-state Storage Device". The ECMWF C90's SSD consists of 4 GBytes of memory. This memory is slower, and hence cheaper, than main memory, of which we have 1 GByte. SSD can be used in several ways, of which LDCACHE and SDS are relevant to this article. The SSD is not directly accessible to user programs. They use it by making system calls to the Kernel. These system calls are the result of doing I/O to files which either reside in SDS or which reside on filesystems which are ldcached.

SDS This is an acronym for "Secondary Data Segments", an area in the SSD that can be reserved by a user program, enabling I/O to perform at speeds from 500 to 3000 MBytes/sec. The main difference, from a user's point of view, between LDCACHE and SDS, is that LDCACHE is shared, potentially by all programs running on the C90, while SDS is reserved for a particular file used by a particular program.

3. File Types and Structures

There are 3 main UNICOS file types or structures. These are TEXT, COS BLOCKED and UNBLOCKED and are illustrated in figure 1.

1) TEXT file:

This consists of bytes of ASCII data. Records are terminated by an ASCII linefeed character (LF - Hex code 0A). Because there is a record structure it is possible to use the BACKSPACE statement to reposition the file at a previous record. Also it is possible to read partial records. The Fortran standard allows the reading of just 1 byte from a record, the next read will occur not at the next byte, but at the first byte of the next record. This is what is meant in this article by "partial reads".

This data structure is produced by default when using Fortran FORMATTED WRITE statements e.g. WRITE(10, 901) X.

It is also produced using C "printf" and "fprintf" routines. It is also the type produced by such Unix utilities as editors, compiler and other utility listings etc.

2) COS BLOCKED file:

This consists usually of words (8 bytes) of data. Records are terminated by a special Record Control Word (RCW). The data is blocked up into 512 word blocks, each of which has a Block Control Word (BCW) at the beginning. There are pointers within the BCWs and RCWs to the next control word and there are special RCWs denoting End-of-File and End-of-Data. This type of file also supports BACKSPACE and partial reads.

This data structure is produced by default using Fortran UNFORMATTED WRITE statements e.g. WRITE(10) X, or using the BUFFER OUT statement.

It can also be produced using the C "ffwrite" routine (FFIO) with an appropriate call to "asgcmd" beforehand. It is not produced by any of the normal Unix utilities and is not recognised by other Unix implementations, as it is Cray specific.

3) UNBLOCKED file:

This consists of a sequence of unstructured bytes. Because there is no structure to it, it has no record structure associated with it. Because there is no concept of a record with this file type, it is not possible to use BACKSPACE to get to the previous file position, nor is it possible to read partial records and expect to be positioned at the beginning of the next record.

It can be produced by Fortran UNFORMATTED WRITES or BUFFER OUT statements, provided that an appropriate "assign" control statement has been used beforehand.

It is also produced using the C "write" and "fwrite" statements.

Of course any file, even TEXT or COS BLOCKED files, may be treated as UNBLOCKED. If, by using an assign statement, the user treats a COS BLOCKED file as UNBLOCKED then the first read executed on that file will cause the BCW for the first block to appear as the first word of data read from the file. This may, but more probably may not, be what the user intended.

4. Data Flow Throughout the System

Figure 2 shows the paths over which data flows between the user's program and the physical disk. There is a "brick wall" shown which symbolises the interface at which the data transfers between the user memory and the UNICOS Kernel memory. Since user programs themselves cannot directly write/read the Kernel memory or physical disk, they have to issue system calls to get the Kernel to do this for them. The amount of time that this will take depends upon several factors, including how much data is to be transferred, and how busy the UNICOS Kernel already is processing other system requests. This time is known as "SYSTEM TIME" and is charged to the user and reported at the end of the job.

The current version of the operating system, UNICOS 7.c.2, is single-threaded. By this we mean that, in general, while the Kernel is processing one system call, it cannot simultaneously

process another. If for example the Kernel were processing a very long request for one process and 15 other processes then made a system request, all 16 CPUs of the C90 would be in use by UNICOS. However, of the 16 CPUs, only 1 would be doing useful work while the other 15 would be waiting for that one to finish. The next major release of the operating system UNICOS 8 is multi-threaded and should solve this problem.

The blocks in figure 2 represent points at which the data may "stopover" on its way between the user process and the disk.

The first block represents an ARRAY in the user's program.

The second block represents a LIBRARY BUFFER or a MEMORY RESIDENT LIBRARY BUFFER. These are blocks of memory within the user's field length, which has been reserved by the I/O library, and as such are "owned" by the user. Each library buffer is specific to a particular file, files do not share library buffers.

The third block represents the SYSTEM BUFFERS. These are blocks of memory within the Kernel's field length (outside the user's field length), which the Kernel uses to match the program's logical I/O requests with physical I/O to the disk. I/O to a physical disk is "quantized", the quantum being a disk sector. It is only possible to write integral multiples of sectors to a disk, whereas the user can, if he so wishes, write any size of data. The system buffers are used to "marry up" this obvious discrepancy in functionality. They are not "owned" by any particular user but are a shared resource, not only between user processes but also between different files in the same process.

The fourth block represents SDS. These are blocks of memory in the SSD. Each SDS buffer is "owned" by a particular user process and is specific to a particular file, but still only accessible via system calls. In this way it acts in a way similar to a MEMORY RESIDENT LIBRARY BUFFER.

The fifth block represents LDCACHE BUFFERS. These are blocks of memory in the SSD. Each buffer is "owned" by the Kernel and used in a way similar to that of the SYSTEM buffers. SDS and LDCACHE are not mutually exclusive, a file with an SDS buffer can, and probably will, still use LDCACHE buffers.

5. Techniques for Improving I/O Performance

When looking at improving I/O performance several factors need to be taken into consideration, but of all the factors which cause inefficiency, the one that causes the most is the inefficient use of the SYSTEM BUFFERS. By cutting out system buffer use it is usually possible to make your programs much more efficient. It was possible to cut down the elapsed time that it took a test program to do its I/O from 30 minutes to 20 seconds, just by bypassing the system buffers. A real production program now runs in 1/3rd of its original time just by doing the same.

The next best way of cutting out inefficiencies is to pre-allocate

your files contiguously on the disk.

The third way is to cut down on the number of system calls needed to transfer the data.

Cutting out the use of SYSTEM BUFFERS involves doing "well-formed" I/O. The term "well-formed" means that the I/O must be done from "User memory space" to "Kernel memory space" in integral multiples of the disk sector size, and the file must always be positioned on a sector boundary. From now on the term "block" will have a specific meaning:

1 Block = 512 words (4 KBytes)

There are 2 types of disk connected to the C90:

DD-62 (Sector size = 1 Block)
DA-62 (Sector size = 4 Blocks)

The C90 DA-62 disks hold only the /tmp filesystem, while all other filesystems are on DD-62 disks. From this we can see that all I/O requests which transfer integral multiples of 4 blocks (2 Kwords, 16 KBytes) are well-formed on both types of disk.

Pre-allocating your files contiguously on the disk has several benefits. Normally users do not pre-allocate their files, though some UNICOS commands, such as "cp" do this internally. Users normally let the Kernel allocate chunks of disk as their file grows. A typical example is this:

The user writes the first sector or 2 of data to disk. The Kernel allocates 1 or 2 sectors to the file. The user writes more data and again the Kernel allocates 1 or 2 sectors to the file. Eventually, when the file reaches a certain size, the Kernel allocates 28 sectors at a time (28 being the size of a track on the disk). When the file is complete it may consist of hundreds of small chunks which are spread over a wide area of the disk. This can have disastrous consequences if the file is LDCACHED. It can also take a long time to read this file because the disk heads will have to do a lot of positioning in order to read the data.

Pre-allocating the file contiguously will get round these problems. This can be done at the control statement level or at the C program level. In the following examples "numblks" is the required size of the file in BLOCKS (4 KBytes) and "filesize" is the required size in bytes.

Control statement:

```
setf -c -n numblks filename || setf -n numblks filename
```

C program example:

```
if (fstat(filename, &st) == 0) filesize -= st.st_size;
/*     Already exists ialloc allocates an additional extent */
if (ialloc(filename, filesize, IA_CONT, 0) < 0)
/*     Contiguous allocation failed */
```

```
if (ialloc(filename, filesize, 0, 0) < 0)
/*     There is insufficient disk space */
{
    (void) perror("No space left / quota exceeded");
    return (1);
}
}
```

Note that the "||" construct is used on the control statement in case there is insufficient contiguous space on the filesystem, at which point we still pre-allocate but without the "-c" (contiguous) parameter. Without the "|| setf .." construct the job would abort if the amount of space requested was not available as one contiguous piece.

6. Fortran I/O Library Processing

Now before everyone dashes off to change their Fortran code to do well-formed I/O, one needs to look at how the Fortran I/O library handles the COS BLOCKED and UNBLOCKED file types. TEXT files are not mentioned here because formatted Fortran output is usually only a small part of the I/O, being used for human-readable results. Having said that, I have in the past come across a Fortran program which used formatted I/O to pass data to another program, where there was no need for it be formatted. This is very inefficient and should be avoided.

COS BLOCKED FILES

As mentioned earlier COS BLOCKED datasets are the default for files read and written using unformatted READ/WRITE or BUFFER IN/OUT statements. They may also be explicitly requested using an "assign" control statement, the format for which is:

```
assign -F cos.synctype:bufsize filename/unit-no
```

Where:

synctype = "sync" for synchronous I/O; "async" for asynchronous I/O which does read-ahead and write-behind; "auto" (default) which is "sync" for bufsize < 64, otherwise "async".
bufsize = size of LIBRARY BUFFER in units of a block, the default is 48.

e.g. To assign a COS BLOCKED dataset, with a buffer size of 56 blocks, using synchronous I/O, accessed via Fortran unit number 10, the assign statement would be:

```
assign -F cos:56 u:10
```

Figure 3 shows what happens when the user program does 4 WRITE statements to create a file with 4 records, each containing the data belonging to a different array. In this example you can see that the 4 WRITE statements simply transfer data between the user's arrays and a library buffer. The data does not actually get transferred to disk via a system call until the buffer is flushed, when the file is closed, and then only 1 system call is

involved. If we change the example to write these arrays 56 times to a file with a buffer size of 56, then we can see that these 224 Fortran WRITE statements will generate 3 system requests each transferring 56 blocks to disk. All these are well-formed requests and so will bypass the system buffers. This is very good, but if the file was continually being created, rewound read, rewound read, rewound ...etc. it would be even better to keep the file totally within in the library buffers until it was closed and flushed to disk. This would require a library buffer of 168 blocks (84 Kwords) and the assign statement would be:

```
assign -F cos.sync:168 u:10
```

UNBLOCKED FILES

Having looked at COS BLOCKED datasets, let us now see how UNBLOCKED datasets are handled by the I/O library. To assign a file as UNBLOCKED the format of the assign control statement is:

```
assign -F syscall filename/unit-no
```

e.g. To assign as unblocked the file "fort.11", the assign statement would be:

```
assign -F syscall f:fort.11
```

Figure 4 shows the same Fortran program as before but this time writing to an UNBLOCKED dataset. One can see that this is very inefficient as every Fortran WRITE statement generates a system call with an I/O request which is not well-formed. Let us look at this in more detail in "pseudo-code" form.

```
User:  does 1st WRITE --> system call
Kernel: if no system buffer is free {
        flush the oldest buffer to disk }
        copy data from the user's array to the buffer
```

```
User:  does 2nd WRITE --> system call
Kernel: if the previous buffer is not still around {
        if no system buffer is free {
            flush the oldest buffer to disk }
        read the previous sector of data from disk into the
        buffer }
        copy the first part of the data from the user's array to the
        end of the buffer
        if no other system buffer is free {
            flush the oldest buffer to disk }
        copy the last part of the data from user's array into the
        buffer
```

```
User:  does 3rd WRITE ....etc.
```

As one can see, a large amount of Kernel and disk activity is going on. This is the main cause of "system time" on the C90. How can this be prevented without having to completely rewrite the I/O portion of the Fortran program? The simplest way is to introduce a "well-formed" LIBRARY BUFFER. The format of

the assign statement to do this is:

```
assign -F cache:bufsize:nbufs filename/unit-no
```

Where:

bufsize = size of LIBRARY BUFFER in units of a block, the default being 8.
nbufs = the number of cache "pages", the default is 4.
e.g. To assign 1 cache page of 56 blocks to an unblocked file "fort.11", the assign statement would be:

```
assign -F cache:56:1 f:fort.11
```

There is no need to specify "syscall" as that is implied, though one could specify it for clarity e.g.

```
assign -F cache:56:1,syscall f:fort.11
```

Now, as can be seen from figure 5, all Fortran I/O calls just operate on data in the LIBRARY BUFFER instead of making ill-formed I/O requests.

For UNBLOCKED files which are small enough to fit within the memory of the job it is better to use "mr" (memory resident), rather than "cache", especially if the files are "scratch" files which are created and used solely within this program. The format of the assign control statement to do this is:

```
assign -F mr.savscr.ovfopt:init:max:incr filename/unit-no
```

Where:

savscr = "save" (default) to pre-load and post-store the data in the file on disk; "scr" to indicate that the data is neither to be read from nor written to the actual disk file.
ovfopt = "ovfl" (default) to allow excess data to overflow onto disk; "novfl" to abort the program if the amount of data exceeds the memory resident buffer.
init = the number of blocks of memory initially allocated to the memory resident buffer.
max = the maximum number of blocks that may be allocated.
incr = the block allocation increment.

e.g. To assign the same file "fort.11" to be memory resident, one would need to reserve 168 blocks of memory, in order for it to fit completely within the buffer. The assign statement would be:

```
assign -F mr.save.novfl:168:168:0 f:fort.11
```

or

```
assign -F mr.scr.novfl:168:168:0,syscall f:fort.11
```

The latter statement explicitly specifies "syscall" for clarity and states that the file is a "scratch" file, that will only exist for the duration of the program.

Figure 6 shows this example. One should not overflow the

memory resident buffer, hence use the "novfl" parameter. If this is not coded and the data overflows the buffer then all the overflow portions of the data will (in this example) be ill-formed and system time will again accrue.

7. LDCACHE Data Flow

Since several of the filesystems on the C90, including /tmp, are LDCACHED it is instructive to see how this works. Figure 7 shows an example of LDCACHE use. In this example the amount of LDCACHE allocated to the filesystem is 4 pages each of size 2 sectors. In reality /tmp has 1280 pages each of 7 sectors. The example shows a well-formed I/O request for 4 sectors of data. It also shows a file which was not allocated contiguously on the disk. The flow is as follows:

```
User:  does the READ --> system call
Kernel: if no LDCACHE buffer is free {
        flush the oldest buffer to disk }
1)  read 2 sectors from disk into the buffer (only 1 of which
    is from our file)
    transfer our 1 sector to the user's field length
    if no LDCACHE buffer is free {
        flush the oldest buffer to disk }
2)  read 2 sectors from disk into the buffer (only 1 of which
    is from our file)
    transfer our 1 sector to the user's field length
    if no LDCACHE buffer is free {
        flush the oldest buffer to disk }
3)  read 2 sectors from disk into the buffer (only 1 of which
    is from our file)
    transfer our 1 sector to the user's field length
    if no LDCACHE buffer is free {
        flush the oldest buffer to disk }
4)  read 2 sectors from disk into the buffer (only 1 of which
    is from our file)
    transfer our 1 sector to the user's field length
```

As can be seen, in this example there could be up to 8 physical disk I/O transfers from wildly different areas on the disk, before the data is actually available in the user's program. If the file was contiguous on the disk this would be reduced to 4 physical I/O requests. In real life, with a buffer size of 7 sectors, there would probably be only 2 physical I/O requests (224 KBytes) for such a contiguous file, instead of 8 physical I/O requests (896 KBytes) and the disk positioning, which is the main cause of slow I/O, would be kept to a minimum.

One point that should be made concerning FFIO. When one codes the "-F" parameter on the assign statement, some of the other parameters lose their default values and the "-s" parameter cannot be specified. One important parameter that may need to be explicitly specified is the "-T off" parameter. This is needed if the program uses the SETPOS routine to position within a file and WRITE a record at that point. If a "-F" parameter is not used but a "-s u" used instead, then the default is "-T off", but when using "-F" the default becomes "-T on", the file will become

truncated immediately after the record just written, and any data which was behind this record in the file will no longer exist.

8. I/O Performance Tools

By far the most important performance tool for monitoring I/O is "procstat". Procstat produces a file which is analyzed by the "procview" (an X-windows application) and "procrpt" utilities. Data is put in the procstat "raw file" every time an I/O operation is performed from within the program being monitored. One of the nice features of procstat is its ability to monitor any program without having to recompile it, in fact procstat can also be used on Unix utilities such as "cp", "tar", "cpio" and the like.

Below is a simple procstat example in which the normal "a.out" command is replaced by:

```
procstat -R $TMPDIR/rawfile a.out
procrpt -s -F all $TMPDIR/rawfile
```

it is as simple as that. Now comes the "tricky" part - interpreting the procrpt output, procview is much easier to interpret.

First, skip to the end of the report to the title:

```
"PROCSTAT FORTRAN FILE REPORT"
```

Here you will find a set of reports, one for each of the Fortran files in your program. Such a report (taken from a test program run) looks like this:

```
Fortran Unit Number      11
File Name                 fort.11
Command Executed          a.out
Date/Time of Open        11/02/93 11:52:43

System File Descriptor    4
Type of I/O               sequent1 unformatted
File Structure            unknown
```

Fortran I/O	Count of Statements	Real Time
READ	128000	15:00.1751
WRITE	128000	6:42.9804
REWIND	200	1.0175
CLOSE	1	.0031

32506 Bytes xferrd per Fortran I/O statement
0.78% Of Fortran I/O statements did not initiate a system request

From the last value (0.78%) we can see that every READ or WRITE statement caused a system call to be made, which in general is a bad thing. This is because the file was assigned as UNBLOCKED (or UNKNOWN as procrpt reports). The higher this number the better, 100% being the target to aim for.

From the REWIND count (200) we can deduce that the file is being continually written, rewound, read and rewound again.

From the time taken to read and write the file, compared to other files in this program which only took 20 seconds in total, it is obvious that this file is a prime candidate for optimisation. One can find out more information about this file by looking earlier in the report. Search backwards for either string:

```
"File Name = fort.11" or "Fortran Unit Number = 11"
```

One will see the above again, but there is also other interesting information, such as that given at the foot of the page.

From this one can see that the file was 4193280 Bytes long (0.5 MWords) and that it was continually being written then read. It also confirms that each of the 12800 Fortran READs and WRITEs issued a system call. This file is a prime candidate for MEMORY RESIDENT or CACHE. Just putting a 56 block CACHE on this file with the assign statement:

```
assign -F cache:56:1 u:11
```

changes the total I/O wait time on it from 31 minutes 41 seconds to 19 seconds; while converting to MEMORY RESIDENT with the assign statement:

```
assign -F mr.scr.novfl:130:130:0 u:11
```

changes the total I/O wait time on it to 0.85 seconds.

9. C Program I/O Optimisation

This article has concentrated on Fortran programs, since these make up the majority of programs run on the C90. With C programs things are not quite so easy.

It is still possible to optimise the I/O of C programs. The use of "ialloc" has been shown in an earlier example. In order to use FFIO, the source code has to be modified to call the FFIO routines (ffopen, fread, fwrite, fclose etc.). Unless special constructs, such as #define macros, are used, this could cause portability problems. Well-formed I/O is an obvious way of

optimising. Library buffering can be used by coding "fopen" calls instead of "open" and "fread/fwrite" instead of "read/write". The size of library buffers (by default 8 blocks) can be specified using the "setvbuf" routine.

e.g.

```
if (setvbuf(file_out, (char *) NULL, _IOFBF, 56*4096) != 0)
{
    perror("Cannot set buffer");
    return (1);
}
```

10. Summary

In this article I have attempted to give some ideas of ways in which I/O can be optimised. In most cases this does not necessarily involve changing any code, merely using procstat and procvew or procrpt and then adding one or two assign control statements to the jobs. In the past few months we have tried to identify jobs on the C90 which are performing inefficiently, as regards the amount of system time they use, since these jobs affect all other jobs on the machine. The user is then approached and encouraged to use the techniques described in this article, with the end result that I/O wait time, and hence turnaround time, decreases dramatically (from say 1 hour to 10 minutes).

Everyone's I/O requirements are different and it would be impossible to design a default I/O system to satisfy everyone's needs. I believe that Cray have done an excellent job in designing procstat and FFIO, and users are more than willing to use these features when they see how easy they are to use and how much benefit they can gain from using them. Obviously it is better to design code from the very beginning with "well-formed" I/O, contiguous data and reasonable buffer sizes, rather than have to rely upon users coding assign statements at a later stage, but for "dusty deck" codes FFIO etc. is the next best thing.

Two Cray manuals describing I/O techniques in detail are:

(SG-3075 7.0): I/O User's Guide
(SG-3076 7.0): Advanced I/O User's Guide

```
Minimum File Size = 4193280 (bytes)
Maximum File Size = 4206592 (bytes)
Final File Size = 4193280 (Bytes)
```

Sys I/O Function	# of Calls	# Bytes Processed	# Bytes Requested	Wait Time (Clock Periods)		
				Max	Min	Total
Read	12800	419328000	419328000	2523192870	19650	215991603956
Write	12800	419328000	419328000	1999536112	20170	240666850980
Seek	200	n/a	n/a	31263627	4294	70589203
Truncate	100	n/a	n/a	39963846	6552	26900319

System I/O Function	Avg Bytes Per Call	Percent of File Moved	Average I/O Rate (MegaBytes/Second)
Read	32760.0	9968.4	0.466
Write	32760.0	9968.4	0.418

Fig 1: UNICOS File Types

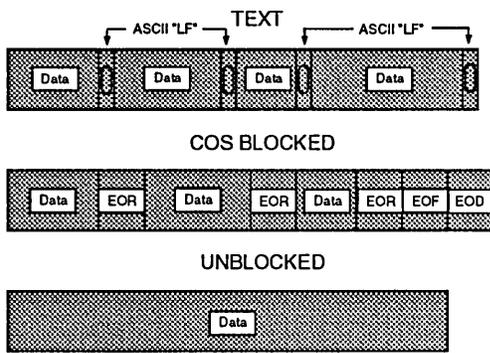
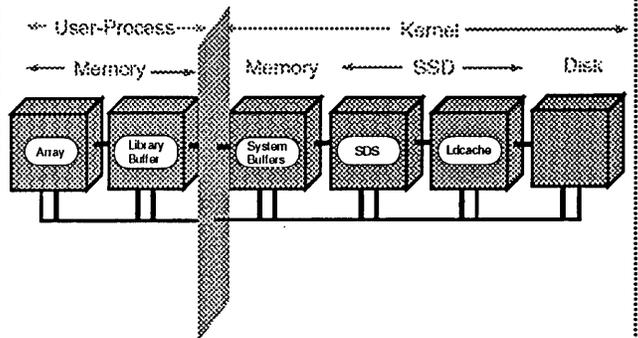


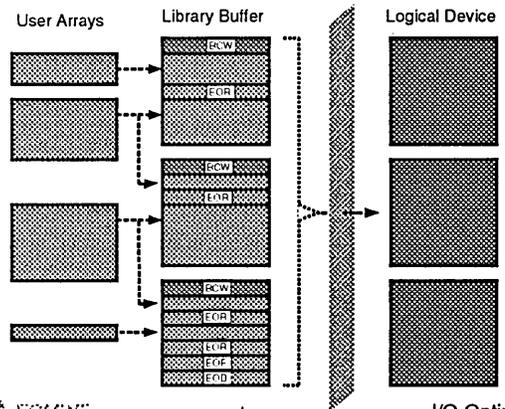
Fig 2: Data Flow Diagram



The COS blocking Layer

- assign -F cos.synctype:bufsize
- synctype = "sync" for synchronous I/O
- synctype = "async" for read-ahead, write-behind
- synctype = "auto" (default). This is "sync" for buffer size < 64, otherwise it is async.
- bufsize = size of working buffer in units of 512-word blocks (default is 48)
- BACKSPACE & partial record READs supported
- Fewer system calls than FORTRAN I/O calls
- the buffer gets flushed when it is full

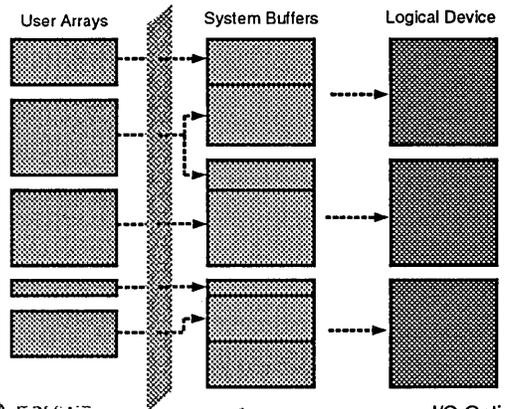
Fig 3: assign -F cos Data Flow



The SYSCALL Layer

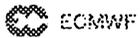
- assign -F syscall
- BACKSPACE and partial record reads are NOT supported
- Each FORTRAN I/O call causes a system call
- Can cause a large system overhead, especially if the I/O requests not "well-formed"
- "well-formed" means "the record size is a multiple of the device sector size and the file is positioned on a sector boundary"

Fig 4: assign -F syscall Data Flow



The CACHE Layer

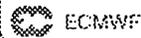
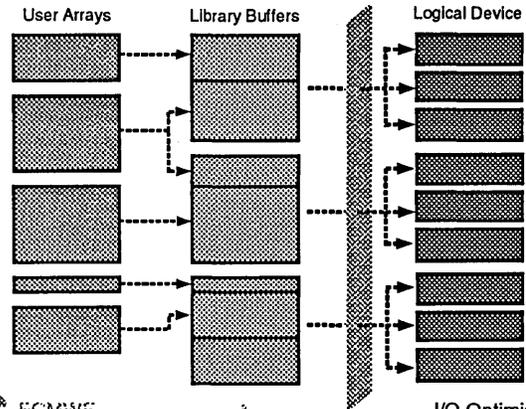
- `assign -F cache:bufsize:nbufs`
- `bufsize` = size of each cache page buffer in units of 512-word blocks (default is 8)
- `nbufs` = number of cache pages (default is 4)
- Fewer system calls than FORTRAN I/O calls
- cache preemption is done on the basis of "least recently used"



7

I/O Optimisation

Fig 5: assign -F cache:syscall Data Flow



8

I/O Optimisation

The MR Layer

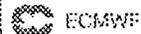
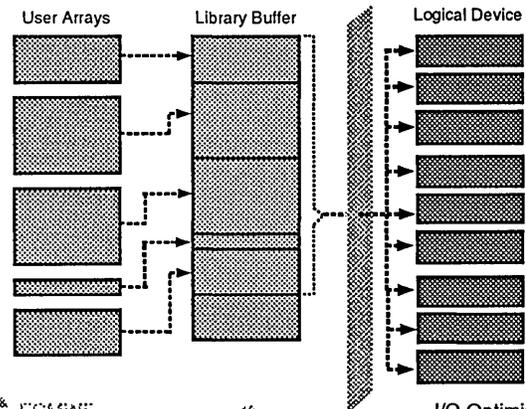
- `assign -F mr.savscr.ovfopt:init:max:incr`
- `savscr` = "save" (pre-load post-store the data in the file)
- `savscr` = "scr" (no pre-load or post-store)
- `ovfopt` = "ovfl" (Excess data overflows to the next layer)
- `ovfopt` = "novfl" (Abort on overflow)
- `init` = number of 512-word blocks initially allocated
- `max` = maximum number of blocks to allocate
- `incr` = block allocation increment



9

I/O Optimisation

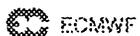
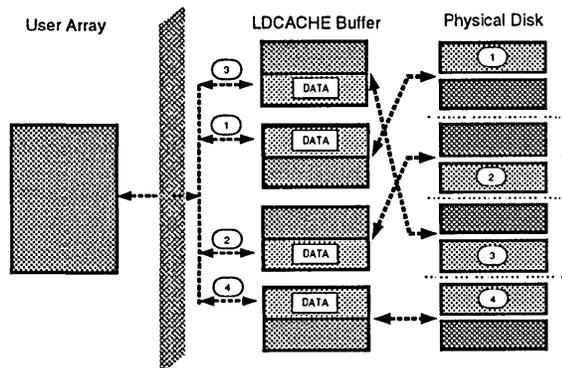
Fig 6: assign -F mr Data Flow



10

I/O Optimisation

Fig 7: Example LDCACHE Data Flow

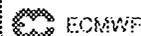


11

I/O Optimisation

SSD Partitions at ECMWF

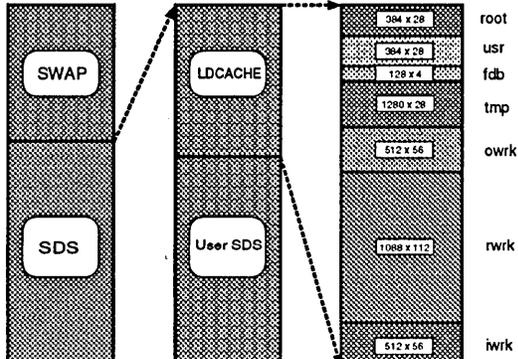
- **Swap Space [190 MW]**
Used since swap to disk is very slow
- **SDS (Secondary Data Segments) [322 MW]**
LDCACHE for various filesystems
User files which are assigned to SDS
Auxiliary arrays (new CFT77 feature)
- **SSD-resident filesystems [0 MW]**
Not used at ECMWF



12

I/O Optimisation

Fig 9: SSD Partitions Diagram



ECMWF

13

I/O Optimisation

I/O Performance tools

- M='ja -m' a.out [parameters]
ja -ol -p \${M}:
- procstat -R rawfilename a.out [parameters]
- procview rawfilename
- procrpt -s -F all rawfilename



ECMWF

14

I/O Optimisation

Procstat output - part 1

```

Fortran Unit Number      11
File Name                 fort.11

Type of I/O              sequential unformatted
File structure            unknown

Fortran I/O      Count of      Real
Statements       Statements    Time

READ              128000    15:00.1751
WRITE             128000     6:42.9804
REWIND              200      1.0175
CLOSE                1       .0031

32506 Bytes transferred per Fortran I/O statement
0.78% Of Fortran I/O Statements did not initiate
a system request
    
```



ECMWF

15

I/O Optimisation

Procstat output - part 2

```

Minimum File Size = 4193280 (bytes)
Maximum File Size = 4206592 (bytes)
Final File Size = 4193280 (bytes)

Sys I/O # of # Bytes Wait Time (Clock Periods)
Function Calls Processed Max Min Total

Read      12800 419328000 2523192870 19650 215991603956
Write     12800 419328000 1999536112 20170 240666850980
Seek        200      n/a 31263627 4294 70589203
Truncate   100      n/a 39963846 6552 26900319

System I/O Avg Bytes Percent of Average I/O Rate
Function    Per Call   File Moved (MegaBytes/second)

Read          32760.0      9968.4      0.466
Write         32760.0      9968.4      0.418
    
```



ECMWF

16

I/O Optimisation

RECOMMENDATIONS

- **PRE-ALLOCATE** your files using:
self -c -n numblks filename || self -n numblks filename
- **Cut down on system calls & buffer use by:**
 - Using the MR layer
 - Using the CACHE layer
 - Using the SDS layer
 - Tailoring your library buffer sizes
- **Cut down on CPU time by:**
 - NOT using FORMATTED I/O
 - NOT using COS Blocked files with well-formed I/O



ECMWF

17

I/O Optimisation

I/O IMPROVEMENTS IN A PRODUCTION ENVIRONMENT

Jeff Zais and John Bauer

Engineering Applications Group
Cray Research, Inc.
Eagan, MN

ABSTRACT

Recent I/O enhancements have dramatically improved I/O throughput for many important engineering applications. Several individual examples (such as NASTRAN and ANSYS jobs) have been discussed at recent CUG meetings. This paper documents improved overall system throughput, not just gains from isolated jobs. A suite of numerous engineering applications were submitted to a dedicated C90 system using commonplace I/O enhancements such as Idcache. The same benchmark suite was then submitted to the same machine, this time using the EAG FFIO routines to reduce the I/O wait time. Job accounting comparisons quantify the improvement in system throughput.

EAG FFIO Background

Work has been in progress on the EAG FFIO layers since 1992. The main goals of the project are to allow individual applications to:

- 1) Provide their own caching of I/O data
- 2) Utilize user striping of files
- 3) Measure I/O performance on a per file basis
- 4) Track I/O activity on an event basis

A key feature of the layers is that no source code changes are required. They can be used from both FORTRAN and C codes.

Details on implementation of the EAG FFIO layers have been presented in previous CUG discussions. In the Spring 1993 meeting, Doug Petesch, John Bauer presented "An Application Independent Intelligent I/O Layer" which gave an overview and several MSC/NASTRAN specific examples. For the Summer 1993 CUG Applications Symposium, John Bauer presented an "I/O Optimization" talk which detailed the I/O process and how I/O could be improved with EAG FFIO.

Purpose of This Study

The purpose of this study is to demonstrate how EAG FFIO can help improve I/O performance. As

most CRAY sites, Idcache is employed for this purpose. This study will compare I/O performance for a C90 system using Idcache and EAG FFIO caching.

In addition, previous presentations of EAG FFIO have concentrated on single jobs. This study will present how EAG FFIO can improve I/O performance in a simulated production environment with many concurrent jobs running in NQS batch queues.

System Configuration

The system used for this study was configured in the following manner:

C90 (serial number 4001)
8 cpus
256 Mw Memory
512 Mw SSD
8 I/O Clusters

The file system used for the I/O consisted of 32 DD60 drives, on 32 I/O channels.

Job Mix Description

The job mix for this study consisted of I/O intensive third-party applications typical of those run at an engineering site:

Structural Analysis:	MSC/NASTRAN ABAQUS ANSYS
CFD:	FIDAP VSAERO
Chemistry:	GAMESS

Even within these codes, the I/O intensity is problem dependent. Therefore, in order to prove the efficiency of EAG FFIO, examples were deliberately chosen so that all jobs would be I/O intensive.

However, the most I/O intensive examples were deliberately not used in the job mix. This is because those examples could not practically run without EAG FFIO (using just ldcache). Therefore, the results were not overtly exaggerated. The detailed descriptions of the jobs are:

MSC/NASTRAN

- N1 = 120,000 DOF Normal Modes
- N2 = 150,000 DOF Normal Modes
- N3 = 580,000 DOF Normal Modes

ABAQUS

- A1 = Linear Statics; 3131 RMS Wavefront
- A2 = Linear Statics; 3131 RMS Wavefront
- A3 = Linear Statics; 3131 RMS Wavefront

ANSYS

- J1 = Normal Modes; 2843 Wavefront
- J2 = Normal Modes; 2418 Wavefront
- J3 = Normal Mopes; 2418 Wavefront

VSAERO

- V1 = Customer CFD Benchmark

FIDAP

- F1 = 10,200 Elements; 388,000 DOF;
3D Compressible Laminar
Temperature Dependent Calculation

GAMESS

- G1 = ab initio single point energy calculation

A schematic of the job mix is shown in Figure 1. All jobs were submitted to NQS at one time. The queues were set so that the longest running jobs ran first. Then, as those jobs completed shorter running jobs filled in the remaining time. In this way, the cpus had very little idle time while waiting for the longest running job to finish. The job mix was tuned so that for the case where EAG FFIO was used, all 8 cpus were busy until 5 minutes before the finish time (out of 55 minutes total).

The job mix results were tabulated by adding up the user cpu, system cpu, and I/O wait times as reported by the "ja" command. The total of user cpu, system cpu, and I/O wait time was used as a measure of time required to complete the individual jobs.

For the case where the job mix was run with ldcache, the first cpus became idle 41 minutes before the longest running job finished. This caused less contention between the jobs for system resources, which should skew the final results in favor of the ldcache case.

The NQS queues were set so that the cpus were

oversubscribed. Initially, 9 jobs were active. One of the jobs (ANSYS job J1) ran with 2 cpus, so there was a demand for 10 cpus, with only 8 available. One of the ABAQUS jobs requested 95 Mw of memory, so that the total memory required for the jobs was 244 Mw. This was less than the available memory, so there was no swapping involved.

The results are summarized in the following tables:

LDCACHE Results

Code	cpu time (sec)	I/O wait (sec)	Data Transfer (Mw)
NASTRAN (3 jobs)	6512	7960	21611
ANSYS (8 jobs)	6700	622	1795
VSAERO (5 jobs)	586	1240	2950
GAMESS (1 job)	1133	717	2191
<u>ABAQUS (7 jobs)</u>	<u>6363</u>	<u>1996</u>	<u>5148</u>
Job Mix	23724	12591	34201

EAG FFIO Results

Code	cpu time (sec)	I/O wait (sec)	Data Transfer (Mw)
NASTRAN (3 jobs)	6159	1359	12585
ANSYS (8 jobs)	6685	118	1648
VSAERO (5 jobs)	602	311	2950
GAMESS (1 job)	1128	110	2185
<u>ABAQUS (7 jobs)</u>	<u>6188</u>	<u>535</u>	<u>5005</u>
Job Mix	23135	2552	24655

The total of C90 "busy time" (cpu time plus I/O wait time) is 10.09 hours for the ldcache system. Using EAG FFIO, this time is reduced to 7.14 hours, a reduction of 30%.

The EAG FFIO EIE cache also has the effect of reducing the total data transferred. This is most dramatic in the NASTRAN results, while the other codes undergo only a small decrease. The data transfer reduction is due to requests for data that happen to reside in the cache, avoiding a transfer from disk.

Even though the reduction in data transferred is greatest in NASTRAN, the I/O wait time is reduced in all codes. Additionally, the cpu time does not increase. In fact, the total cpu time decreases, since less system cpu time is spent managing I/O tasks.

Memory vs. SDS Resident Cache

The first experiment established that EAG FFIO was effective in reducing I/O wait time. A second experiment was run in order to determine if the cache is more efficient as memory resident or SDS resident.

In this experiment, a subset of the jobs in the first mix was used. This was done so that the jobs plus their EAG FFIO cache could both remain in memory.

<u>Code</u>	<u>Job</u>	<u>Mem (Mw)</u>	<u>Cache (Mw)</u>	
ANSYS	J1	24	8	
NASTRAN	N1	13	36	
NASTRAN	N2	15	18	(2 copies)
NASTRAN	N3	37	18	
GAMESS	G1	2	2	(2 copies)
ABAQUS	A1	15	6	
<u>ABAQUS</u>	<u>A2</u>	<u>15</u>	<u>13</u>	
Totals		121	101	

Therefore, a total of 222 Mw (Memory plus EIE cache) are required to run this job mix entirely in memory.

The results of the study were as follows:

<u>Item</u>	<u>SDS Cache</u>	<u>Memory Cache</u>
user cpu (sec)	18790	18830
system cpu (sec)	383	305
total cpu (sec)	19173	19135
I/O wait (sec)	2559	2643
Data Transfer (Mw)	30438	29898

As the table shows, there are no significant differences in performance between the SDS cache and the Memory cache. Slight differences exist in the distribution of cpu time. For the SDS cache, the system time increases from 305 to 383 seconds. This is because system calls are required to manage the SDS resident EIE cache, while a memory resident cache is managed under the user cpu time. While the SDS cache system cpu increases, the user cpu time decreases. The net effect is a slight increase in cpu time for the SDS cache, but the increase is very slight compared to the total cpu time.

System Configuration Suggestions

The job mixes have demonstrated that a system using EAG FFIO can offer better performance than a system using ldcache. Setting up a system with ldcache is easier. This section presents some suggestions on configuring systems for EAG FFIO.

One issue is whether the cache should be SDS or memory resident. The example of the previous sections suggests that there is no significant

difference in performance, so this can be determined strictly on local resources.

It is important that EAG FFIO and ldcache work together. For the codes which are not I/O intensive or are not running EAG FFIO, users will still want the benefit of ldcache. When SDS space is used for ldcache, an SDS resident EIE cache presents no problems. This is because the EIE cache will bypass the ldcache area and communicate directly with disk.

However, with an SDS ldcache and a memory resident EIE cache, the default path is for the EIE cache to use the ldcache to and from disk. This problem can be avoided by using the "set.ldraw" option from EAG FFIO. Then the memory cache will bypass the ldcache and use raw I/O directly to and from disk.

Another issue is the size of SDS space reserved for EAG FFIO usage, compared with SDS space available for swapping and ldcache. At one site, nearly all the cpu time is consumed by ABAQUS jobs. This machine is a Y-MP 2E, with 32 Mw of memory and 32 Mw of SSD. Separate tests showed that at most 5 ABAQUS jobs should run concurrently. Each requires a 3 Mw EIE cache. Therefore, 16 Mw of SDS space is reserved for EAG FFIO, while the rest is used for swapping and ldcache.

Another site runs multiple applications (50% PAM-CRASH, 30% MSC/NASTRAN, 10% ABAQUS). Here the SDS division is more complicated, with some SDS space reserved for I/O intensive codes (NASTRAN, ABAQUS), and the other SDS space used as ldcache for PAM-CRASH.

Summary

This paper has demonstrated the following items:

- For I/O intensive applications, EIE cache can provide superior performance than ldcache.
- For a C90 system simulating an I/O intensive production environment, EIE cache increased system throughput by 40%.
- Production environment C90 demonstrated no significant difference between performance of SDS and Memory based EIE cache.
- Some sites beginning to allocate SSD in order to make SDS space regularly available to EAG FFIO users.

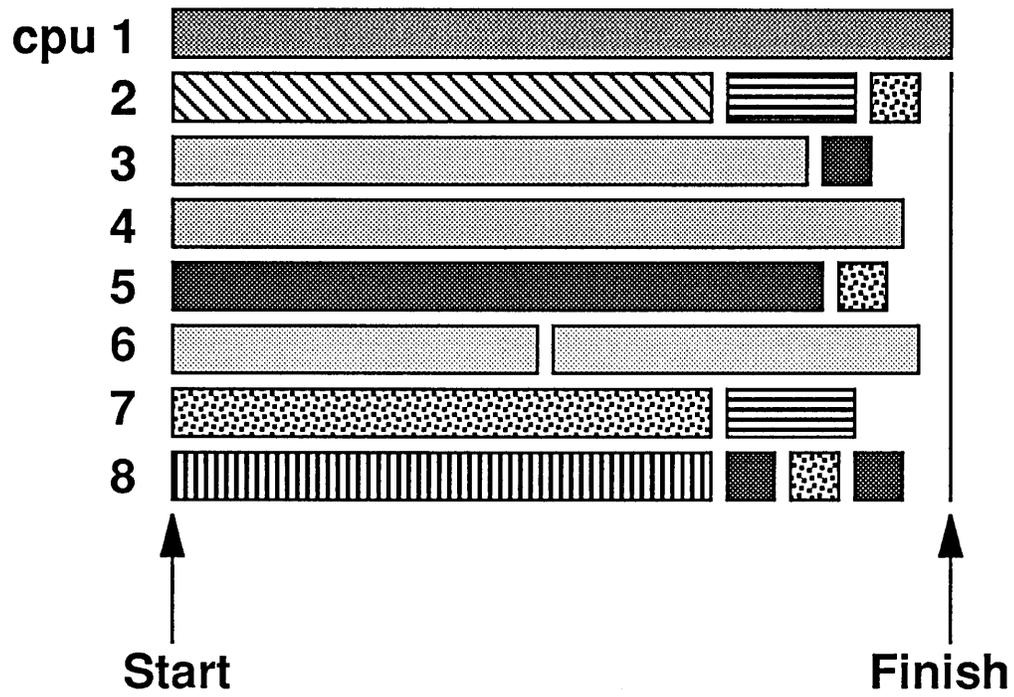


Figure 1.

Schematic of job mix activity. Longest-running jobs were submitted to queues first. When they completed, shorter jobs filled in remaining time so that idle time near the finish of the job mix was minimized.

NEW STRATEGIES FOR FILE ALLOCATION ON MULTI-DEVICE FILE SYSTEMS

Chris Brady
Cray Research, Inc.
Boulder, Colorado

Dennis Colarelli
NCAR
Boulder, Colorado

Henry Newman
Instrumental
Minneapolis, Minnesota

Gene Schumacher
NCAR
Boulder, Colorado

Abstract

As UNIX-based operating systems are required to support large datasets, multi-device file systems will become more standardized. Cray Research supercomputers have made extensive use of multi-device file systems for many years and have customers using file systems with over 100 devices. Performance problems due to file fragmentation and system allocation overhead occur as file sizes and the number of files grow, and as a significant percentage of the file system space is used.

In this paper we look at how different file allocation strategies find contiguous disk blocks in multi-device file systems, and how different bitmap scan techniques enable efficient searches for free space. Performance factors include: contiguous free space allocation, system response time, file system fragmentation and free space distribution.

Our results are based on measurement and simulation. The measurement data and the simulation parameters were taken from a Cray Research Y-MP8/864 located at the National Center for Atmospheric Research (NCAR). The results show the dramatic effect that the choice of file allocation strategies can have on large, multi-device file systems.

1. Introduction

This project began as a result of observed performance degradation of climate models running on the Cray Y-MP8/864 at the National Center for Atmospheric Research (NCAR). The cause was traced to fragmentation of the large files used by these models. We set out to explore alternate methods of allocating space for files which would reduce fragmentation and allocation time.

When allocating space, the file system would attempt to find contiguous regions within the partition where the file was created. Selection of a partition for new files was on a round-robin basis, where each new file was placed on succeeding partitions. We observed that it was often the case that when the selected partition was unable to allocate a contiguous region for a file, other partitions had sufficient contiguous space. Even when no partition had a sufficiently large contiguous free region, judicious scanning of the partitions could result in a file with only a few fragments.

In addition, we found that file systems which were nearly full caused on average excessively high allocation times. The allocation cpu time charged to the user would on

occasion cause jobs to fail by exceeding the cpu time limits estimated by the user.

To improve this situation, we investigated alternate methods of finding contiguous free regions in multi-partition file systems. We knew if we were to search more of the allocation bitmap to find free contiguous regions, we would also need more efficient search methods than currently being used.

In Section 2 we look at the effects file fragmentation has on I/O performance. Section 3 describes different strategies for allocating contiguous free regions. Section 4 describes bitmap scan methods used to locate free blocks in the file system's allocation bitmap. In Section 5 we discuss the simulation models and performance analysis of the different allocation and bitmap scan methods. Section 7 presents a summary of the performance analysis and conclusions which guided our implementation decisions.

2. Fragmentation and I/O Performance

In an attempt to quantify the adverse effects of file fragmentation, a synthetic test case was created. In this test we measured I/O wait time, allocation time, and write system time with varying amounts of fragmentation.

I/O wait time is the period of time spent waiting for an I/O request to complete. In general it can be thought of as the sum of seek and rotational latency and data transfer time, subject to IOS write behind buffering. Before each I/O can begin, the disk heads must be positioned to the correct track (seek latency) and the position of the disk sector with respect to the disk heads must be correct (rotational latency). Since the disk driver will write each fragment of a file as a separate operation, file fragmentation greatly increases the probability of incurring additional latency.

Allocation time is the CPU time used by the kernel to identify and allocate disk space. When the available space on a disk is fragmented, the file system bitmap must be searched repeatedly until enough fragments have been found to satisfy the request. Fragmentation of the file system free space results in increased file allocation overhead.

The write system time is the CPU time used by the kernel to process the write request. For each fragment in a file the kernel must identify the starting location and length and make a separate I/O request to the disk driver. Consequently, file fragmentation increases the amount of work that the kernel must do.

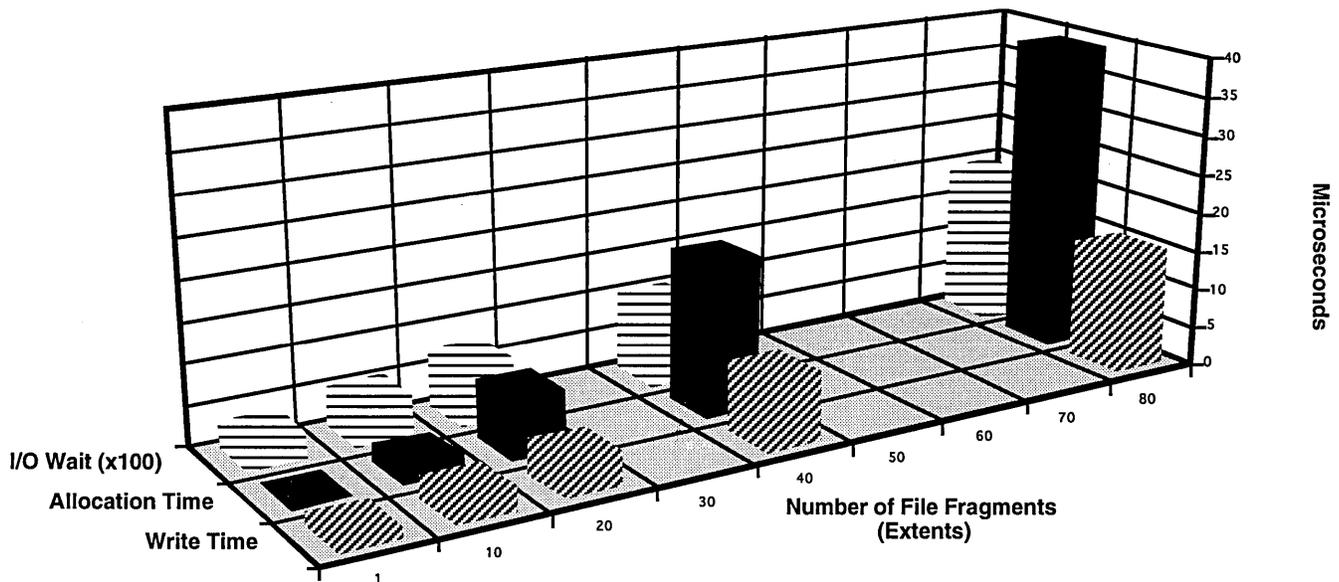


Figure 1. Effect of Number of File Fragmentation on Write, Allocation and I/O Wait Times

The test case consisted of allocating and writing 800 block (3 Mbyte) files which had varying amounts of uniform fragmentation. The file system where the files were written was pre-fragmented with uniformly spaced allocation of 20 blocks each. The space between allocation was adjusted so that the 800 block allocation would result in the desired amount of fragmentation.

Figure 1 shows the results of the tests. Note that for all of the factors measured there are significant performance penalties associated with fragmented files.

3. Allocation Strategies

We looked at three different strategies for identifying partitions for file allocation: round-robin, first-fit and best-fit. Each of these methods have characteristics which are advantageous for different distributions of file sizes and unique performance costs which are functions of size distribution.

There are three round-robin allocation strategies currently available for Cray Research systems [1]. *Round-robin all files* (rrf) places each new file in succeeding partitions. Directories and i-nodes are placed in the first partition whenever possible. *Round-robin first level directories* (rrd1) places those directories directly under the root directory in succeeding partitions. Subsequent level directories and normal files are placed in the partition associated with the directory at the beginning of their path. *Round-robin all directories* (rrda) places each newly created directory in succeeding partitions. Normal files are placed on partitions associated with their parent directory. Round-robin all files is the default and the strategy selected for our study. We refer to it simply as round-robin in the remainder of the paper.

Within a partition, file space for round-robin is allocated on a first-fit basis. The partition is scanned from the beginning for the first available contiguous region which will satisfy the allocation request. If there are no contiguous regions which can completely satisfy the request, the largest region is allocated, and smaller regions are selected from within the partition to complete the allocation. If the partition becomes full before the request can be satisfied, allocation spills over to the succeeding partition.

First-fit is a technique we borrowed from memory allocation systems [2]. With first-fit, each partition is scanned until one is found which contains a contiguous region of free space sufficiently large enough to satisfy the allocation request. If no region in the file system is able to provide a contiguous region large enough to satisfy the request, the largest region is allocated and another first-fit search begins to satisfy the remaining allocation. This procedure is continued until the request is completely satisfied or the file system becomes completely full.

If the first-fit search began at the first partition for each allocation request, files would tend to become unevenly distributed across partitions, with most of the files residing in the lower partitions of the file system. This could possibly result in performance penalties manifested primarily in I/O wait time. Partitions with a disproportionate number of files will most likely receive more I/O requests, resulting in channel contention and increased seek and rotational latency. To avoid this problem, we begin the first-fit search on the partition following the last allocation. This seems to give a reasonable distribution of files across the file system's partitions.

Like first-fit, we borrowed best-fit from memory allocation techniques [2]. Unlike first-fit, best-fit searches for contiguous regions of free space that either exactly satisfy the allocation request or selects the smallest region whose size is greater than the allocation request. Note that this frequently requires searching the entire file system before an allocation can be made. We deal with unsatisfiable requests and file distribution in the same manner as first fit. If a contiguous region cannot be found which is greater than the request size, the largest region is allocated and another best-fit search attempts to satisfy the remaining request. Likewise, subsequent searches begin following the partition of the last allocation to avoid free space clustering.

4. Bitmap Scan Methods

The current UNICOS bitmap search function offers fairly good performance, 14.0 milliseconds/megabyte. However the increased demands of alternative allocation strategies and the significant growth of file system sizes require much higher performance bitmap functions.

The good performance of current bitmap search function in UNICOS is primarily due to the use of the Cray *leading zero* hardware instruction. This instruction allows contiguous bit regions within a word to be processed in one operation, offering significantly better performance than bit at a time processing. However the current implementation's potential performance is limited by instruction scheduling. In this study we used a significantly faster scalar bitmap search function with performance of 5.7 ms/megabyte, a speedup of approximately 2.5, which was created by Dean Nelson of CRI software development.

In addition, a technique was developed that allows vector processing to be used for bitmap searches. To use the vectorized bitmap search the minimum number of zero words that must be located in the bitmap to satisfy the request is determined. This is done by subtracting 63 from the request size and then dividing by the word size. For example, an allocation of 191 bits would require at least $(191-63)/64$ or 2 words of zeros in the bitmap to satisfy the request. A bitmap region with two zero words would have 128 to 254 free bits depending on the state of the preceding and trailing words. To locate space for a 191 bit allocation "candidate" regions of 2 or more zero words are located in the bitmap using a vectorized word at a time search. When a "candidate" region is found the preceding and trailing words are examined to determine the exact size and starting location of the region. This technique can be used for any allocation size but suitable free space may not be identified for sizes less than 126 bits, depending on alignment. In this study the vector bitmap search was only used for allocations of at least 126 bits.

The performance of the vector bitmap search is .31 ms/megabit, 45 times the speed of the current scalar search. Since the startup costs for the vector search are low the vector search performance exceeds the improved scalar search for all allocation sizes unless the total bitmap size is very small (300-500 bits).

5. Simulation Models

In order to quantify the differences in these methods we built two models of file system allocation and ran a number of simulations. The factors we were interested in were file fragmentation, free space fragmentation and allocation time. We did not simulate I/O wait time or write system time as it was clear from our measurement study that these factors were a function of file fragmentation and not directly a function of allocation methods.

The first model was used to measure the effect of allocation strategy on file fragmentation and allocation strategy/bitmap scan methods on allocation time. This model consisted of taking a snapshot of the temporary file system on NCAR's Cray Y-MP8/864. This file system is comprised of 16 partitions with three different sizes and 56 gigabytes total space. The file system had 42% free space when the snapshot was taken. Allocation requests ranging from 1 megabyte to 300 megabytes were submitted to the simulation.

Figure 2 shows the simulation results for the effect of allocation strategy on file fragmentation. First-fit and best-fit produced considerably less file robin allocation constrains the search to the currently selected device. Round-robin allocation is unable to utilize more optimal free regions that may be available on other devices.

Round-robin performance may be particularly poor with file systems comprised of different size devices. Since the round-robin allocation method does not take into account different disk sizes, the smaller disks tend to fill up first. As the free space on a disk nears zero, there are frequently a large number of small remaining fragments. Since allocation is restricted to the selected disk so long as there is available space, a large allocation will be satisfied with many small allocations, resulting in severely fragmented files. As both first-fit and best-fit are able to examine the entire file system, dissimilar device sizes will not adversely affect file system performance.

Best-fit produced slightly less average file fragmentation than first fit. We attribute this to the fact that best-fit saves larger regions for later allocations. The small remainder regions produced by best-fit were productively used by small allocation requests.

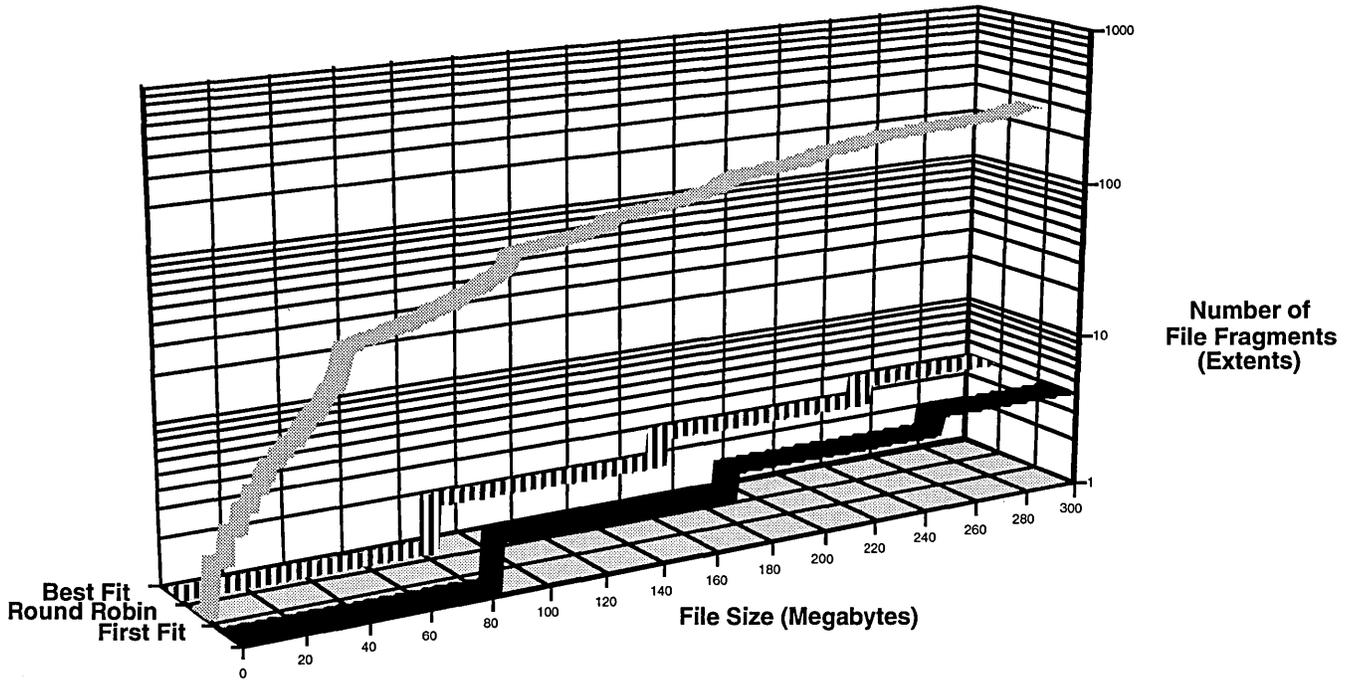


Figure 2. Effect of Allocation Strategy on File Fragmentation

Figures 3 and 4 show the effects of allocation strategy and bitmap scan methods on allocation time for files ranging in size from 1 megabyte to 300 megabytes. To get accurate timing data, actual kernel bitmap search functions were used in the simulation.

First-fit vector resulted in the least amount of CPU time for file allocation. The vector bitmap scan is substantially faster than the scalar for all allocation methods, but the greatest gains were for first-fit and best-fit.

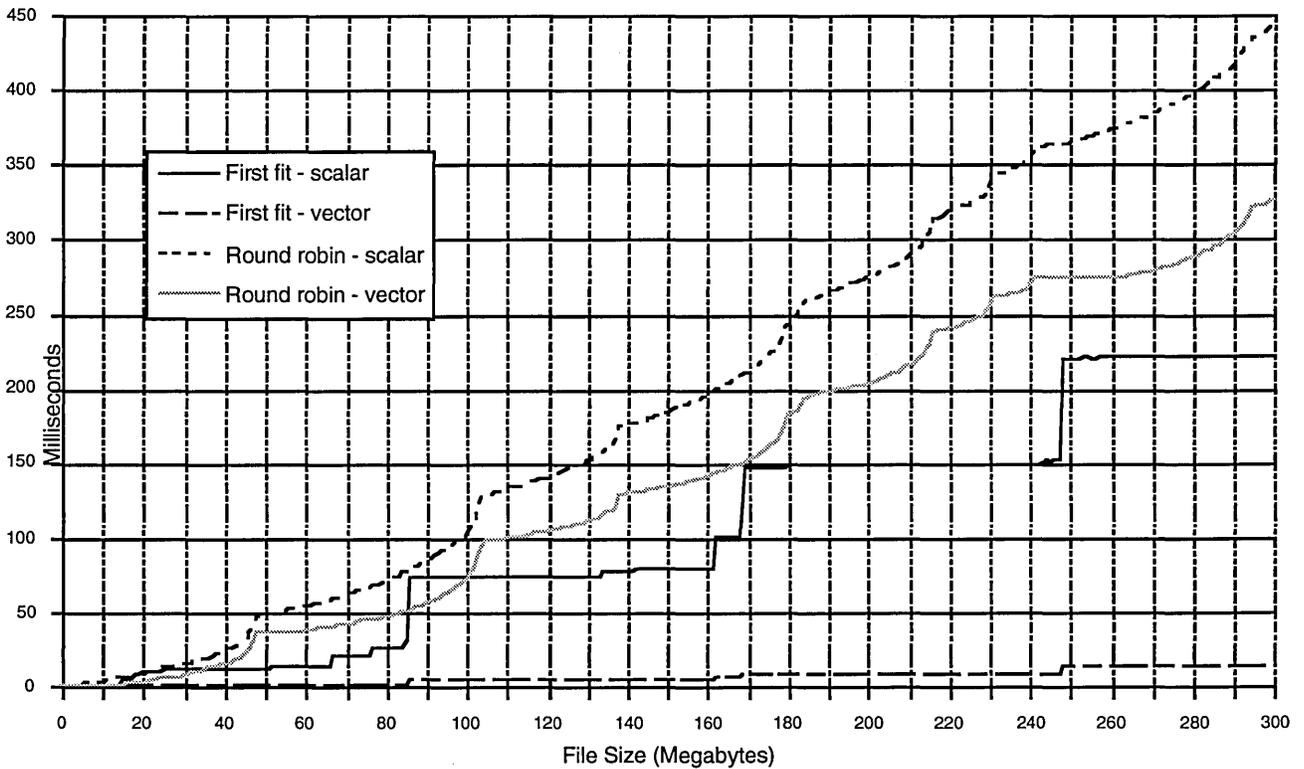


Figure 3. Allocation Time

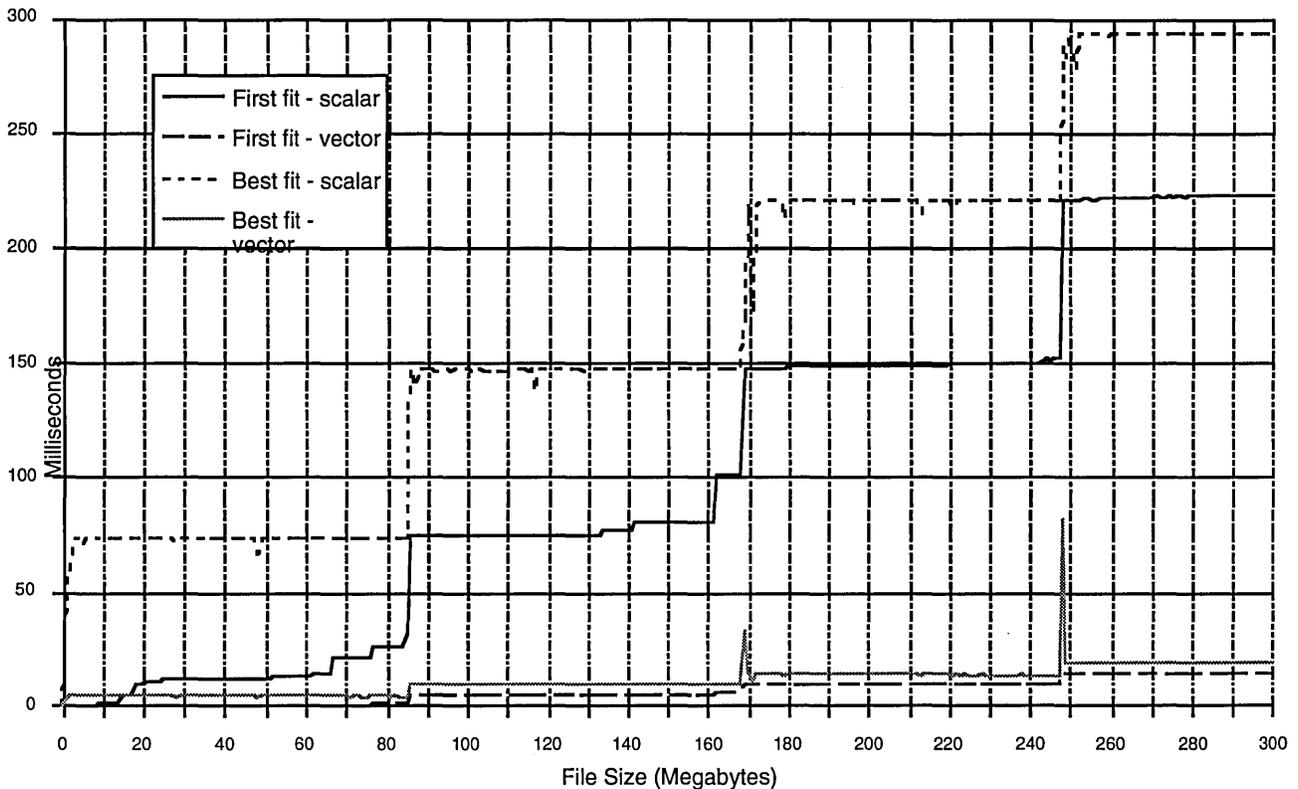


Figure 4. Allocation Time

The amount of CPU time required to do an allocation is closely correlated to the resulting number of file fragments. This is because the bitmap within a disk boundary must be re-scanned for each file fragment. Note the clear steps for first-fit and best-fit. These steps at 84, 168, and 248 megabytes correspond to 1, 2, 3 and 4 file fragments.

An important finding that is difficult to see in the chart is the performance of best-fit with smaller allocation sizes. Best-fit performance is the worst for file sizes up to 11 megabytes for vector and up to 125 megabytes for scalar. This is important when a large proportion of files are small. The poor performance of best-fit, relative to first-fit and round-robin, with small files is due to the requirement of finding an exact fit before stopping the search. With first-fit and round-robin an available region of sufficient size can often be found for small allocations without going very far into the bitmap. As the file size increases, first-fit and round-robin must go deeper into the map to find a suitable allocation, narrowing the difference between them and best-fit.

The second model was used to measure the effect of allocation strategy on file system fragmentation and provide a different view of allocation strategy/bitmap scan methods on allocation time. In this model the file system consisted of four partitions of varying sizes (1084368, 1084368, 813504, and 524640 blocks) with a total size of 3,524,880 blocks. File sizes were geometrically distributed with a mean size of 2 megabytes. The

simulation allocated files until the desired percentage of free space was reached and then random deletions were done to maintain the targeted free space. Allocations continued until a steady state was reached, after which the file system bitmap was analyzed for fragmentation.

Assuming files are uniformly distributed in a file system and file sizes are geometrically distributed, file system fragmentation will increase as free space decreases. This is because there are a larger number of small files deleted than large files deleted, and small files are primarily responsible for file system fragmentation.

Figure 5 shows the effect of allocation strategy on file system free space fragmentation. One of the unexpected findings was that first-fit resulted in the greatest amount of file system fragmentation. Since first-fit always allocates from regions large enough to satisfy requests (or the largest available), highly fragmented areas are often left untouched. Round-robin, on the other hand, allocates only from the selected partition, sometimes using up numerous small regions to satisfy a large allocation. While this results in file fragmentation, file system fragmentation is reduced.

It is interesting to note that file system fragmentation for round-robin starts to drop off as the file system reaches 90% full.

One of the arguments against the use of best-fit is that it leaves behind a large number of small unusable fragments.

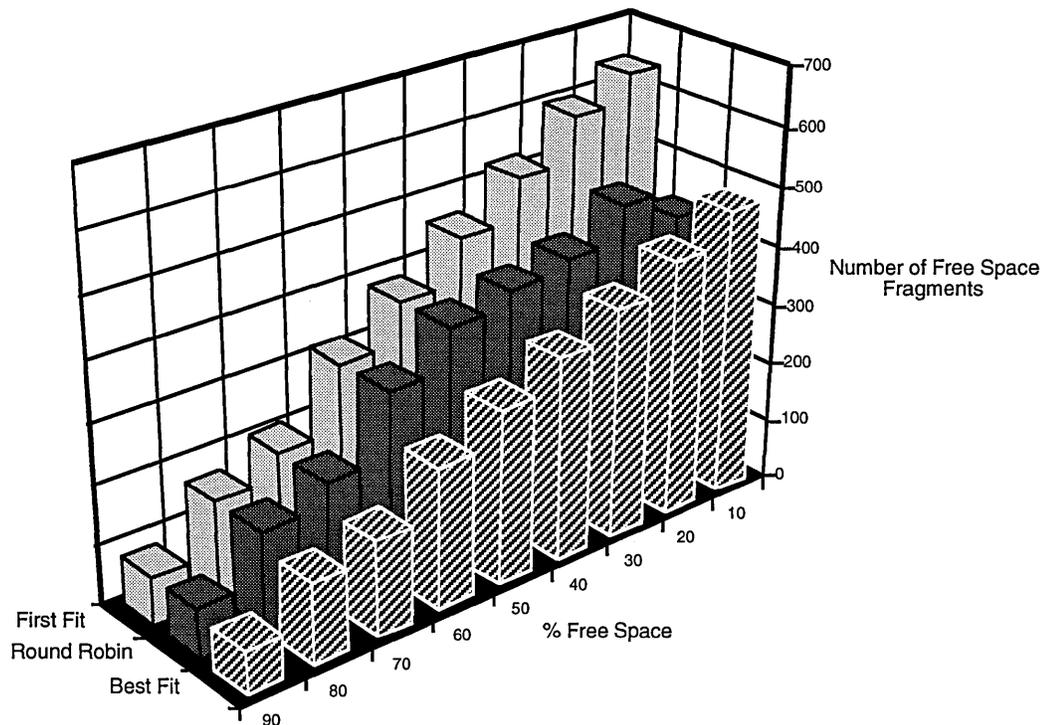


Figure 5. Effect of Allocation Strategy on File System Fragmentation

Yet in our study, best-fit produced the smallest amount of file system fragmentation. We attribute this to the distribution of file sizes was such that there were enough small files to effectively use the fragmented regions.

Figure 6 shows the effect of allocation strategy and bitmap scan method on allocation time. The results for first-fit and round-robin are fairly intuitive. File system fragmentation increases as the file system fills, as does the average search time. Significant gains can be seen at all points when the vector scan is used. Note that first-fit and best-fit benefit more from the vector search than does round-robin. As expected, the cost of best-fit allocation is significantly greater than the other methods. Note that there is a slight decrease in the allocation time for best-fit as the file system fills up, just the opposite of first-fit and round-robin. This can be attributed to the fact that as the file system fills there are a greater number of fragments, increasing the probability of finding an exact fit. Since best-fit frequently must search the entire bitmap, the average search distance does not increase as the file system fills.

Another result is that first-fit does a much better job of balancing the disk usage. With best-fit, the simulations show a large increase in allocation time between 50% and 40% file system free space. This increase is caused by a

partition that has become nearly full, whereas in the first-fit case, none of the disks fills until between 20% and 10% free space has been reached.

6. Summary and Conclusions

Of the three allocation methods studied, first-fit and best-fit resulted in less file fragmentation, since both can scan other disks when large blocks of free space cannot be found in the currently selected disk. Round-robin allocation must continue to allocate from the same disk until there is no remaining space on that disk.

First-fit allocation is faster than round-robin because it results in fewer file fragments. The close correlation in Figure 1 of CPU time and number of file fragments clearly illustrates the cost of file fragments. As expected, best-fit required the most CPU time for allocation due to the increased search distances to find an exact fit.

An important detail not illustrated in the charts is that the vector bitmap scan is significantly faster for all allocation sizes that are large enough to take advantage of vector scan. In this study the size threshold for using vector scan was 126 blocks. At the vector threshold the average allocation time for the scalar scan was 471 microseconds, compared to 29 microseconds for the vector scan.

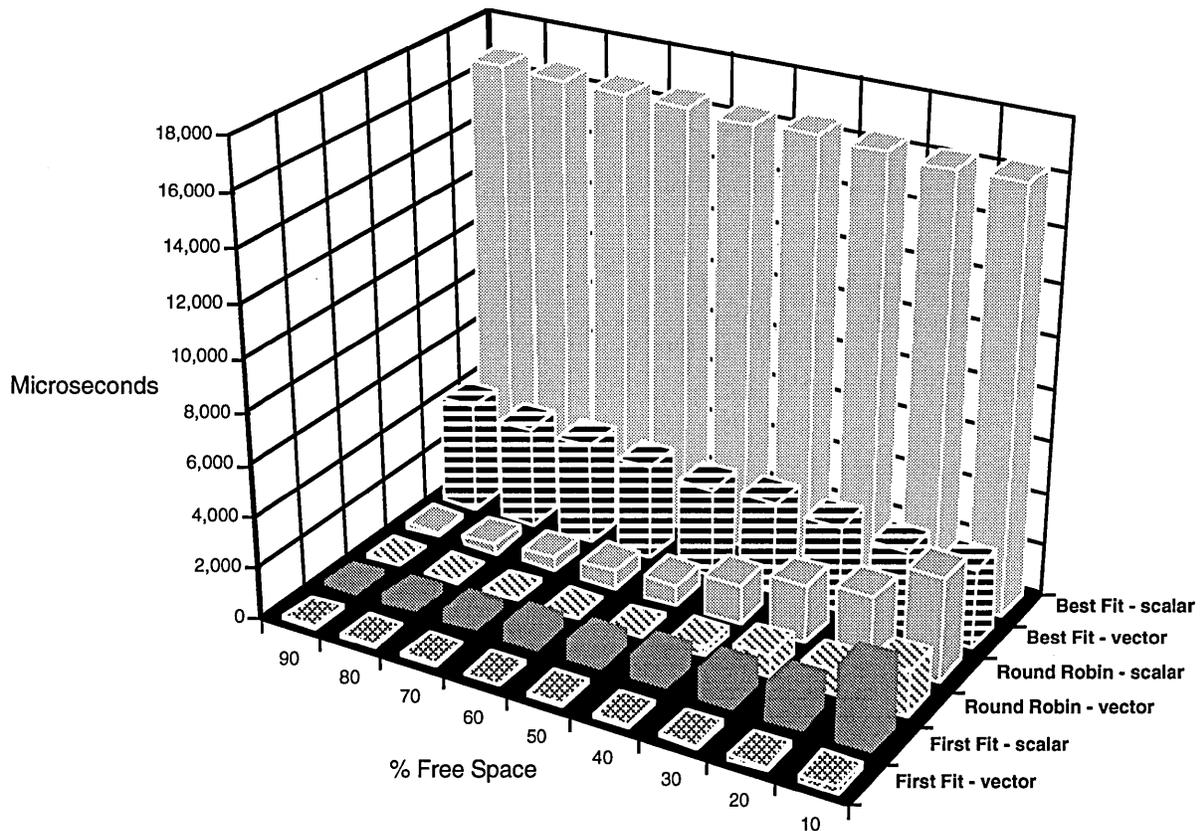


Figure 6. Effect of Allocation Strategy on Allocation Time.

Figure 6 shows that for file allocation best-fit required the greatest amount of CPU time. However Figures 3 and 4 show that the allocation time for large files using best-fit is often less than that for round-robin. This is because the performance of best-fit is far worse than the other methods for small files. Since the file sizes used in the tests illustrated in Figure 6 are geometrically distributed, there are many more small files in the simulation.

Since first-fit resulted in the most free space fragmentation, one would also expect greater file fragmentation, however, this was not the case. Since first-fit is able to use the most contiguous regions, it is less affected by free space fragmentation.

With these findings we conclude that first-fit allocation with vector bitmap scan is the best choice. This combination results in less allocation time than any other combination. File fragmentation is slightly greater than best-fit, but the difference is so small that it is an acceptable tradeoff given the significantly greater allocation time for best-fit.

We have implemented first-fit allocation/vector scan on NCAR's Cray Y-MP8/864 and Cray Y-MP/2D under UNICOS 6.1.6 and UNICOS 7.0.3. First-fit allocation/vector scan will be available on all NC1FS file systems in UNICOS 8.1.

7. Acknowledgements

We would like to thank Greg McArthur and John Sloan for their helpful comments and Belinda Housewright for her help in preparing this paper.

8. References

- [1] UNICOS System Administration, Volume 1, SG-2113 7.0.
- [2] D. E. Knuth, "The Art Of Computer Programming, Volume 1, Fundamental Algorithms, Addison-Wesley (1975).

Performance Evaluation/MPP MIG

The Performance of Synchronization and Broadcast Communication on the Cray T3D Computer System

F. Ray Barriuso

Cray Research, Inc.
Eagan, Minnesota

ABSTRACT

The Cray T3D Computer System provides a set of hardware features that allow for the efficient implementation of synchronization primitives. This paper will briefly describe the hardware support for fast synchronization and discuss the design and performance of the synchronization routines that implement barriers, locks, critical regions, and events as well as the broadcast routine used by the Cray T3D Fortran Compiler System (CRAFT).

1.0 Introduction

The CRAFT Fortran Programming Model [Pa93] describes a set of shared memory synchronization primitives. These primitives include barriers, locks, critical sections, and events. The Cray T3D Computer System contains some efficient mechanisms for implementing these primitives. These mechanisms include the hardware barrier/eureka network, the atomic swap mechanism, and automatic cache invalidation. The barrier/eureka network is used to implement barrier synchronization which notifies all processors when all processors have reached a particular point in a program. This network is also used to implement a specific type of event communication called eureka synchronization which can be used to alert all processors when one processor has reached a particular point in the program. The atomic swap mechanism is used for the implementation of locks, critical regions, and memory based event synchronization. Automatic cache invalidation is used to insure that blocked processors do not consume memory bandwidth while waiting to pass through a synchronization point. This paper will discuss in some detail the hardware support for synchronization and the design and performance of the different synchronization routines.

Broadcast is a form of communication where one processor sends some amount of data to many other processors. This paper will discuss the algorithms used to implement broadcast communication on the Cray T3D and then present the performance of broadcast for different numbers of processors and different size word transfers

2.0 Hardware Support for Synchronization

A brief description of the T3D hardware features used to support the synchronization primitives follows.

2.1 Barrier Registers

The hardware supports a pair of 8-bit memory-mapped barrier synchronization registers, BSR0 and BSR1. BSR0 has valid bits in positions 0 - 7 and BSR1 is valid in bit positions 8 - 15. Each processor possesses its own private pair of barrier registers. The T3D UNICOS-MAX operating system has reserved BSR1 for its own use, so only BSR0 is available for user synchronization. Although there are eight barrier bits in BSR0 that are available to the user, not all eight bits are necessarily available to any one application program. In fact, although this may change in the future, currently the operating system allocates exactly one barrier bit for each user application.

2.2 Eureka Barriers

The T3D hardware supports a mechanism in which the bits in each barrier register can be programmed to function in the conventional barrier mode or in *eureka* mode. In eureka mode, the conventional barrier AND tree is used as an OR tree using negative logic. So, instead of each processor waiting for all other processors to satisfy the barrier, all processors wait for one processor to satisfy the eureka. Logically this is all processors waiting for an event that one of the processors will eventually satisfy. As with conventional barrier bits, each user application is allocated one eureka bit by the operating system.

2.3 Atomic Swap

The T3D hardware supports an atomic swap capability that is ideally suited for shared memory locks. This capability allows a processor to perform an indivisible read-write sequence on any memory location, local or remote.

2.4 Automatic Cache Invalidation

The DEC Alpha EV4 RISC processor chip contains an 8K byte data cache [DEC91b]. It is a write-through, direct-mapped, read-allocate, physical offset-tagged cache that is organized as 256 lines of 4 64-bit words each.

A processor can set itself up such that any remote write into its local memory can invalidate the data cache line associated with that local address. A blocked processor can then spin on a local memory lock by doing a cached read once and then continue testing the lock in cache without consuming memory bandwidth. When a remote processor writes to the lock, the cache line on which the processor was spinning will be automatically invalidated causing the local processor to miss the cache and fetch the updated memory location containing the lock.

2.5 Write Barriers

The hardware supports a memory barrier instruction that can be used to insure that all subsequent loads or stores to local memory will not access memory until after all previous loads and stores to local memory have completed.

The hardware also implements a memory-mapped register, the User Status Register (USR), that indicates if there are any outstanding remote write request. This can be used to verify that all remote writes have reached their destinations.

Together, the memory barrier instruction and the USR can be used to insure that all writes, whether local or remote, have been written to memory before continuing with execution. This is similar to executing a *cmx* instruction on a Cray Y-MP computer system.

3.0 Synchronizations Primitives

3.1 Barriers

Barriers are a fast mechanism for synchronizing *all* tasks at once. They are expected to be extremely fast. Barriers are implemented with the following CRAFT compiler directive:

```
CDIR$ BARRIER
```

Implicit barriers occur in CRAFT programs at the bottom of distributed loops, when shared data is redistributed, and when shared arrays are allocated. Within serial regions of code, the barrier directive is a no-op.

Fortran programs not using the CRAFT Fortran Programming Model can call a barrier library routine.

```
CALL BARRIER ()
```

The barrier routine provides direct access to the hardware barrier mechanism and as such, is somewhat faster than the barrier directive. Unlike the barrier directive, if the barrier routine is called from outside a parallel region, the program will deadlock. The barrier routine must be called from within a parallel region and all processors must participate. The user can guard against calling the barrier routine from outside of a parallel region with the use of the `IN_PARALLEL` compiler intrinsic [Pa93].

The barrier mechanism actually consists of two parts, setting the barrier and waiting for the barrier to clear. As discussed in [Fr92], the point at which a processor sets the barrier and the point at which the processor waits for the barrier need not coincide. They do coincide with the `BARRIER` directive and the `BARRIER` routine. However, there are some user applications that lose a large amount of time waiting at barriers when the computation preceding the barrier is not homogeneous. The following three routines allow early arriving processors to move forward into an independent phase of the computation while the slower processors catch up:

```
CALL SET_BARRIER ()
CALL WAIT_BARRIER ()
L = TEST_BARRIER ()
```

The C version of these library routines are defined as:

```

void barrier(void);
void set_barrier(void);
void wait_barrier(void);
int test_barrier(void);

```

SET_BARRIER sets the barrier. It indicates that the calling task has arrived at a barrier synchronization point. WAIT_BARRIER suspends task execution until all tasks arrive at the barrier. TEST_BARRIER returns the state of the barrier: zero (C) or .FALSE. (Fortran) if barrier is not satisfied, nonzero (C) or .TRUE. (Fortran) otherwise.

Example:

```

      (block 1: must be completed
before block 2 is started)
      CALL SET_BARRIER();
      (unconstrained calculations)
      CALL WAIT_BARRIER();
      (block 2: cannot be started
until block 1 is completed)

```

3.1.1 Barrier Algorithm

The basic barrier algorithm is not at all complex. Ignoring deadlock detection and debugging support, the barrier routine looks like:

```

flush the write buffers;
wait for remote writes to complete;
set the barrier bit;
while (barrier bit is set)
  continue;

```

3.1.2 Barrier performance

The barrier routine was timed in the following fashion:

```

CALL BARRIER()
T1 = IRTC()
CALL BARRIER()
TIME = IRTC() - T1

```

The mean time to execute the barrier routine is approximately 1.5 microseconds (μ secs) and this is constant with respect to the number of processors as the following data indicates (times are in clock periods):

<u>Number of PEs</u>	<u>Mean Time</u>	<u>Std. Deviation</u>
32	242	19
64	255	21
128	257	24
256	258	26

The small, relatively invariant mean time indicates a fast, efficient barrier network that scales well with the number of processors.

3.2 Locks

Locks are used to serialize access to shared data. The lock operations are supported by three subroutines:

```

CALL SET_LOCK(lockword)
CALL CLEAR_LOCK(lockword)
L = TEST_LOCK(lockword)

```

The argument *lockword* must be a shared integer variable or array element. The lock routines issue an error message and abort if *lockword* is not declared to be shared. The subroutine SET_LOCK sets the lock. If the lock is set, the processor calling SET_LOCK spin-waits until the lock is cleared, otherwise the lock is set immediately. CLEAR_LOCK clears a lock whether it is set or not. TEST_LOCK atomically sets a lock and returns the state that the lock had (whether set or cleared) prior to the test.

3.2.1 Design Considerations

The lock routines were designed with the following assumptions and constraints:

1. Locks are used to serialize access to data as opposed to code. Therefore, they do not generally expect heavy contention for the shared lock word.
2. Blocked processors should not consume any memory bandwidth.
3. The lock routines should be able to efficiently support a large number of concurrently active locks.

The hardware atomic swap and automatic cache line invalidation mechanisms are ideally suited for implementing a non-polling software lock. The atomic swap mechanism is used to insure sequential access to the lock word. Automatic cache invalidation is used to allow a processor that is blocked on a lock to spin-wait without consuming any network or memory bandwidth. Spin-waiting in cache versus memory is important if other processors are reading or writing large amounts of data into the blocked processor's memory. However, the cost to set up and clear automatic cache invalidation is noticeable; It cost about 120 clock periods (cps) to set up automatic cache invalidation and another 120 cps (roughly 0.8 μ secs) to clear it again. The cost to set up automatic cache invalidation can generally be hidden since the processor is blocked from entering the critical region. The overhead to clear this mechanism, however, is totally exposed.

The lock algorithm implements a FIFO wait queue in which the user defined lock word is used to hold the

virtual PE numbers of the processors that are at the head and tail of the wait queue. Blocked processors are added to the tail of the queue and unlocked processors (processors that were blocked but are now being allowed into the protected region) are removed from the head of the queue. The wait queue itself is a :block(1) distributed array that is $N\$PES$ long [Pa93].

3.2.2 Lock Algorithm

The basic algorithms for SET_LOCK and CLEAR_LOCK follow:

SET_LOCK:

```
while ((lock = atomic_swap(lockword,BUSY)) == BUSY)
    delay(); /* wait a bit before trying again */
if (lock == LOCKED) { /* then add to tail of wait queue */
    if (lock.tail != EMPTY) {
        wait_queue[mype] = lock.tail;
        lock.tail = mype;
    } else {
        lock.head = lock.tail = mype;
    }
    local_lock[mype] = BLOCKED;
    lockword = lock;
    while (local_lock[mype]== BLOCKED)
        continue; /* spin-wait in cache */
} else {
    lockword = LOCKED;
}
```

CLEAR_LOCK:

```
while ((lock = atomic_swap(lockword,BUSY)) == BUSY)
    /* no delay gives priority to CLEAR_LOCK */
    continue;
flush write buffer;
wait for all remote writes to complete;
if (lock == LOCKED) {
    if (lock.head != EMPTY) { /* wakeup next PE */
        if (lk.head != lk.tail) {
            next_node = atomic_swap
                (wait_queue[lk.head],ZERO);
            local_lock[lk.head] = UNLOCKED;
            /* move next PE to head of queue */
            lk.head = next_node;
        } else { /* only one task is blocked */
            local_lock[lk.head] = UNLOCKED;
            lock.head = lock.tail = EMPTY;
        }
        lockword = lock;
    } else { /* No tasks waiting */
        lockword = UNLOCKED;
    }
}
```

3.2.3 Lock performance

The time to set an unlocked lock without competition and the time to clear a lock that does not have any waiting processors is approximately 2.8 μ secs each. The time to unlock a lock with blocked processors depends on the number of processors blocked as the following table shows:

Number of blocked PEs	Mean Time (μ secs)
1	7.5
3	9.0
7	10.1
15	11.6
31	12.6

When only one processor is blocked on the lock, the processor calling CLEAR_LOCK performs one atomic-swap on the lock word and then one remote write into the blocked processors memory in order to wake that processor up. When there are more than one processors blocked, the processor calling CLEAR_LOCK must also move the next processor in the wait queue to the head of the queue. This takes an additional atomic-swap. However, this alone does not account for the increase in latencies as more processors are blocked on the lock. For this experiment, when only one processor was blocked on the lock, it was always the inter-node processor neighbor. (i.e. When processor 0 had the lock, it was always processor 1 who was blocked and vice versa.) When there are more than one blocked processors, then the majority, if not all, of the atomic-swaps and remote writes traverse the network. As the number of processors increase, the average distance in the torus between the processor who has the lock and the next processor in the wait queue also increases. Thus, the mean time to acquire the lock, wake the processor at the head of the queue, and move the next processor to the head of the queue also increases.

Another interesting measurement is to determine the time it takes to get the last processor into a highly contended locked region. The following code segment attempts to measure this by computing the mean time for each processor to get into the critical region over a large number of runs and then finding the maximum mean time for any processor:

```

do i = 1, ntests
  call barrier()
  t1 = irtc()
  call set_lock(lock)
  times(i) = irtc() - t1
  call clear_lock(lock)
enddo
mean = compute_mean(times, ntests)
max_mean = max(mean)

```

Notice that this doesn't strictly measure the time for the last processor for each loop iteration since one can never know which processor will be the last to arrive. However, by taking the maximum of the averages across all processors, we are determining the mean time into the locked region for the processor who was most often at the tail of the queue.

Number of Processors	Max. Mean Time (μ secs)
2	9.0
4	35.5
8	75.7
16	198.4

3.3 Critical Sections

A critical section is a specialized form of lock that protects a region of code from being executed concurrently. Critical sections are implemented with the following CRAFT compiler directives:

```

CDIR$ CRITICAL
CDIR$ END CRITICAL

```

Every CRITICAL directive must have a matching END CRITICAL directive within the same program unit and the directives must be perfectly nested. Branching into and out of critical sections is not allowed.

3.3.1 Design Considerations

Critical regions were designed with the following assumptions:

1. Critical regions are used to serialize access to code as opposed to data. Therefore, heavy contention at the CRITICAL directive is expected.
2. Critical regions will be used to protect relatively small sequences of instructions, such as incrementing a shared counter. The important performance consideration is the time it takes to get all processors through the protected region. Therefore, these algorithms do not take the time to enable and disable automatic cache invalidation; Blocked processors spin-wait in memory.

The code for CRITICAL and END CRITICAL use a mutual exclusion algorithm proposed by [Sc90]. This algorithm is more efficient than that used by SET_LOCK and CLEAR_LOCK in the following ways:

1. In the algorithm for CRITICAL, the processor entering an uncontested critical region simply performs one remote atomic swap operation to acquire the critical region lock. With SET_LOCK, the processor performs one remote atomic swap and one remote write in order to acquire the lock.
2. In the algorithm for CRITICAL, a processor that is blocking on the critical region lock performs one remote atomic swap and one remote write. With SET_LOCK, the processor performs *at least* one atomic swap and two remote writes. If the lock word is in the BUSY state, the processor may issue several remote atomic swaps.
3. In the algorithm for END CRITICAL, a processor leaving a highly contended critical region performs one remote write in order to wake up the processor at the head of the wait queue. With CLEAR_LOCK, the processor performs one remote atomic swap and two remote writes.
4. In the algorithm for END CRITICAL, a processor leaving a critical region in which no other processors are waiting performs one remote atomic swap. With CLEAR_LOCK, the processor performs one remote atomic swap and one remote write.

Although more efficient, the critical region mutual exclusion algorithm is unsuitable for use in the general lock routines. The largest drawback is that every user lock must have its own wait queue. This means that in the general case, if there are 64 locks in a user application, there is a possibility that the application will also need 64 wait queues. For a given application, there is no way for the library to know how many wait queues to allocate statically. The wait queues could be allocated dynamically, but the cost of this dynamic allocation plus the cost of managing the different queues within SET_LOCK/CLEAR_LOCK seems at least as expensive as the current lock routines.

A smaller drawback of the critical region mutual exclusion algorithm is that it does not guarantee a FIFO wait queue. There does exist a small timing window where an early arriving processor to the critical region could be shuffled to the end of the wait queue. Although this window is fairly small, it does exist and early arriving processors do sometimes get shuffled to the end of the wait queue. Having a strict

FIFO wait queue has been an important issue for the LOCKON/LOCKOFF routines on the Cray Y-MP/C90 computer systems with some of Cray's customers but it is not clear if this is an issue for MPP systems.

Although these drawbacks limit the algorithm's usefulness for the general lock routines, it is a useful algorithm for critical regions. In the absence of task teams, critical regions are implemented with one lock. This implies that only one wait queue is needed and it can be statically allocated by the start-up routine. Also, it is believed that a strict FIFO wait queue for critical regions is not necessary.

3.3.2 CRITICAL/END CRITICAL Algorithm

The general algorithms for CRITICAL and END CRITICAL follow.

CRITICAL:

```
lock = _atomic_swap(critical_lock,mype);
if (lock != UNLOCKED) {
    local_lock[mype] = BLOCKED;
    crit_wait_queue[_MY_PE()] = ZERO;
    /* add PE to the wait queue */
    crit_wait_queue[lock] = mype;
    while (local_lock[mype]== BLOCKED)
        continue;
}
```

END CRITICAL:

```
flush write buffer;
wait for all remote writes to complete;
if (crit_wait_queue[mype] == ZERO) {
    /*
     * No other PEs appear to be waiting. We
     * need to make sure.
     */
    lock = _atomic_swap(critical_lock,ZERO);
    if (lock == mype)
        return;
    /*
     * We have accidentally removed a processor
     * from the wait queue. We have to put it back.
     */
    usurper = atomic_swap(lockword,lock);
    while (crit_wait_queue[mype] == ZERO)
        continue; /* wait for PE to finish enqueueing */
    if (usurper != ZERO) {
        /*
         * Some PE got into the queue ahead of our
         * victim. Place the victim at the tail of
         * of the wait queue. For this unfortunate
         * PE, the wait queue is no longer FIFO.
         */
        crit_wait_queue[usurper] = crit_wait_queue[mype];
    } else {
        local_lock[crit_wait_queue[mype]] = UNLOCKED;
    }
} else {
    local_lock[crit_wait_queue[mype]] = UNLOCKED;
}
```

3.3.3 Critical Region Performance

Probably the most interesting statistic for critical regions is the time it takes to get a specified number of processors through an empty critical section. This time was measured using the following code segment:

```
do i = 1, ntests
    call barrier()
    t1 = irtc()
cdir$ critical
cdir$ end critical
    times(i) = irtc() - t1
enddo
mean = compute_mean(times,ntests)
max_mean = max(mean)
```

As with the lock performance test, this test finds the maximum of the averages across all processors in order to compute the mean time into the critical region for the processor who was most often the last to arrive. The results from this test follow:

<u>Number of PEs</u>	<u>Max. Mean Time (μsecs)</u>
1	5.7
8	26.0
32	85.0
64	184.0
128	394.0
256	945.0

The time for one processor to execute CRITICAL/END CRITICAL is almost identical to the time for one processor to execute SET_LOCK/CLEAR_LOCK. However, heavily contended critical regions scale much better with the number of processors than do the lock routines.

3.4 Events

Events provide a method of program synchronization that is used to communicate the state of execution of one task to other tasks. The event operations are supported by four subroutines which can be called to execute in memory or eureka mode:

```
CALL SET_EVENT([event])
CALL WAIT_EVENT([event])
CALL CLEAR_EVENT([event])
S = TEST_EVENT([event])
```

SET_EVENT sets, or posts, an event. It declares the event to have occurred. WAIT_EVENT suspends task execution at a cleared event until the specified event is posted by SET_EVENT. CLEAR_EVENT clears the event. TEST_EVENT returns the state of the event: .TRUE. if set, .FALSE. otherwise. The argument to the event routines is optional. If an argument is supplied to the event routines, it must be a shared integer variable or array element. (Note that this restriction implies that memory mode events are only available to CRAFT programmers.) If these routines are called without an argument, then the hardware eureka mechanism is used for event communication.

3.4.1 Design Considerations

There are two ways that the event functions can be used: in memory mode or in eureka mode. In memory mode, the argument to the event routines must be a shared integer variable. If the event routines are called without an argument, then they use the T3D hardware eureka mechanism for event communication. Depending on the mode, the event routines are used in slightly different ways.

3.4.2 Memory Mode Events

Like the lock routines, memory mode events will use the hardware atomic swap mechanism and automatic cache line invalidation. The sign bit of the event word will be used for event communication. That is, SET_EVENT will set the sign bit in the event word and CLEAR_EVENT will clear the sign bit. As with traditional Cray macrotasking, CLEAR_EVENT is usually called immediately after the call to WAIT_EVENT.

The memory mode event routines implement a binary tree wait queue. A binary tree is used to shorten the time it takes to wake up the last arriving processor from N to $O(\log_2 N)$ where N is the number of processors blocked on an event. Another property of using a binary tree structure is that multiple processors can place themselves onto the tree concurrently. That is, one processor can be enqueueing itself on the right side of the tree while another processor is enqueueing itself on the left.

The data structures used to implement this wait queue are the same data structures used for the lock routines. The difference between events and locks is that the event routines place two PE numbers in each element of wait queue in order to implement a binary tree. The enqueueing algorithm is designed to insure that the tree is always balanced.

3.4.3 Eureka Mode Events

Eureka mode events use the T3D eureka mechanism. This mechanism requires that all processors participate. Specifically, *all* processors must call CLEAR_EVENT before any processor can test, wait, or set the eureka event. In eureka mode, CLEAR_EVENT is used to initialize the eureka barrier bit to the cleared state. It does this by setting the eureka and barrier bits and then waiting at the barrier until all processors have armed their eureka bits. Although it is necessary for all processors to clear the eureka mode event before beginning the eureka activity, it is not necessary for all processors to ever wait for the eureka event to be set. However, before another eureka activity can be started, all processors must once again call CLEAR_EVENT to initialize the eureka barrier bit.

Since eureka mode events do not rely on shared memory, this event mechanism is made available to C applications through the following routines:

```

void set_event(void);
void wait_event(void);
void clear_event(void);
int test_event(void);

```

These routines are used exactly like their Fortran counterparts with the exception that the C versions take no arguments and only execute in eureka mode.

3.4.4 Event Algorithm

The general algorithms for the memory-mode portion of SET_EVENT and WAIT_EVENT follow. Many of the details have been left out for clarity and brevity. Note that since the event word is the root node of the wait queue, it has to be special cased for the first two processors blocking on the event; All other processors will write to the wait queue.

```

SET_EVENT:
while ((event = atomic_swap(event,BUSY)) == BUSY)
  /* no delay favors SET_EVENT over WAIT_EVENT */
  continue;
if (event.state != POSTED) {
  if (event.left != EMPTY)
    local_event[event.left] = POSTED;
  if (ev->right != EMPTY)
    local_event[event.right] = POSTED;
  event.right = event.left = EMPTY;
  event.state = POSTED;
  flush the write buffer;
  wait for remote writes to complete;
}
event[0] = ev;

```

```

WAIT_EVENT:
while ((event = atomic_swap(eventword,BUSY)) == BUSY)
  delay(); /* wait a bit before trying again */
if (event.state == CLEARED) {
  pe_state = UNASSIGNED;
  do { /* add PE to the binary tree wait queue */
    if (event.q_state == LEFT) {
      event.q_state == RIGHT;
      if (event.left == EMPTY) {
        event.left = mype;
        pe_state = ASSIGNED;
        /*
         * Set up local wait word
         * and wait in cache.
         */
        local_event[mype] = BLOCKED;
        [eventword | wait_queue] = event;
        while (local_event[mype] == BLOCKED)
          continue; /* spin in cache */
        /*
         * Event occurred; Wake up PEs in the next node.
         */
        while ((next_node = atomic_swap
          (wait_queue[mype],BUSY)) == BUSY)
          continue;
        local_event[next_node.right] = POSTED;
        local_event[next_node.left] = POSTED;
      } else { /* get next node in wait queue */
        while ((next_node = atomic_swap
          (wait_queue[event.left],BUSY)) == BUSY)
          continue;
        /*
         * Next node is BUSY so free the node above.
         */
        [eventword | wait_queue] = event;
        event = next_node;
      }
    } else {
      ev->q_state = RIGHT;
      (same algorithm as for LEFT using RIGHT)
    }
  } while (pe_state != ASSIGNED);
} else {
  event[0] = ev; /* event is already posted */
}

```

3.5 Event Performance

3.5.1 Eureka Mode Events

The routine CLEAR_EVENT executes in 1.5 μ secs when executed in eureka mode which, not coincidentally, is the same time as executing the BARRIER routine. This makes sense since CLEAR_EVENT clears the eureka bit and then waits at a barrier until all processors have cleared their eureka

bit. The clearing of the eureka and the setting of the barrier bit occur in the same store instruction.

The SET_EVENT routine executes in 1 µsec. This includes the time to flush all writes to memory, check on the completion of remote writes, overhead from deadlock detection, and debugging support.

The routine WAIT_EVENT executes in 1.5 µsecs. This should be expected since waiting on a eureka and waiting on a barrier are the same thing as far as the algorithm and the hardware are concerned.

3.5.2 Memory Mode Events

The memory mode version of CLEAR_EVENT executes in 2.5 µsecs.

The SET_EVENT routine execution time depends upon the number of processors blocked on the event as follows:

<u>Number of blocked PEs</u>	<u>Mean Time (µsecs)</u>
0	3.5
1	4.5
> 1	5.0

Currently, it takes approximately 5 µsecs to execute WAIT_EVENT when the event is already posted. This time should be around 2.9 µsecs; It appears that this poor performance is being caused by inefficiencies in the MPP C compiler. This should be fixed by our first release.

When timing the WAIT_EVENT routine when the event is not posted, what is actually measured is the time to empty the entire event wait queue. The following section of code was used to measure this:

```

do i = 1, ntests
  call barrier()
  if (mype .eq. (n$pes - 1)) then
    delay = irtc() + (n$pes * 2000)
3    if (irtc() .le. delay) goto 3
    t1 = irtc()
    call set_event(evar)
    call barrier()
    times(i) = (irtc() - t1) - barrier_wait_time
    call clear_event(evar)
  else
    call wait_event(evar)
    call barrier()
  endif
enddo
if (mype .eq. (n$pes - 1)) then
  mean = compute_mean(times, ntests)
endif

```

The results from this experiment follow:

<u>Number of blocked PEs</u>	<u>Mean Time (µsecs)</u>
1	11.2
3	15.5
7	19.3
15	22.6
31	26.3
63	29.3

This table shows that not only are memory mode events fast, they also scale very well with the number of processors.

4.0 Broadcast Communication

The COPY parameter of the END MASTER directive generates a call to a library broadcast routine which broadcasts the data from processor 0 to all other processors. This broadcast routine implements two algorithms: one for low-latency, small-word-count transfers called the bisection broadcast and another for high-bandwidth, large-word-count transfers called the pipelined broadcast. When the broadcast routine is called, it chooses the appropriate algorithm to execute based on the number of processors participating in the broadcast and the number of words to broadcast.

The bisection broadcast algorithm works as follows (“→” indicates a write operation of the broadcast data):

First time step:

```
processor[0] → processor[n$pes/2]
```

Second time step:

```
processor[0] → processor[n$pes/4]
processor[n$pes/2] → processor[3n$pes/4]
```

Third time step:

```
processor[0] → processor[n$pes/8]
processor[n$pes/4] → processor[3n$pes/8]
processor[n$pes/2] → processor[5n$pes/8]
processor[3n$pes/4] → processor[7n$pes/8]
```

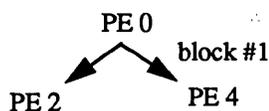
etc.

This algorithm seems to be most efficient when the number of processors participating in the broadcast is less than or equal to 32 or when the number of words to broadcast is less than approximately 1024 words. It should also be noted that this algorithm is fairly insensitive to the shape of the partition (the number of processors in each of the X, Y, and Z directions on the physical torus).

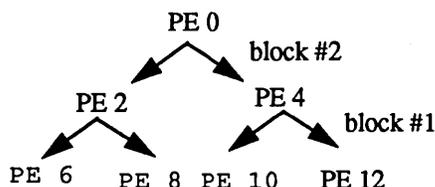
The pipelined broadcast algorithm is a pipelined node broadcast where the even numbered processor in each

node sends a subset or block of the broadcast data to the even numbered processor in the next two nodes in a binary tree. The number of words in each block of broadcast data written depends on the number of processors participating in the broadcast and the number of words to broadcast. Once a processor receives the first block of broadcast data, it immediately begins to send it to the next node in the broadcast tree. As the broadcast continues, soon all of the even numbered processors are sending blocks of data to their children in the broadcast tree concurrently. Once each even numbered processor has finished sending all of its broadcast data to its two even numbered children, it then sends all of the data to its internode, odd-numbered processor partner in one large block. An illustration follows. The numbers shown are virtual processor numbers.

First Time Step:



Second Time Step:



Last Time Step:

```

processor[0] → processor[1]
processor[2] → processor[3]
processor[4] → processor[5]
etc.
  
```

This algorithm is most efficient when more than 32 processors are participating in the broadcast and when the number of words to broadcast is larger than 1024. The drawback to this algorithm is that it is fairly sensitive to the shape of the partition. It is fastest when the partition is shaped as a perfect cube.

4.1 Broadcast Performance

The following table shows the time to complete a broadcast for different numbers of processors and different numbers of words.

PEs	Mean Time (µsecs)			
	1 word	128 words	1k words	16 K words
32	11.1	53.9	337	5162
64	12.7	64.1	404	4836
128	14.2	74.1	470	5001
256	16.0	83.8	538	6770

The shapes of the partitions for this experiment were:

Partition Size (PEs)	Partition Shape (X x Y x Z)
32	8 x 2 x 2
64	8 x 2 x 4
128	8 x 4 x 4
256	16 x 4 x 4

The broadcast routine is, in general, very fast. An anomaly in the performance data does exist where it is faster to broadcast 16K words of data to 128 processors than it is to broadcast the data to 32 processors. This is because the bisection broadcast algorithm is fastest for all word sizes when the broadcast involves just 32 processors; The start-up costs for the pipelined broadcast is too large for so few processors. However, the pipelined broadcast algorithm is able to broadcast 16K words to 128 processors faster than the bisection broadcast algorithm can broadcast 16K words to 32 processors.

5.0 Conclusion

The hardware support for the synchronization routines was briefly described and the algorithms used to implement barriers, locks, events, and the broadcast routines were discussed. The trade-offs were presented between using locks and critical region directives and between using eureka or memory mode events. It is clear from the performance results presented that synchronization and broadcast communication on the Cray T3D Computer System is very efficient.

6.0 References

[DEC91b] *The EV3 AND EV4 SPECIFICATION - DC227, DC228*, Version 2.0, May 3, 1991. Digital Equipment Corp.

[Pa93] *MPP Fortran Programming Model*, Douglas M. Pase, Tom MacDonald, and Andrew Meltzer. CRAY Internal Report, February 1993. To appear in "Scientific Programming," John Wiley and Sons.

[Fr92] *Fast Mechanisms for Quasi Synchronization*, P. Frederickson, K. Lind. Draft 1.0, Technical Report, August 21, 1992. Cray Research Inc., Los Alamos, NM.

High Performance Programming Using Explicit Shared Memory Model on the Cray T3D¹

Subhash Saini² and Horst D. Simon
Numerical Aerodynamic Simulation Facility
NASA Ames Research Center, Mail Stop 258-6
Moffett Field, CA 94035-1000, USA

email: {saini, simon}@nas.nasa.gov

Phone: S. Saini: 415-604-4343, H.D. Simon: 415-604-4322

FAX: 415-604-4377

and

Charles Grassl

Cray Research, Inc.

655F Lone Oak Drive

Egan, MN 55121

Phone: 612-683-3531

FAX: 612-683-5599

email: cmg@magnet.cray.com

Abstract

The Cray T3D system is the first-phase system in Cray Research, Inc.'s (CRI) three-phase massively parallel processing (MPP) program. This system features a heterogeneous architecture that closely couples DEC's Alpha microprocessors and CRI's parallel-vector technology, i.e., the Cray Y-MP and Cray C90. An overview of the Cray T3D hardware and available programming models is presented. Under Cray Research's Adaptive Fortran (CRAFT) model four programming methods (data parallel, work sharing, message-passing using PVM, and explicit shared memory model) will be available to the users. However, at this time data parallel and work sharing programming models are not yet available. The differences between standard PVM and CRI's PVM are highlighted with performance measurements such as latencies and communication bandwidths. We have found that the performance of neither standard PVM nor CRI's PVM exploits the hardware capabilities of the T3D. The reasons for the less than optimal performance of PVM as a native message-passing library on T3D are presented. This is illustrated by the performance of the NAS Parallel Benchmarks (NPB) programmed in the explicit shared memory model on the Cray T3D. In general, the performance of standard PVM is about 4 to 5 times less than what can be obtained by using the explicit shared memory model. The issues involved (such as barriers, synchronization, invalidating data cache, aligning data cache etc.) while programming in the explicit shared memory model are discussed. Performance data for the NPB using the explicit shared memory programming model on the Cray T3D and other highly parallel systems such as the TMC CM-5, Intel Paragon, Cray C90, IBM-SP1, etc. is presented.

1. S. Saini and H.D. Simon are employees of Computer Sciences Corporation. This work is supported by NASA Ames Research Center through contract NAS-129461.

2. Presenting author.

1: Introduction

The introduction of the CRAY T3D by Cray Research Inc. (CRI) in late 1993 has been a significant event in the field of massively parallel supercomputing. The T3D promises a major advance in highly parallel hardware with respect to a low latency (1 micro second) and high bandwidth (125 MB/sec) interconnect. For comparison, latency for Intel's Paragon is 120 micro seconds under Open Software Foundation (OSF)/1 AD and 90 micro seconds under Sandia University of New Mexico Operating System (SUNMOS). For the Paragon the communication bandwidths under OSF/1 AD and SUMOS are 35 MB per second and 170 MB per second respectively [1]. The latency for CM-5 is about 80 micro seconds and communication bandwidth is 9 MB per second [1]. It is clear that the latency and communication bandwidth of T3D are much better compared to Paragon and CM-5. The low latency and high communication bandwidth make the T3D scalable up to the maximum number of processing elements (PEs) that are available today, which is 256 PEs at the time of writing [1]. This scalability is clearly shown by the Class A NAS Parallel Benchmarks (NPB) for Processing Elements (PEs) ranging from 32 to 256 [2-5]. What makes the T3D unique is its ability to globally and non-uniformly address the entire memory without the interruption of remote processors using an explicit shared memory model. In the T3D, a PE can access its local memory about 3 to 4 times faster than memory of remote PEs. In Section 2, the overview of Cray T3D including macro architecture and micro architecture is presented. Section 3 describes the memory hierarchy of the Cray T3D. In Section 4, we present the shared distributed memory model of the T3D. In Section 5, operating system and compilers are discussed. In Section 6, CRAFT programming model is discussed. Section 7 contains CRI's extensions to the standard PVM. Section 8 contains a discussion of the explicit shared memory model. Section 9 is a brief description of the NPB. Section 10 contains the results and discussion and Section 11 presents our conclusions.

2: Cray T3D Overview

CRAY T3D has a two-tier architecture consisting of *macro architecture* and *micro architecture* [6, 7]:

2.1 Micro architecture The *micro architecture* will vary as technologies advance to achieve *Tflops/s* of sustained performance. Micro architecture refers to the microprocessor chip used in the Cray T3D. CRI is committed to use most appropriate (performance, features, technology and

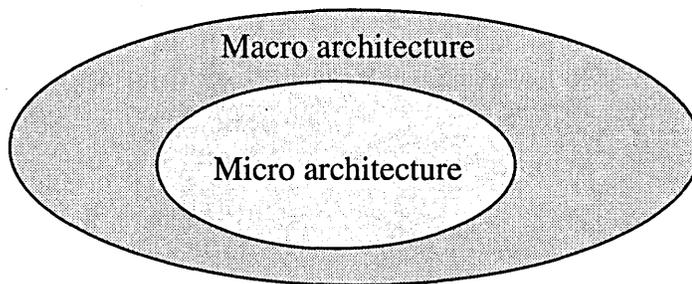


Figure 1: Two-tier architecture of Cray T3D.

availability) microprocessor in each generation. The DECchip 21064 (also called Alpha chip) used in the Cray T3D consists of four independent functional units. The block diagram of the Alpha microprocessor is shown in Figure 2. The Alpha chip consists of four main components IBOX, EBOX, FBOX and ABOX. A brief description of each box is given below [6-8].

(a) **Central control unit (IBOX):** The IBOX performs instruction fetch, resource checks, and

dual instruction issue to the EBOX, ABOX and FBOX or branch unit. It also handles pipeline stalls, aborts and restarts.

(b) Integer execution unit (EBOX): The EBOX contains a 64-bit fully pipelined integer execution data path including adders, logic box, barrel shifter, byte extract and mask, and independent integer multiplier. In addition, it also contains a 32 entry 64-bit integer register file (IRF).

(c) Floating point unit (FBOX): The FBOX contains a fully pipelined floating point unit and independent divider, supporting both IEEE and VAX floating point data types.

(d) Load/store or address unit (ABOX): The ABOX contains five major sections: address translation data path, load silo, write buffer, data cache (DCACHE) interface and external bus interface unit.

2.1.1 Pipeline organization:

The Alpha chip uses a seven stage pipeline for integer operation and memory reference instructions, and a six stage pipeline for a floating point operation instructions. The IBOX maintains state for all pipeline stages to track outstanding register writes.

2.1.2 Cache organization:

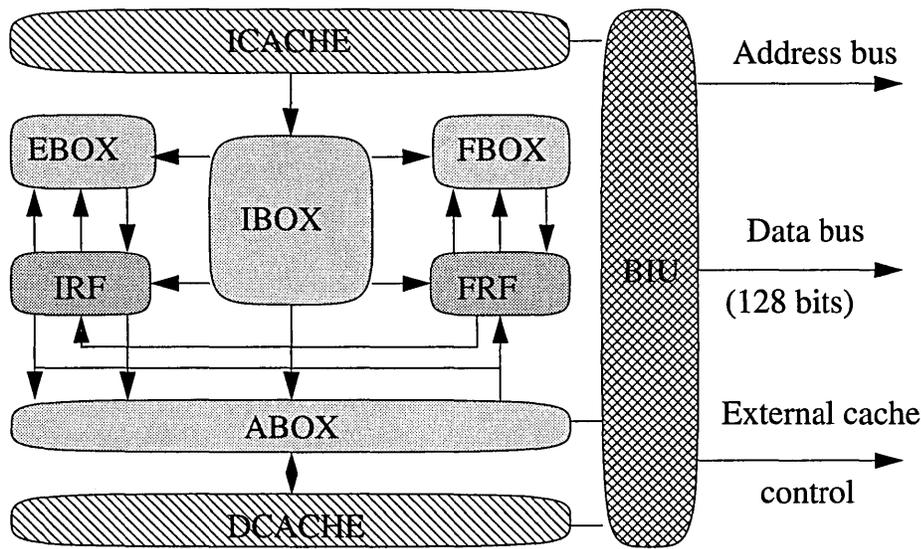


Figure 2: Block diagram of DECchip 21064 Alpha Microprocessor.

The Alpha chip contains two on-chip caches; data cache (DCACHE) and instruction cache (ICACHE). The chip also supports secondary cache, but it is not used in the version utilized in the T3D. The data cache contains 8 KB and is a write through, direct mapped, read-allocate physical cache with 32-byte blocks. The data cache is "direct mapped", unlike the Intel iPSC/860 or Paragon where it is "set associative". A direct cache has only one image of a given cache line. It is "read allocate" which means that entries into the cache only happen as result of a cacheable load from local memory. During a cache hit data is loaded into register from DCACHE and during a cache miss one cache line is loaded from DRAM.

The instruction cache is 8 KB and is a physical direct-mapped cache with 32-byte blocks. The Alpha chip supports secondary cache built from off-the-shelf static RAMs although it is not used in the T3D. The chip directly controls the RAMs using its programmable secondary cache interface, allowing each implementation to make its own secondary cache speed and configuration trade-offs. The secondary cache interface supports cache sizes from 0 to 8 MB and a range of operating speeds which are sub-multiples of the chip clock. The virtual address is a 64-bit unsigned integer that specifies a byte location within the virtual address space. The Alpha chip checks all

64-bits of a virtual address and implements a 43-bit subset of the address space and so supports a physical address space of 16 GB.

2.1.3 Features and Specifications of Alpha Chip:

The main features and specifications of the Alpha chip are given in Table 1. The content of a register is available immediately. A word in a data cache is available in three clocks and every subsequent word in its cache line needs another one clock. The chip takes 20 clocks to access data from its local memory. To achieve high performance on a single PE, one should always use small strides and short loops. The Alpha chip uses IEEE format for its floating point representation. If the application is distributed between the T3D and the Cray YMP/C90, one has to convert the data to the correct representation and this should always be done using very fast routines available on Cray YMP/C90 [6-8].

Table 1: Specifications for The DECchip 21064 Alpha microprocessor

Characteristics	Specification
Technology	CMOS, 0.75 micron
Transistors count	1.68 million
Physical dimensions	1.4 by 1.7 cm
Number of signal pins	291
Cycle time	150 MHz (6.67 nano second)
On-chip D-cache	8 KB, physical, direct mapped, write-through, 32 byte line, 32-byte fill
On-chip I-cache	8 KB, physical, direct mapped, 32 byte line, 32-byte fill, 64 ASNs
On-chip DTB	32-entry; fully associative, 8-KB, 4-MB page sizes
On-chip ITB	fully associative, 8 KB page plus 4-entry, fully-associated, 4-MB page
Latency of data cache to memory	3 clock periods (20.01 ns)
Bandwidth of data cache to memory	64 bit per clock period
Floating-point unit	On-chip FPU supports both IEEE and VAX floating point
Bus	Separate data and address bus 128-bit/64-bit data bus
Serial ROM interface	Allows the chip to directly access serial ROM
Virtual address size	64 bit checked; 43 bits implemented
Physical address size	34-bits implemented
Page size	8 KB
Issue rate	2 instructions per cycle to A-box, E-box, or F-box
Integer pipeline	7-stage pipeline
Floating pipeline	10-stage pipeline
Number of floating point registers	32
Size of floating point register	64 bit
Number of integer register	32
Size of integer register	64-bit

2.1.4 Single PE Optimization

Currently, optimizing a code for a single PE on the Cray T3D is much more difficult than for the C90 single CPU because of the following: optimizations are state dependent, data locality is always the issue, bandwidth is a limitation factor, not as many functional units are pipelined, compilers and various tools are not as mature as those available on C90. The following are the problems associated with the Alpha chip used in Cray T3D: **(a)** all memory operations stall upon cache miss, **(b)** the slow external bus makes the DRAM bandwidth suboptimal, **(c)** there are no integer to floating point or SQRT instructions, **(d)** divide and integer multiply are non-pipelined. A division operation produces one result every 64 clock periods and integer multiply produces one result every 20 clock periods.

Every DRAM request results in a cache line load of four 64-bit words - one for the actual request and the other three words which are mapped to the same cache line. Aligning data on a cache line boundary (word 0 of any cache line) enhances the performance. The cache alignment can be done by using a compiler directive `CDIR$ ALIGN_CACHE`. The SAXPY operation $Y(I) = Y(I) + \alpha * X(I)$ would perform better assuming X and Y are cache aligned and are on different cache lines. Performance can also be enhanced by scalar replacement, by holding the value of a temporary scalar in a register to reduce the number of memory accesses. Cache utilization can also be enhanced by loop interchange so that stride in the inner loop is one. Large stride in the inner loop causes cache misses. The DRAM memory of the Alpha chip is interleaved and one should ensure page boundary alignment. Page hit occurs when either current and previous references are to the same even or same odd page, or current and previous reference have different chip select (cs) bit. Page miss occurs when current and previous references are to different even or different odd pages. Page hits take 8 clock periods whereas a page miss takes 2 clock periods. For details on single PE optimization of T3D see reference [1].

2.2 Macro architecture

The *macro architecture* will remain the same in all the three phases of *MPP* project. The macro architecture will be stable from one generation to the next in order to preserve the applications development investment of the users. In addition to two PEs and a Block Transfer Engine, each computational node has support circuitry which includes Data Translation Buffers (DTB), Message Queue Control (MQC), Data Prefetch Queue (DPQ), Atomic Swap Control (ASC), Barrier Synchronization Registers (BSR), and PE Control. For details see references [6,9].

2.2.1 Block Transfer Engine:

Each computational node has two identical PEs which function independently of each other. Each node has support circuitry including but not limited to network interface, network router and block transfer engine (BTE). The network interface formats the information and the network router deformats it before sending it to either PE 0 or PE 1. BTE is asynchronous and is shared by the two PEs. It can move data independently without involving either the local PE or the remote PE. It also provides gather-scatter functionality in addition to data pre-fetch with a constant stride. It can transfer data up to 64 K words and can be used to select PE number and memory offset bits using the virtual global memory address facility. The use of BTE requires making system calls. It also involves performing local work first, and then double buffering the remote data transfers and working on those buffers. However, the start up time for BTE is very high.

2.2.2 Interconnect Network Topology

Each computational node is connected to other nodes via a three dimensional torus interconnect network as shown in Figure 4. This network operates at 150 MHz, identical to the clock of the Alpha chip used in Cray T3D. The network is 16 bits wide and can send simultaneously bi-

directional traffic in all three directions (X, Y and Z). The T3D network transmits system control information and user data. The control packets vary in size from 6 to 16 bytes. The data packets range in size from 16 bytes to 52 bytes. The amount of data in these packets is 8 or 32 bytes with the remainder being header and checksum information. The headers and checksums contribute to a load factor which affects attainable data transfer rates.

3: Memory Hierarchy

The memory hierarchy very strongly impacts the performance of an algorithm both at global and local levels. Normally, one would expect this kind of optimization to be handled by the compiler; however, the T3D user still needs to worry about organizing his data accesses so that the compiler can recognize opportunities for memory related improvements.

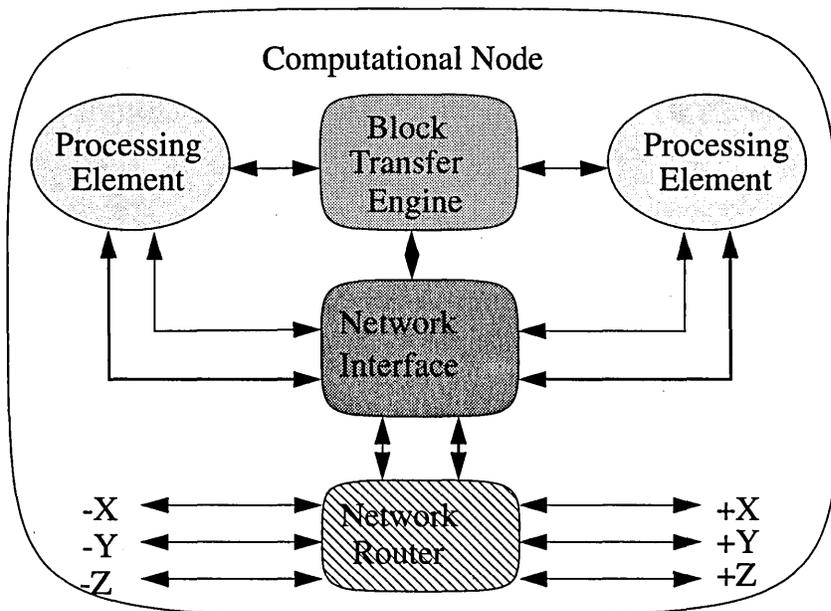


Figure 3: Computational node of Cray T3D.

The T3D memory hierarchy has the following four layers as shown in Figure 5.

(a) Data Cache: Data cache is a small, high speed random access memory that temporally stores frequently or recently accessed local data without user's intervention. The data cache is 8 KB that can access data from the memory at the rate of 1200 MB per second.

(b) Local Memory: Local memory consists of DRAM with read bandwidth of 320 MB per second per PE. The memory chips are organized in banks to read or write multiple words at a time rather than at a single words. Memory is interleaved in blocks of four 64 bit words to permit cache line memory actions. In other words, addresses of four banks are interleaved at a word level.

(c) Remote Memory: The Cray T3D is designed with memory physically distributed among all PEs but globally shared and addressable. The remote memory is 4(16) GB for a machine with 16(64) MB per PE. Using SHMEM_PUT function, it can be accessed at the rate of 125 MB per second. The cost of remote access is of the order of 4 to 5 times that of a local memory access not including overhead of PVM calls, etc.

(d) Secondary Memory: It consists of hard disks up to a TB and is connected to the T3D through a high speed channel (HISP). This can be used for out-of-core problems when the DRAM of T3D is not enough. The typical sustained speed of the DD-60 disks is about 20 MB per second.

The following principles and guidelines should be followed to exploit the memory hierarchy of the T3D: **(a)** use data that is readily available in the order register, data cache, local memory,

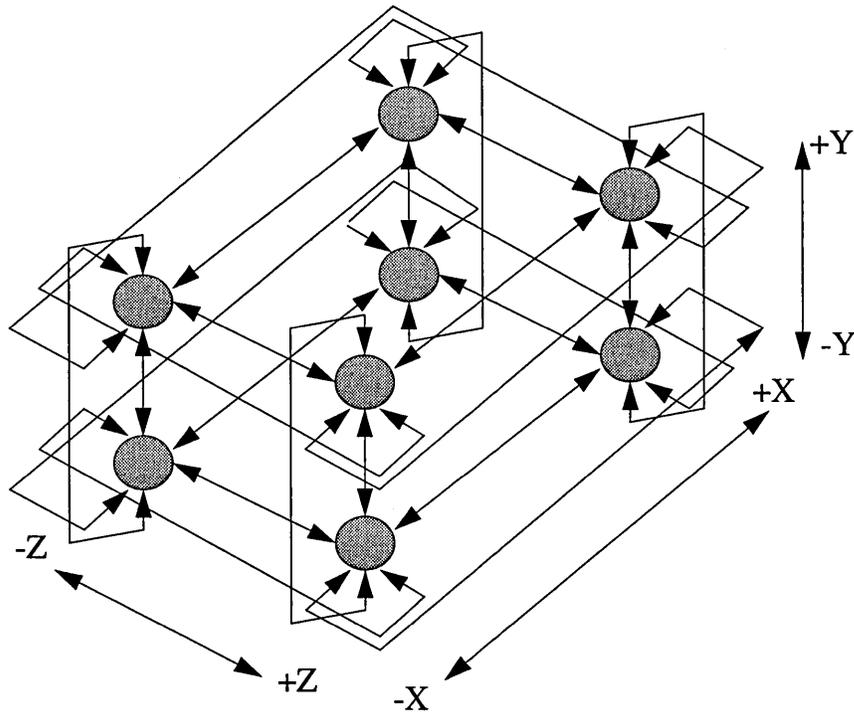


Figure 4: Interconnect topology for Cray T3D.

remote memory, and secondary memory, **(b)** load the data from local or remote memory much before it is needed and reuse that data, **(c)** hide latency with computation whenever possible [9].

4: Shared Distributed Memory

The Cray T3D is designed with memory physically distributed among all PEs, with each PE having a favored **(a)** low latency, high bandwidth path to a local memory, and **(b)** longer latency, lower bandwidth path over an interconnect network to remote memory associated with other PEs. Each PE uses memory addressing that references any word in shared memory. This address, the virtual address, is initially generated by the compiler. The virtual address is converted into a logical node number, PE number and address offset by the PE and other components in the computational node [6, 9, 10].

5: Cray T3D Operating System

UNICOS MAX 1.0 is a distributed operating system. UNICOS runs on the host Cray (YMP or C90) platform, whereas MAX is a microkernel residing on each PE of Cray T3D. MAX takes about 4 MB on each PE (MAX is based upon the Open Software Foundation (OSF) MACH 3.0 microkernel). Functionality which is not available in the microkernel is performed by UNICOS on the host system via message-passing using low speed channel (6 MB/sec) for control and high speed channel (200 MB/sec) for data. The operating system used is UNICOS MAX 1.0 and the compiler used is CF77_M 6.0. Latency for requesting data from a host YMP/C90 is 10 ms

involving seven system calls.

Table 2: Characteristics and specifications for Cray T3D system (see also [13]).

Characteristic	Specification	Comments
Local bandwidth	640 MB per second 320 MB per second	mem <----> PE
Cache memory latency	20×10^{-9} second	read-ahead
Local memory latency	90×10^{-9} second	no read ahead
Local memory latency	150×10^{-9} second	per node
Peak bandwidth	300 MB per second	
Switch bandwidth	2100 MB per second	per node
Bi-section bandwidth	76.8 GB per second	1024 PEs
PE <-----> PE latency (COMMS1) (COMMS2)	(1 message) 0.5-1.5 micro sec. (1 message) 4.0-5.0 micro sec. (2 message) 1.3 micro second	CRI MP PVM CRI MP
Data transfer rate of LOSP	6 MB per second	In each direction
Data transfer rate of HISP	200 MB per second	In each direction
DRAM memory	16 MB and 64 MB	4 - 16 MB chips
DRAM memory	19.2 - 38.4 GFLOPS per second	4 - 16 MB chips

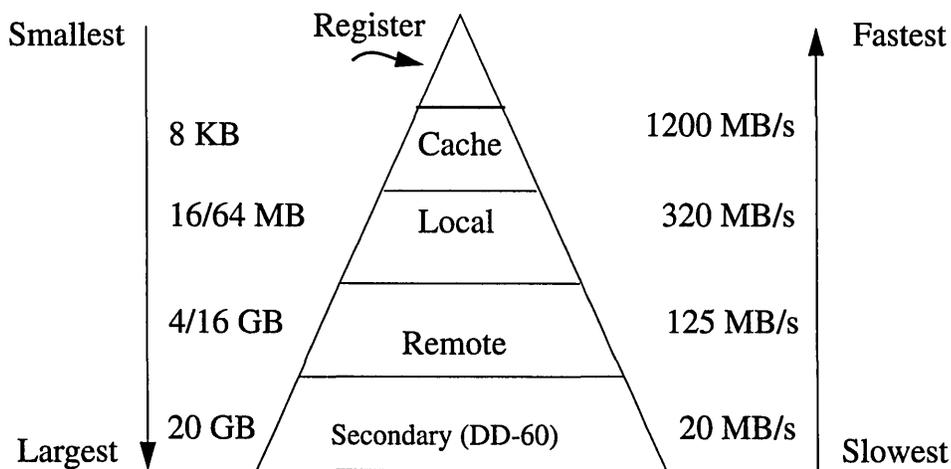


Figure 5: Cray T3D memory hierarchy.

6: Cray Research Adaptive Fortran (CRAFT) model

CRAFT derives its features from Fortran 77, Fortran 90, High Performance Fortran (HPF), Fortran D and PVM. The two programming models High Performance Fortran (HPF) and CRAFT are compatible with each other in the sense that HPF targets portability whereas CRAFT aims for high performance. CRI does not support HPF or Message Passing Interface (MPI) on the T3D but has adopted standard PVM and has optimized it and added several extensions to it for T3D.

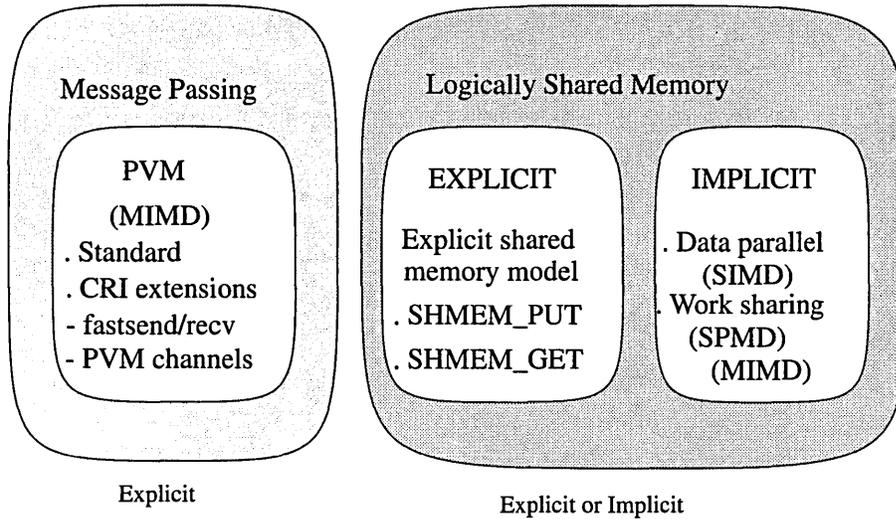


Figure 6: CRAFT programming model available on Cray T3D.

At this time, two programming models, data parallel and worksharing, are not available. So we are left with only two choices of programming models, namely message-passing and explicit shared memory model. Parallel Virtual Machine (PVM) is a public domain software and was developed at Oak Ridge National Laboratory and University of Tennessee. PVM was designed for loosely coupled network of workstations where network is much slower compared to the speed of the processors. CRI has taken this public domain PVM and extended it in several ways. One of the important feature of PVM is a process of buffering the messages both at the sending and receiving node. This feature is a boon for a slow network but is a curse in a fast network like T3D. It is a boon in slow network as no handshake is needed with the receiving node before the message is sent by the sender, and no acknowledgment is needed from the receiving node. The penalty of buffering is relatively small compared to the communication cost of the network. However, in the Cray T3D where the communication network is very fast, the penalty of buffering both at the sending node and receiving node is enormous. To overcome the cost of buffering the messages, CRI has provided a optimization feature called "PVMDDataInPlace". This feature should be used with a great caution. The memory should not be reused until after the data has been unpacked by the receiving PE. Importantly, the memory address should be accessible by the other PE which is true for COMMON or static DATA but may not be true for data allocated dynamically. Table 3 gives the latency and communication bandwidth for various types of communication available on

Table 3: Measured latencies and bandwidths.

Type of Communication	Latency in microseconds	Bandwidth in MB per second
PVM send and recv	60 - 80	30 ^a 40 ^b
PVM fastsend and fastrecv	15 - 20	10 ^c
PVM channels (send & recv)	5	100
SHMEM_GET ^d	1.5	30
SHMEM_PUT ^d	1.5	120

a PvmDataRaw packing

b PvmDataInPlace packing

c For 256 bytes - for small messages latency

d One need not match SHMEM_GET with SHMEM_PUT

7: CRI's PVM Extensions

Performance of standard PVM, CRI's PVM and shared memory GET and PUT is shown in Table 3. The **PvmDataRaw** packing includes an extra memory-to-memory operation as part of the **PACK** call [7,8]. The advantage of using standard PVM is that the code is portable; the disadvantage is that the performance is poor due to buffering of data both at sending PE and receiving PE. CRI has provided extensions to standard PVM to exploit the hardware features of T3D. The Cray MPP version of PVM provides a pair of functions that provide shorter latencies for short messages, called **fastsend** and **fastrecv**. Latencies for short messages for these **fastsend** and **fastrecv** functions are about 15 - 20 microseconds compared to about 60 - 80 microseconds for regular PVM **send** and **recv**. The **fastsend** and **fastrecv** functions are limited to 256 bytes of data (by default). This bandwidth is not very high, but still much higher than the bandwidth of standard PVM **send** and **recv**. Messages that contain more than 256 bytes must transfer the data in a second transfer. For details see reference [12].

7.1.1 PVM Channels on Cray T3D:

CRI provides new functions in PVM called PVM channels. The PVM channels are most useful for applications where communication is both regular and its pattern is repetitive. Two PEs establish a channel with a single data buffer and matching destination data buffer. Each transfer request specifies a channel, and transfer occurs at very high speed. PVM channel send/recv functions transfer data at about 100 MB per second with latencies of about 5 microseconds. For details see reference [12].

8: Explicit Shared Memory Model

Given the latency and communication bandwidth of standard PVM and CRI's extensions to PVM, it is clear that performance of the applications using standard PVM cannot exploit the hardware capability of the T3D. In view of this all the NPB were written in the explicit shared memory model using SHMEM_GET and SHMEM_PUT functions. The function SHMEM_PUT has latency of about 1.5 microseconds and communication bandwidth of 125 MB per second. Message-passing deals only with private addresses (private/local memory). In the explicit shared memory model, the user is responsible for explicitly managing the data transfers and communication and does so by explicitly specifying two addresses: a local address, and a remote

address which is really a tuple (PE number, local address). The two functions (SHMEM_GET and SHMEM_PUT) are used to read (GET)/write (PUT) from/to any remote PE and map directly onto the low level CRAY T3D concepts of loads and stores. Unlike message-passing, the use of these functions does not involve any PE (sender or receiver).

```
CALL SHMEM_GET(TARGET, SOURCE, LEN, PE)
```

```
CALL SHMEM_PUT(TARGET, SOURCE, LEN, PE)
```

TARGET - Address of array into which data is transferred. This must be a word address

SOURCE - Address of array from which data is transferred. This must be a word address.

LEN - Length, in words, of the array

PE - PE number of other PE

The function SHMEM_GET returns when the data has been loaded into the local data array and is thus a blocking function and uses remote loads. The function SHMEM_PUT returns when the data has been sent from the chip (but may not yet have arrived at the other PE) and is thus non-blocking and uses remote stores. For details see reference [11].

8.1 Knowledge of Remote Address

To use the functions SHMEM_GET and SHMEM_PUT one needs to know the addresses at both local and remote PEs. The easiest way to know the remote address is to use COMMON blocks in Fortran and global data or static data in C. On the T3D, such data is statically allocated at the same location on all PEs. Therefore, if one knows a local address of one of these variables on one PE, then one knows its local address on all PEs.

Example:

In C:

```
static double x[50], y[50]
shmemp_put(y, x, 50, mype+1);
```

In Fortran:

```
COMMON/abc/ x(50), y(50)
CALL SHMEM_PUT(y, x, 50, mype+1)
```

Copy 50 words of data from array *x* on this PE to array *y* on the PE numbered *mype+1*. The base addresses of arrays *x* and *y* are the same on the two PEs.

8.2 Issues involved in using explicit shared memory model

A user must be aware of the following two issues:

8.2.1 Data Cache Alignment

Alignment of COMMON blocks in data cache enhances the performance of the application. The reason for this is that first word takes about 3 clocks and every subsequent word in a cache line takes just one more clock. This alignment can be done by using a compiler directive CDIR\$ ALIGN_CACHE [10].

8.2.2 The Cache-coherency Problem

Because of the data cache, data can be found in memory or in the cache. As long as local PE is the only device changing or reading the data, there is a little danger in the PE seeing the old or stale copy. However, programming in explicit shared memory model, any PE can both read and write data to the memory of remote PEs. Remote PEs means the opportunity exists for other PEs to cause copies to be inconsistent or remote PEs to read the stale copies. This is generally referred

to as the cache-coherency-problem. In Figure 7 we have illustrated the problem associated with cache-coherency while programming in the explicit shared memory model. In the first column A' and B' refer to the cached copies of A and B in memory and therefore it does not matter whether PE reads the value from the memory or the data cache and we say that cache and main memory are in coherent state. In the middle column, let us assume that PE 1 writes-back 555 into A. Some other PE reads the value of A from the memory of PE 1 using SHMEM_GET. In this case, PE 1 would read the stale value of A and not the updated value of A (which is A') and such a situation is called cache-incoherent. In the last column, let us assume that PE 21 writes the value of B using SHMEM_PUT and PE 2 reads the value of B from the cache and it will read the stale value of B and not the updated value. In this situation, the cache of PE 2 should be invalidated [10].

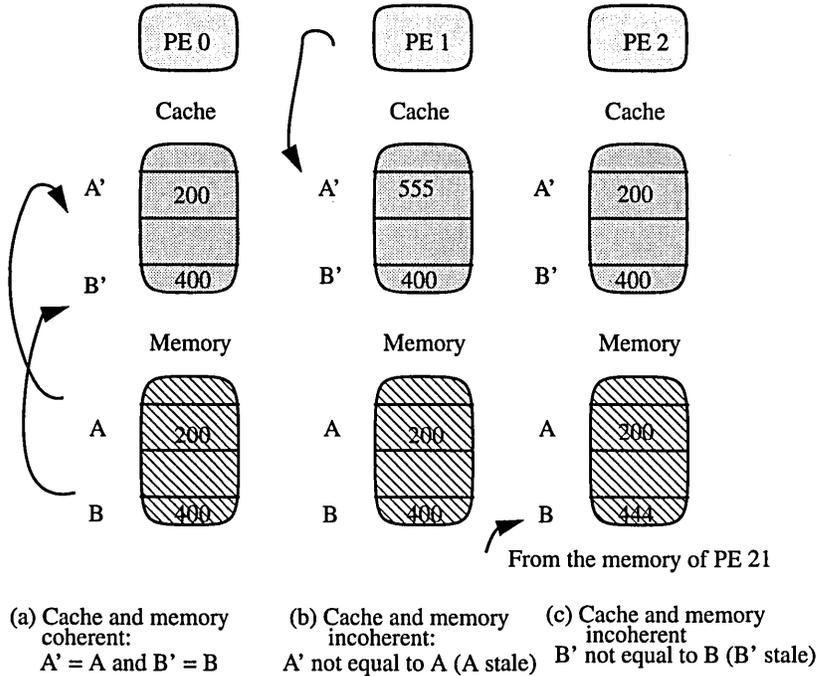


Figure 7: Cache coherency and invalidation of data cache.

9: NAS Parallel Benchmarks

The NPB were developed to evaluate the performance of highly parallel supercomputers. One of the main features of these benchmarks is their pencil and paper specification, which means that all details are specified algorithmically thereby avoiding many of the difficulties associated with traditional approaches to evaluating highly parallel supercomputers. The NPB consist of a set of eight problems each focusing on some important aspect of highly parallel supercomputing for computational aerosciences. The eight problems include five kernels and three simulated computational fluid dynamics (CFD) applications. The implementation of the kernels is relatively simple and straightforward and gives some insight into the general level of performance that can be expected for a given highly parallel machine. The other three simulated CFD applications need more effort to implement on highly parallel computers and are representative of the types of actual data movement and computation needed for computational aerosciences. The NPB all involve significant interprocessor communication with the exception of the Embarrassingly Parallel (EP) benchmark which involves almost no interprocessor communication [3-5].

10: Results and Discussion

The results and discussion are presented for two classes of NPB, namely class A and class B and they given separately.

10.1 Class A NPB Results

In Figures 8-15 are shown the class A NPB. All the results are normalized to one processor of Cray YMP and is denoted by Cray YMP/1. Note that IBM SP-1 results are for only 64 nodes. Figure 8 shows the results for EP. This benchmark gives an estimate of the upper achievable limits for floating point performance as it does not have any significant interprocessor communication except at the end to collect the results. This kernel uses two intrinsic function SQRT and LOG. Figure 9 are the results for the MG kernel, which solves Poisson equation with periodic boundary conditions and requires highly unstructured long distance communication and thus is a good test for communications bandwidth.

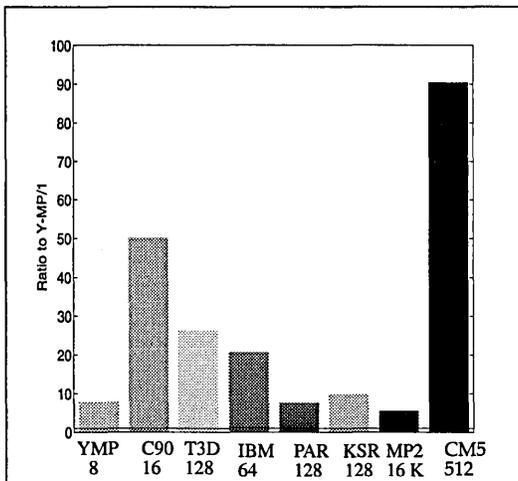


Figure 8: Results for EP class A benchmark.

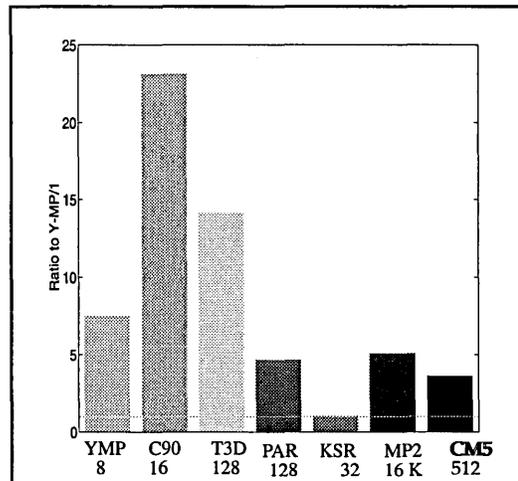


Figure 9: Results for MG class A benchmark.

In Figure 10 are shown the results for CG kernel. This kernel approximates the smallest eigenvalue of a large sparse, symmetric, positive definite matrix. This kernel tests irregular long-range communication and basic computations are done by sparse matrix-vector multiplication. Figure 11 gives the results for FT kernel and we find T3D performs the best.

Figure 12 shows results for IS kernel and uses a sorting operation that is useful in "Particle method" and it tests the integer computation speed and communication bandwidth. Figure 13 shows the results for LU, pseudo CFD application used in NASA code INS3D-LU. Here performance of T3D is the best among all MPPs.

Figure 14 shows the results for SP benchmark, which is the basic kernel in the NASA code ARC3D. It uses an implicit factorization scheme. This benchmark solves multiple independent systems of non-diagonally dominant, scalar pentadiagonal equations. Figure 15 shows the results for block tridiagonal (BT) benchmark that solves multiple independent systems of non-diagonally dominant, block tridiagonal equations with a block size of 5x5.

10.2 Class B Results

Figure 16 shows the results for EP. For C90 and T3D results are obtained by algebraic and table-lookup methods that will be disallowed in the future. The performance of T3D is better than C90. Figure 17 shows the results for MG kernel. Here the performance of T3D is very close to C90 and performance of the Paragon is about two processors of C90. Figure 18 shows the results of CG kernel. Performance of the Paragon is about half processor of the C90. Figure 19 shows FT

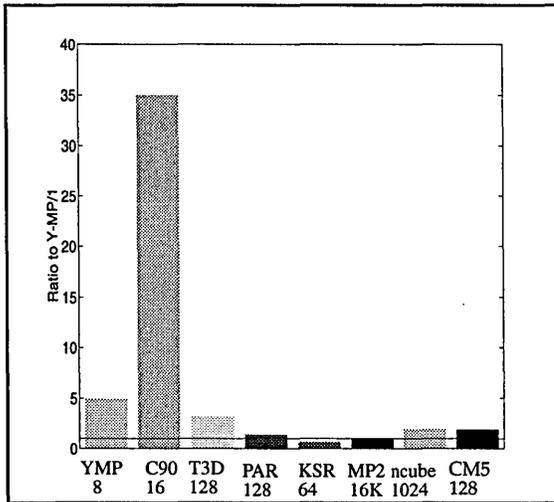


Figure 10: Results for CG class A benchmark.

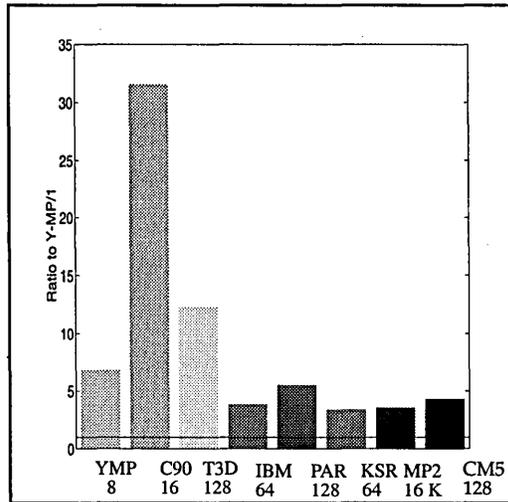


Figure 11: Results for FT class A benchmark.

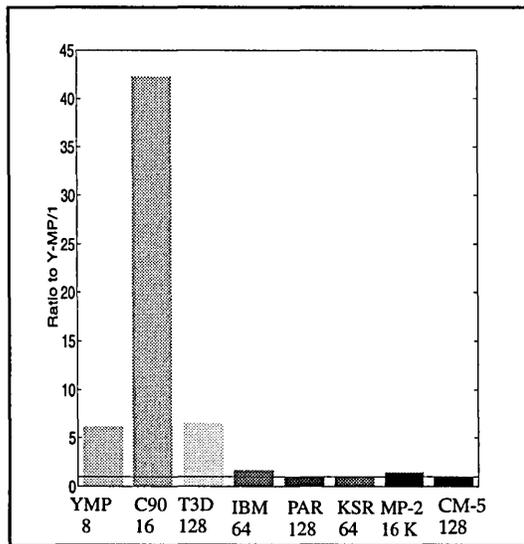


Figure 12: Results for IS class A benchmark.

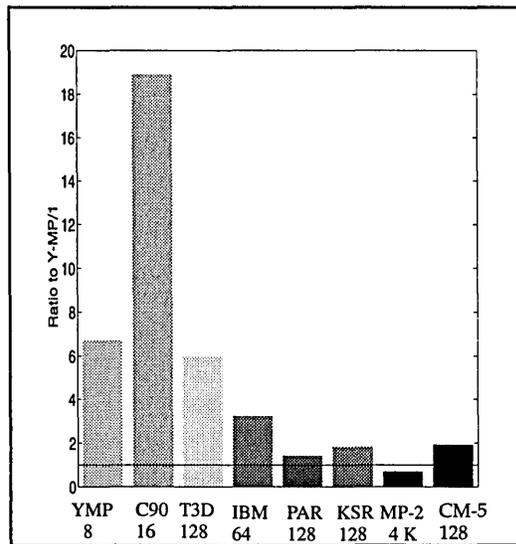


Figure 13: Results for LU class A benchmark.

kernel and all the machines use either 1-D, 2-D or 3-D assembly-coded routines. Figure 20 shows the results of IS on C90, T3D, Paragon and CM-5E. Figure 21 shows the results for LU. Here we notice that performance of Paragon is less than even one CPU of C90 and performance of CM-5E is slightly better than 1 CPU of C90. In Figure 22 are the results for SP and we notice that the performance of the Paragon is less than 2 CPU of C90 and for the CM-5E it is about 2 processors of C90. In Figure 23 are shown the results for BT benchmark.

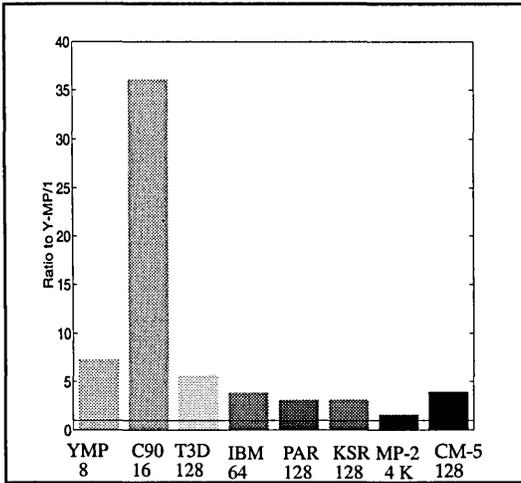


Figure 14: Results for SP class A benchmark.

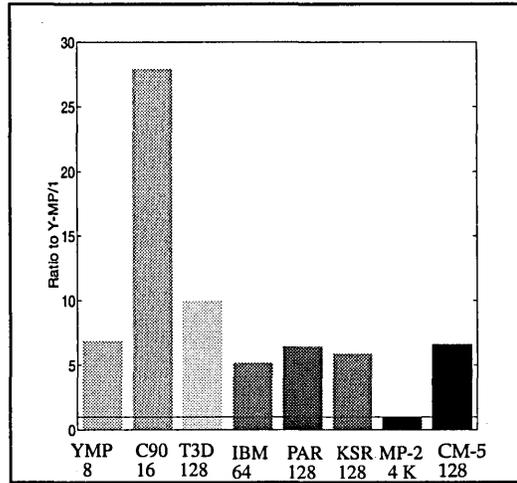


Figure 15: Results for BT class A benchmark.

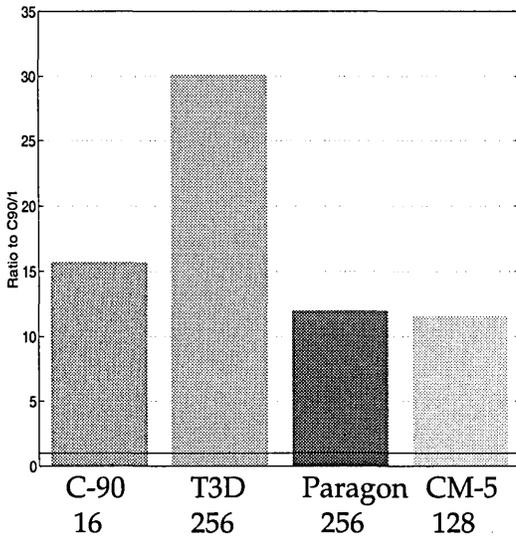


Figure 16: EP for class B NPB.

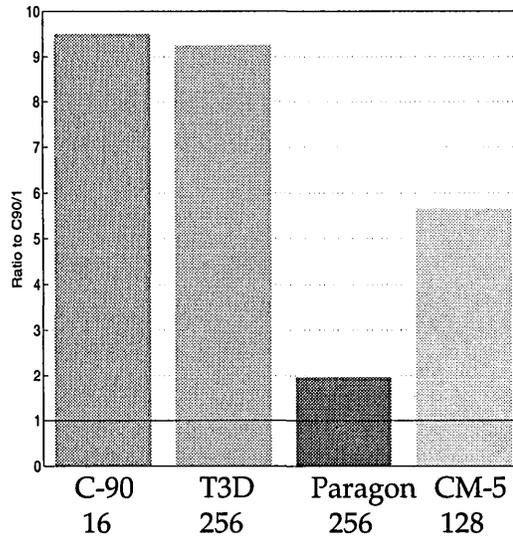


Figure 17: MG for class B NPB.

11: Conclusions

1. Out of four only two programming models (message-passing using PVM and explicit shared memory model) are available today on the Cray T3D. The other two programming models namely, data parallel and worksharing models are not yet available.
2. Performance of PVM is not very good due to the need of buffering of the messages both at sending PE and receiving PE, which involves making an expensive copy of the data from user space to the system space. Although the speed of the Cray T3D network is fast, copying is a relatively slow process. The overall performance of PVM is dictated by the time needed to buffer the messages at both ends involving PE.
3. The explicit shared memory model has the advantage that it offers better performance than message passing. The disadvantage is that the code is non-portable, however, the performance is about 4 to 5 times better than using PVM.

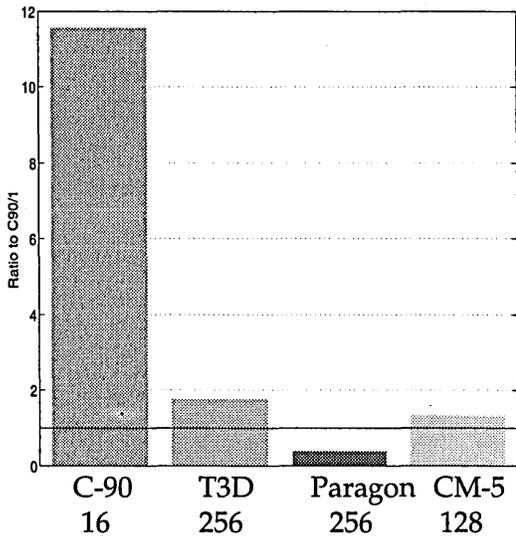


Figure 18: CG for class B NPB.

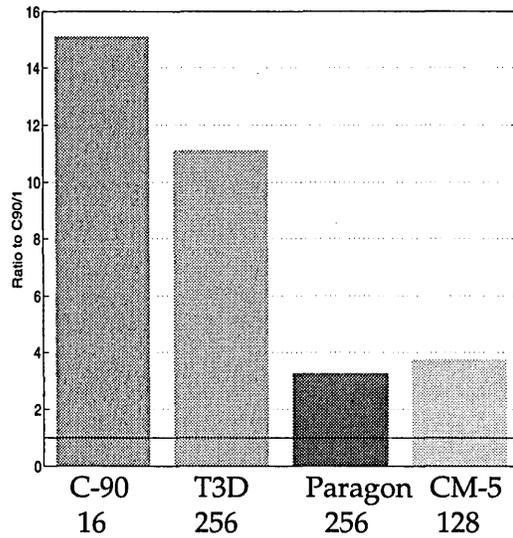


Figure 19: FT for class B NPB.

4. The explicit shared memory model should be used with great deal of care because of the following: (a) the cache-coherency problem, (b) cache boundary alignment problem, (c) barrier synchronization problem. Leaving out a necessary barrier synchronization introduces a race condition.

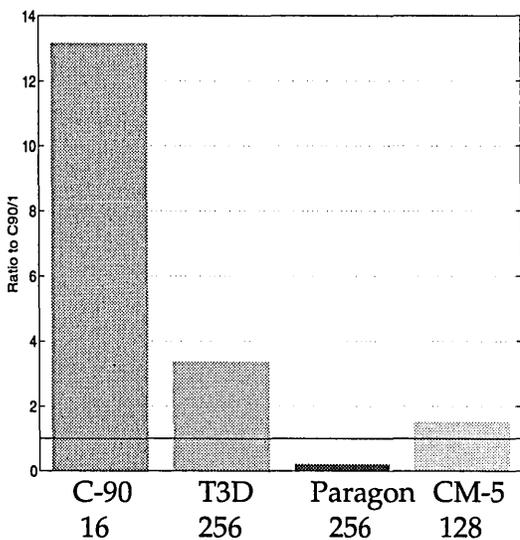


Figure 20: IS for class B NPB.

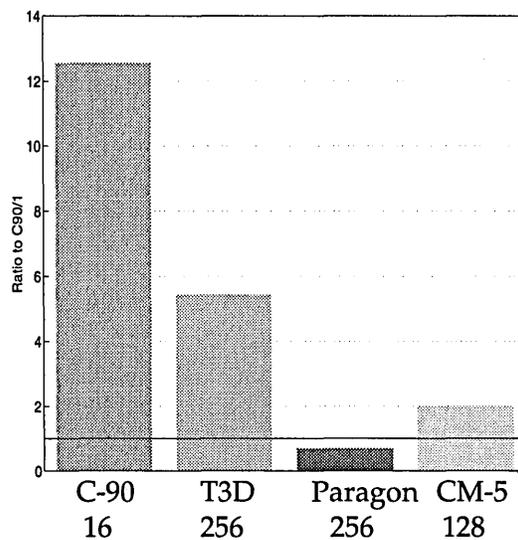


Figure 21: LU for class B NPB.

5. In order to obtain good performance, a user has to optimize the application for a single PE which requires a detailed knowledge of the memory hierarchy.

6. The function of Block Transfer Engine seems to be redundant as start up time is very large and the same functionality is provided by explicit shared memory model.

7. For both class A and class B NPB, Cray T3D outperforms both Paragon and CM-5E.

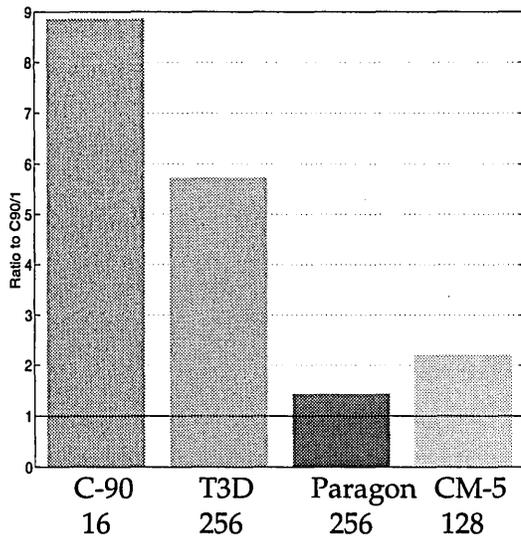


Figure 22: SP for class B NPB.

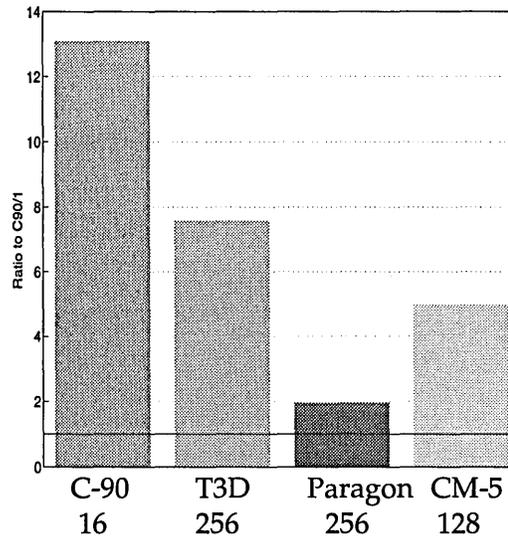


Figure 23: BT for class B NPB.

References:

- [1] S. Saini and H.D. Simon, to be published
- [2] S. Saini et al., Proceedings of Cray User Group Spring 94 Conference, March 14-18, 1994, San Diego, California.
- [3] D. Bailey et al., eds, *The NAS Parallel Benchmarks*, Technical Report RNR-91-02, NAS Ames Research Center, Moffet Field, California, 1991.
- [4] D. Bailey et al., The NAS Parallel Benchmark Results, IEEE Parallel & Distributed Technology, 43-51, February 1993.
- [5] D. Bailey et al., eds, The NAS Parallel Benchmark results 394, Technical Report RNR-94-06, NAS Ames Research Center, Moffet Field, California, 1994.
- [6] S. Saini, Overview of Cray T3D Hardware and Software, *NAS User Televideo Seminar*, Sept. 22, 1993, NASA Ames Research Center. For a free copy of the video tape send e-mail to saini@nas.nasa.gov.
- [7] *Alpha Architecture Handbook*, 1992, Digital Equipment Corporation.
- [8] DECchip™ 21064-AA Microprocessor Hardware Reference Manual, Order Number: EC-N0079-72, Digital Equipment Corporation October, 1992, Maynard, Massachusetts.
- [9] *CRAY T3D System Architecture Overview*, HR-04033, Cray Research, Inc.
- [10] S. Saini, "High Performance Programming Using Explicit Shared Memory Model on Cray T3D", *NAS, User Televideo Seminar*, Jan, 19, 1994, NASA Ames Research Center. For a copy of the video tape send e-mail to saini@nas.nasa.gov.
- [11] *Cray Research MPP Software Guide*, SG-2508 1.0 DRAFT - 11/15/93.
- [12] PVM and HeNCE Programmer's Manual, SR-2501 3.0, November 1993, Cray Research, Inc.
- [13] K. E. Gates and W. P. Petersen, A Technical Description of Some Parallel Computers, *Int. J. High Speed Computing*, 1994 (to appear).

Architecture and Performance for the CRAY T3D

Charles M. Grassl
Cray Research, Inc.
655A Lone Oak Drive
Eagan, MN 55121

cmg@cray.com

The architecture and performance of the CRAY T3D system is presented. Performance of the single PE, network and full system is examined using several benchmark tests. The effect of architectural features on programming constraints and methods is discussed.

I. Introduction

The CRAY T3D, introduced in September, 1993, is the beginning of a series of three MPP systems which have a common macro architecture. The basis of the common macro architecture is a 3D torus. The micro architecture is implemented in the EV2064 (Alpha) microprocessor from DEC.

We will present features and performance of the torus network and of the microprocessor. We will emphasize features which are of interest to programmers and users of the T3D. This discussion will highlight features which directly affect performance and we will also point out features which have little effect on performance.

II. Macro Architecture

The 3D torus is essentially a three dimensional grid with periodic boundary conditions. That is, nodes on opposite edges of the cube are connected. The 3D torus topology was chosen by Cray

Research for various reasons, including the following:

- Scaling properties.
- Low latency.
- High bisection bandwidth.
- Low contention.

2.1 Network topology

The wiring in a single dimension is interleaved as is illustrated below in the Figure 1. Because of the interleaving, an edge node is not further from a neighbor on the opposite edge than is an interior node from its neighbors.

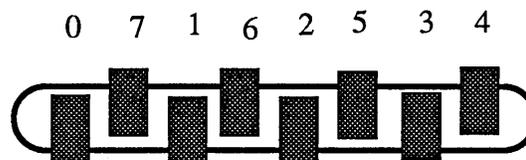


FIGURE 1. Node interleaving in 1 dimension. Logically adjacent nodes are separated.

Due to the periodic nature of the 3D grid, all positions are logically equivalent and each node has essentially an equivalent position in the cube. Because of the connections along edges, message traffic along an axis in the 3D cube can travel in either direction. This features alleviates most network contention. In general, programmers do not have to be aware of the physical position of a logical PE.

The T3D interconnection network uses a “dimension order routing” for propagating messages. When messages are traversing the network, there is a bias for first traveling in an X direction, followed by a Y direction and lastly a Z direction. The effect of this bias is very small. The graph in Figure 2 shows the magnitude of this bias. In this graph, we have plotted the measured message passing latency relative to a logical node 0 for each PE in a CRAY T3D with 256 nodes.

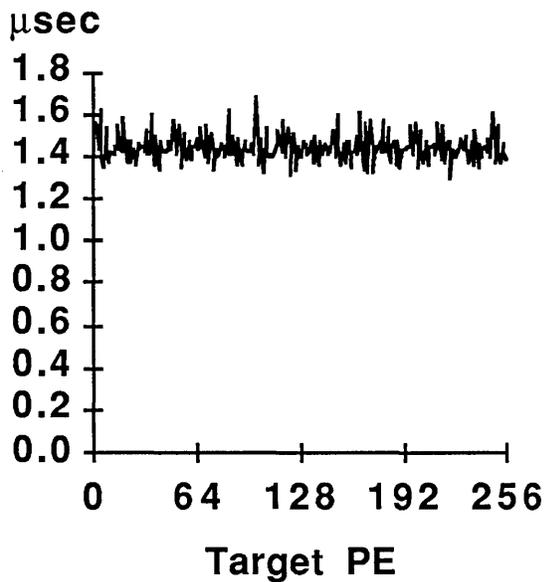


FIGURE 2. Measured inter-node latency as a function of node number.

In the graph, we see the slight effect of dimension order routing. The plot has a major period of 16, which is the size of the x dimension of the system on which the test program was run.

2.2 Hardware implementation

The inter-PE communication network is implemented with technology developed for traditional Cray Research supercomputers. The T3D system’s interconnect uses C90 style 10K gate arrays which are mounted on Y-MP style modules. The modules in the T3D have two boards, whereas the Y-MP and C90 modules have four boards. The T3D system has a clock rate of 150 MHz. This technology leads to the low latencies for communication between nodes.

2.3 Partitions

The details of the network configuration are generally transparent to the user. The node configuration of each of the T3D various marketed systems are within a factor of two of being perfect cubes. That is, no dimension of a T3D system has more than twice as many nodes as does any other dimension. In actual use, hardware partitions can have any shape within the system. “Unusual” hardware partitions can have some effect on communication patterns.

Software partitions can also have some effect of communication patterns, but generally less. Software partitions allow communications to pass through other partitions, so boundaries are so not “hard”.

2.4 Network performance

Each Processing Element (PE) at a node has its own memory, but the pair of PEs share a single network switch. Each node in the T3D has two

Processing Elements (PEs). These PEs, which are DEC Alpha microprocessors, will be described later.

2.4.1 Synchronization

The T3D network has hardware synchronization primitives. The most obvious effect of this hardware is fast synchronization or barriers. The hardware synchronization time is lightly related to the system size. For a 64 node (128 PE) system, the synchronization time is less than 1 microsecond. The lower curve in the figure below shows the synchronization time versus the number of PEs for a primitive user programmed barrier. The upper curve is from measurements using the BARRIER() function in libc.

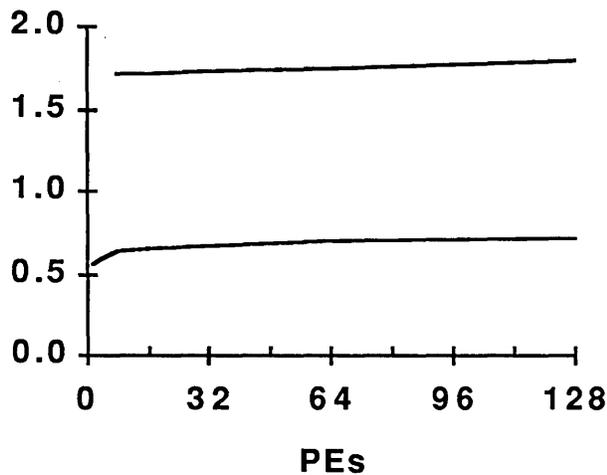


FIGURE 3. Synchronization time as a function of number of PEs. The lower curve is for synchronization without deadlock detection. The upper curve is for synchronization with deadlock detection.

2.4.2 Single channel performance

The interconnect channels are 16 bits (two bytes) wide and are bidirectional. That is, a message can be sent and received at full speed on a single network channel. The bandwidth of a channel is 2 bytes times 150 MHz, or 300 Mbyte/s. A packet of data on the network is 8 bytes long. Of this eight bytes, four bytes are

data, two bytes are header information and two bytes are trailers. The load factor is 50%, so the bandwidth available to the user is 150 Mbyte/s in a single directions.

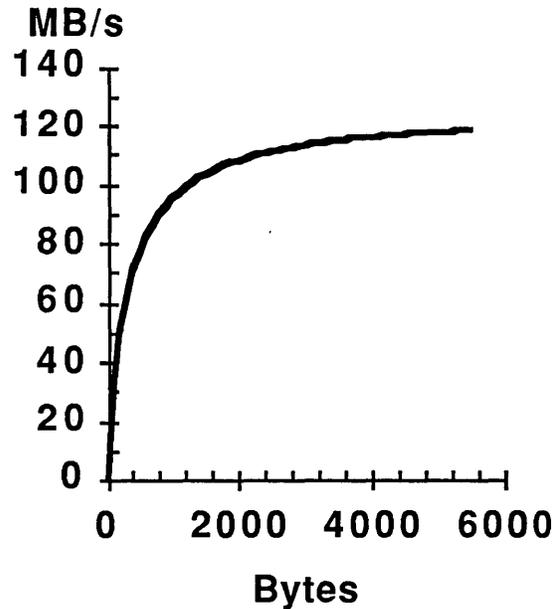


FIGURE 4. Internode communication rate as a function of message size.

For node to node message passing, the minimum measured latency is 1.3 microseconds and the measured asymptotic bandwidth is 125 Mbyte/s. The message size at which the bandwidth is one half of maximum, or $N_{1/2}$, is approximately 200, or 25 words. A typical plot of bandwidth versus message size is shown in Figure 2. These figures are for small messages which can be contained in the data cache. That is, for messages smaller than 16 Kbytes. For large messages, the asymptotic bandwidth is the same, but the latency is approximately 1.7 microseconds.

The shape of the curve in Figure 3 is reminiscent of vector speed curves. In fact, the parameterization of this curve is from Prof. Roger Hockney, one of the developers of the

parameterization using R_{∞} and $N_{1/2}$ to characterize vector processing.

TABLE 1. Message Passing Rate using PUTs. "Small if for cache contained messages. "Large" is for messages larger than cache (16 KB)). Message Passing Rate using GETs.

R_{∞}	$N_{1/2}$	T_{start}	Size
125 MB/s	164 Bytes	1.3 μ sec.	Small
127	380	3.0	Large

TABLE 2. Message Passing Rate using GETs. "Small if for cache contained messages. "Large" is for messages larger than cache (16 KB).

R_{∞}	$N_{1/2}$	T_{start}	
35 MB/s	180 Bytes	5.2 μ sec	Small
37	144	3.9	Large

2.4.3 Multiple channels

For global communication, we have attained an average node-to-node bandwidth of over 120 Mbyte/s. This bandwidth is close to the rated bandwidth of the network interconnect. Note that because there are two PEs per node, the average rate per PE is over 60 Mbyte/s.

Examples of demanding global communication algorithms are transpose and a global collection. Bandwidth results for these two algorithms are shown in Table 3. These tests demonstrate both a high sustained bandwidth and the lack of contention in the network.

TABLE 3. Global bandwidth examples.

Algorithm	Bandwidth per PE
Transpose	60 MB/s
Global collection	63 MB/s

III. Micro Architecture

3.1 DEC Alpha

The T3D uses the EV2064 microprocessor from DEC. This microprocessor is more commonly known as the "Alpha" microprocessor. The Alpha microprocessor is used in DEC's line of AXP workstations which run at clock speeds of from 140 MHz to 200 MHz. The microprocessor version which is used in the T3D runs at 150 MHz, the same speed as that of the network interconnect.

The Alpha microprocessor used in the T3D is not binary compatible with the AXP computers manufactured by DEC. The version of the Alpha microprocessor used in the T3D has several different instructions from what are used in the AXP products. The different instructions allow, among other features, the implementation of shared memory. The shared memory features will allow more efficient parallel processing and easier program development. The compromise in the selection of the modifications to the Alpha instructions is that binaries generated for AXP workstations will not directly run on the T3D.

Another significant difference between the two versions of the Alpha microprocessor is the absence of a secondary memory in the T3D's PEs. Among the reasons for not implementing a secondary cache are that there would be additional coherence considerations for parallel processing.

We have not been able to run tests which accurately evaluate the single PE performance impact of not having a secondary cache. Many quoted benchmarks for the AXP products benefit from the secondary cache, but we do not know if "typical" MPP applications would likewise benefit from this cache.

Measured, or sustained, speeds for individual PEs range up to 105 Mflop/s for matrix multiply kernels. More typical Fortran code runs from 10

Mflop/s to 40 or 50 Mflop/s. The Alpha microprocessor can issue one floating point add or one multiply every clock period, so the “burst” or “theoretical peak” speed is 150 Mflop/s.

Table 2 shows single PE speeds for the NAS Parallel benchmarks. The listed speeds are extracted from data for 32 PEs. The median speed is 16.6 Mflop/s. Note that the speed for EP (Embarrassing Parallel) is artificially high because the implemented algorithm uses fewer operations than the official operation count. The floating point speed for IS (Integer search) is low because the program performs few floating point operations..

TABLE 4. Single PE performance for the NAS Parallel Benchmarks. Results extracted from 32 PE runs

Program	Speed
EP	83.0 Mflop/s
MG	20
CG	4.2
FT	26.9
IS	3.6
LU	10.5
SP	16.4
BT	21.2

Each PE in the T3D has an 8 Kbyte (1 Kword) data cache and either 16 Mbytes (2 Mwords) or 64 Mbytes (8 Mwords) of local memory. The Alpha microprocessor has a separate 8 Kbyte instruction cache. The size of the local memory is great concern to the programmer. The microkernel running on each PE requires approximately 3 or 4 Mbyte of memory. On the 16 Mbyte models, this leaves 1.3 Mwords for user space. 64 Mbyte models have over 7.5 Mwords of user space.

3.2 Cache organization

The size and organization of the data cache is of significance to the programmer. The contents of

and the coherence of the data cache can be manipulated by the programmer. For some shared memory message passing, the coherence of the data cache must be explicitly controlled by the programmer.

Programmers striving for the highest possible performance should be aware of the 1 Kword size of the data cache. Blocked algorithms should be manipulated to work on blocks of this size. The data cache size is also of significance for algorithms which use look-up tables. Because look-up tables often involve gather operations, tables which fit in cache are significantly faster than those which must gather data from local memory.

The Alpha microprocessor data cache is direct mapped. This means that each memory location is mapped onto one unique cache location. The ramification of this feature to the programmer is that concurrent references to data which have memory addresses differing by multiples of 1024 words should be avoided. These kinds of concurrent references can usually be avoided by padding arrays.

An simple example of array padding is shown below in Figure 4. In this example, if the padding were not present, the reference to A(i) and B(i) would be to the same data cache locations. The padding size was chosen so that one variable reference could get several cache lines ahead of the other variable.

```
REAL*8 A(1024*10, B(1024*10)
COMMON /program/ A, pad(32), B

DO i=1,n
A(i) =...B(i).
END DO
```

FIGURE 5. Example of padding of arrays to avoid cache conflicts.

The bandwidth for various operations is shown in Table 5. The two results given for the an array copy: one for using the hardware read-ahead feature and one without this feature. We see that

the read-ahead feature enhances the bandwidth by over 25%. The result for the library SAXPY operation shows that the peak attainable memory bandwidth is 350 Mbyte/s.

TABLE 5. Node Bandwidth. Rates are in MB/s

Operation	Rate	Comment
Copy	193 MB/s	Fortran w/o read-ahead
Copy	258	Fortran w/ read-ahead
SAXPY	348	Libsci

IV. Conclusion

The salient feature of the T3D macro architecture is the low latency high bandwidth torus. Because of the torus' connectivity, the programmer generally does not have to be concerned with the relative location of PEs or the location of a partition within a system.

A programmer working with the T3D should be aware of several micro architecture features. In particular, the organization of the cache leads to certain conflicts which can be remedied by adjusting relative addresses.

Also of significance to the programmer is the memory bandwidth of the PEs. This bandwidth will eventually limit the performance of well written programs.

User Services/Software Tools

CONFESSIONS OF A CONSULTANT
(A compendium of miscellaneous tools/scripts/hints/tips/techniques)

Tom Parker

**National Center for Atmospheric Research
Boulder, Colorado**

Abstract

Consultants require more tools than the average user, so that they can access a wide variety of information to assist users. This talk discusses various tools that the NCAR consultants have informally developed/acquired/borrowed. These miscellaneous tools help us locate system resources, resolve user problems, examine files, debug Fortran, and provide pointers to other sources of information, both on and off the Cray. We also have written several "reference cards" for popular, local software, to improve ease-of-use.

Introduction

This presentation consists of a compendium of varied and miscellaneous tools, tips and techniques. These were written or acquired over the years by the author, a user consultant. They have proved useful in assisting users or in managing Unix files and directories.

Due to the nature of the subject matter, this document consists mainly of lists and pointers, rather than a narrative. 50 tools are summarized in Appendix 1. Several tips about vi, cft77, and reference cards are in Appendix 2. About 70 miscellaneous tips and techniques are listed in Appendix 3. Many good sources of information are described in Appendix 4.

Thus, this paper/talk presents a "grab bag" of miscellaneous tools and tips. The author hopes that the reader will find something useful for their needs.

All of the non-proprietary items mentioned here, are also available via anonymous ftp from ftp.ucar.edu in directory /cug/5c.

Appendix 1 - Summary of Tools

<u>Command</u>	<u>Category</u>	<u>Lang</u>	<u>Man</u>	<u>Avail+</u>	<u>Function</u>
allsys	misc	csH			Run a command on several systems.
beav	files	c	y	archie	Binary editor and viewer.
bed	files	c	y	archie	Binary editor.
beep	misc	csH			Beep your terminal.
calc	misc	perl			Calculator.
ccc	fortran	csH			Remove comment/empty fortran lines.
cfint	fortran	c	y	CRI*	Analyze fortran source.
cmdcount	commands	csH			Count system command usage.
cosconvert	files	cf77	y		Convert COS-blocked dataset.
cosfile	files	cf77	y		Analyze a COS-blocked dataset.
cossplit	files	cf77	y		Split multifile COS-blocked dataset.
dir	directory	alias			Find directory and cd to it.
dirsize	directory	csH			Calculate size of directory (bytes).
each	files	csH			Run a command against a list of files.
exist	files	csH			Beep when a file exists.
ff	files	csH			Run findfile and finddir.
fi	files	csH			Find a file and vi it.
files	files	csH			Display files or directories.
filetime	files	c			Display three timestamps of a file.
findcmd	commands	csH			Find all about a command.
finddir	directory	csH			Find a directory.
findfile	files	csH			Find a file.
findlib	libraries	csH			Find a library.
findman	commands	csH	y		Find a man page.
findme	misc	csH			Find info about me.
findstring	files	csH			Find a string in files.
findsub	libraries	csH	y		Find a subroutine in a library.
flint	fortran	c	y	IPT*	Analyze fortran source.
gzip	files	c	y	archie*	GNU File compressor.
large	directory	csH			List files by size.
lc	files	csH	y		Convert file names to lower case.
libx	libraries	csH			List names of system libraries.
linemax	files	csH			Find largest line.
loc	files	csH			Search for strings.
lrecl	files	c	y		Show min/max record lengths.
makev	files	csH			Truncate trailing blanks.
most	files	c	y	archie	Hex/Ascii viewer.
olddate	files	csH			Change date of a file.

<u>Command</u>	<u>Category</u>	<u>Lang</u>	<u>Man</u>	<u>Avail+</u>	<u>Function</u>
pcprint	files	csH	y		Print a file on PCattached printer.
realpath	misc	csH			Display full pathname of a file.
rnhome	rn	csH			Run rn on extra-curricular newsgroups.
rot13	files	csH			Simplest of encryption techniques.
ruler	misc	csH			Display a ruler (1-80).
run	fortran	csH			Compile, link, run.
textfile	files	cf77	y		Display info about a text file.
trail	files	csH			Display lines with trailing blanks.
tree	directory	c			Visual display of tree structure.
xdump.perl	files	perl			Perl script to dump a file in hex.
xv	files	c	y	archie*	Great image viewer that can hex dump.

+ All items are available from anonymous ftp site "ftp.ucar.edu" in directory "/cug/5c", except those marked with * in the "Avail" column.

Appendix 2 - Tips and Techniques: vi, reference cards, cft77

There is a lot of good information about the 'vi' editor, in

<ftp.uu.net:/pub/text-processing/vi/>

You can customize your 'vi' session via settings in the vi initialization file, normally called ".exrc". Below are some .exrc settings I have found useful. For example, to lowercase the line I'm on in vi, I can type ";l".

```
" Some settings I like.
set ic number showmode
"
";f and ;g = Paragraph formatters.
:map ;f !}fmt -81
:map ;g !}fmt -61
"
";u = Uppercase a line.
:map ;u :s/.*^U&/
"
";l = Lowercase a line.
"
:map ;l :s/.*^L&/
"
```

```

";z = Insert my signature file
:map ;z :r ~/.sig-elm
"
";> = Insert '>'s til EOF.
:map ;> :.,$s/^/> /
"
";a = switch between 2 files
:map ;a :e#
"
";r = Insert long ruler (80 columns)
:map ;r 0O.....1.....2.....3.....4 -- .6.....7.....8
"
";s = Insert short ruler (72 columns)
:map ;s 0O.....1.....2.....3.....4 -- .6.....7..
"
" Allow arrow keys to work in VI "text insert" mode
:map! OA ka
:map! OB ja
:map! OC la
:map! OD ha

```

Reference cards are a handy source of information. Some good UNICOS cards include:

- "Shells SQ-2116"
- "User Commands SQ-2056"
- "CF77 SQ-3070"

You can also write your own reference cards, for popular local software. For example, at NCAR we have cards for several popular systems. (Free copies available upon request to the author).

cft77 Debugging Tips

Here are a few tips I've found useful in debugging Fortran problems.

- o First and foremost: flint (from vendor IPT) or cflint (from CRI)
- o Check subscripts and arguments: cft77 -Rabc
- o Uninitialization problems: cft77 -ei

- o SAVE problems: `cft77 -ev`
- o Optimization problems: `cft77 -o off`
- o Clobbering code problems: `segldr -n`
- o Memory problems:

`PRINT *, 'AVAIL. memory (in words) = ', IHPSTAT(12)`

- o `'assign -V'`
- o `'cdbx'`
- o `'explain'`
 - `explain sys-2`
 - `explain lib-1051`
- o Four common run-time errors:

1) Operand Range Error

An ORE occurs when a program attempts to load or store in an area of memory that is not part of the user's area. This usually occurs when an array is referenced with an out-of-bounds subscript.

2) Program Range Error

A PRE occurs when a program attempts to jump into an area of memory that is not part of the user's area. This may occur, for example, when a subroutine in the program mistakenly overwrites the B/T save area. When this happens, the address of the routine from which the subroutine was called is lost. When the subroutine attempts to return to the calling routine, it jumps elsewhere instead.

3) Error Exit

An error exit occurs when a program attempts to execute an instruction parcel of all zeroes. This error usually occurs when the program's code area has been mistakenly overwritten with words of data (for example, when the program stores in an array with an out-of-bounds subscript).

4) Floating-point Exception

An FPE occurs when a program attempts to perform a floating-point division of 0. It may also occur with floating-point operations in which one of the operands is not a valid floating-point value. An invalid floating-point value may be created when a real array has been equivalenced to an integer

array, and integer values are stored in the array and then later referenced as real.

Appendix 3 - Tips and Techniques: Miscellaneous

This appendix list about 70 miscellaneous tips and techniques.

Neat way to check permissions:

```
find $HOME -perm -002 -print # Shows files that have OTHER write permission
```

Grab last 30 characters in a string: `echo $string | rev | cut -c1-30 | rev`

Hex stuff: `echo "ibase=16;obase=16rest of function"|bc`
BTW, you can use variables in the echo above.

Remove non-printable characters from a file (except NewLine and Tab).
`/usr/bin/tr -dc '\011\012 -~' < $1`

Insert spaces between every character in a string:
`echo "string" | sed -e "s/^(.)\1 /g"`

Demo of basename and dirname
`echo 'basename /a/b/c' ---> c`
`echo 'basename 12345 45' ---> 123`
`echo 'basename myprog.c .c' ---> myprog`
`echo 'dirname a/b/c/d.e' ---> a/b/c`

Use 'dd' for: ASCII<-->EBCDIC, Upper<-->Lower, Swap VAX bytes

Uppercase a file: `tr '[a-z]' '[A-Z]' < input > output`

Count number of blanks in a file: `tr -cd ' ' < input | wc -c`

Change multiple blank lines to a single blank line -- several methods:

```
cat -s      fn
sed '$!N;/\n$/D;P;D' fn
sed '/./,/^$/!d' fn
```

To list just 'dot' files: `ls -d .*`
To list just directory files: `echo */` or: `/bin/ls -ld */`
or: `/bin/ls -ld */ .*/`

To list size of directories: `du -s */` or: `my 'dirsize'`

```

# To rename all .old files to .new, type this in csh:
  foreach x(*.old)
    mv $x $x:r.new
  end

# Find all lines that end in blank(s): grep "$ myfile
  To remove trailing blanks while in vi: :%s/ *$// (or use my 'makev' command)

# Find a string in a directory:
  find . -type f -exec grep 'string' {} /dev/null ';'
  Or, use my 'findstring' command.

# Neat commands: colrm, col, colcrt, fmt, fold, hc, join, nl, paste, pr
  xmessage

# Handy datacomm commands: traceroute, host, whois, nslookup

# See if site is on Internet: whois

# Some ways to double space a file: sed G file; pr -dt file

# Print first 50 lines of a file: head -50 file; sed '50q' file

# Print lines 30 to 35: sed -n '30,35p;35q' file

# To reverse the order of lines in a file: tail -r <file>
  Or: perl -e 'print reverse <'
  Or, in vi: :g/^/m0

# To find a G in columns 1-5:
  grep '^.\{0,4\}G' ~/cds/master

  Or more generally, to find string xxx in columns m-n:
  grep '^.\{m-1,n-1\}xxx' file-to-search

# Change null bytes to blanks: perl -pi -e tr'^\000/' <file>

# To extract the first 5 letters from a variable using awk:
  echo $1 | awk '{print substr($1,1,5)}'

  similarly for the last three:
  echo $1 | awk '{print substr($1,length($1)-2,3)}'

# awk 'length >= n' file # Print lines >= n characters.

# To print first token of each line: awk '{print $1}' file
  ('cut' works too: cut -d' ' -f1 <file)

```

Print first two fields in opposite order: `awk '{ print $2, $1 }' file`

Print line number and number of tokens:

```
awk '{print "Line number="NR,"Tokens="NF}' file
```

Add up sizes of .zip files:

```
find . -name '*.zip' -ls | awk '{s += $2} END {print s}'
```

```
ls -al *.zip |
```

```
awk '{i = i + $4;j++;} END {print j " files use " i " bytes"}'
```

Print every 10th line of a file: `awk '(NR %10 == 0) {print}' file`

To pass a shell variable into awk, you need to put it on the command line:

```
awk <awk-program> <variable-assignments> <filename>.
```

E.g. to prepend every line of /etc/hosts with the value of the variable A;
assume for this run that A's value should be "Sam"

```
awk '{printf("%s: %s", A, $0)}' A=Sam /etc/hosts
```

If you wish to assign multiple variables, you can do them one at a time before the filename. Note that if the input should come from stdin, you must replace the file name with "-".

Performance commands: `vmstat`; `pstat -s`; `top`

To capture a telnet session: `telnet wherever | tee my.session`
(Doesn't seem to work though, for 'ftp').

Change all occurrences of a string in a file:

```
sed 's/old.string/new.string/g' filename > file.out
```

To bypass an alias, eg "alias rm 'rm -'", use: `\rm xxx`

Also, in a script with "`#!/bin/csh -f`", the `-f` will bypass all aliases, etc.

To test status of a command, in csh:

```
if { cat a } echo "Command worked"
```

```
if ! { cat nothere } echo "Command failed"
```

To get a random number: `ksh 'echo $RANDOM'`

```
... or: set test1 = 'perl -e 'srand; print int(rand(200)+1)''
```

```
or: set test1 = 'perl -e 'print int(rand(100))''
```

To see if file1 is newer than file2:

```
if('find file1 -newer file2 -print' == "") echo "file2 newer"
```

```
-or-
```



```
a:h = /a/b
a:t =   c.d
a:r = /a/b/c
a:e =   d
```

```
# Quoting variables, in csh -- :q and :x
If variable is a wordlist, use :q to quote it & preserve wordlist.
If variable is NOT a wordlist, use :x to quote it & create wordlist.
```

```
# LIMITATIONS in csh (from man page)
Words can be no longer than 1024 characters. The system
limits argument lists to 1,048,576 characters. However, the
maximum number of arguments to a command for which filename
expansion applies is 1706. Command substitutions may expand
to no more characters than are allowed in the argument list.
```

```
# Global substitution, in csh:
```

```
> echo hi hi hi
hi hi hi

> !:gs/hi/hello
hello hello hello
```

```
# Emoticons are things like: :-), ;-) For more, check:
```

- (1) mercury.unt.edu /pub/misc/EMOTICON.TXT,
- (2) rascal.ics.utexas.edu /misc/misc/EMOTICON.PS,
- (3) ugle.unit.no /pub/misc/jargon/expanded/emoticon, or
- (4) wiretap.spies.com /Library/Humor/Nerd/.cap/emoticon.txt

```
# If VT100 screen fonts are hosed, try: tput rmso; tput reset
```

```
# To see a file in FTP: get file.of.interest "|more"
or get file.of.interest -
To view a big ls: ls . "|more" # Need the ".".
or ls -lt "|more"
or dir . "|more" # Need the ".".
```

```
To capture a big directory: ls -lt xxx # Puts it in file xxx back home.
```

```
# To 'tar' entire directory (including . files): "tar -cvf my.tar ."
To 'tar' entire directory (excluding . files): "tar -cvf my.tar *"
```

```
# To make global changes: perl -pi.bak -e 's/old/new/g' *
```

```
# Change all null bytes to blanks: perl -pi.bak -e 'tr/\000/ /' tezz
```

```
# Erase words with less than n letters (the words are one per line):
```

```

perl -ne 'print if (length() >= n)' <infile >outfile

# List last mod time of file >6 months old:
perl -e 'require "ctime.pl"; print &ctime((stat("filename"))[9])'

# Print current time in seconds from epoch: perl -e 'print time,"\\n"'

# Print 3 times associated with a file, in seconds since epoch:
perl -e 'printf "%d %d %d\\n", (stat(shift))[8..10]' .cshrc

# To remove trailing blanks while in vi: :%s/ *$//

# To spell check a file in vi: :!spell %
To test a few words: spell; words words ; words words ; EOT

# To expand tabs, in vi: :%!expand

# Delete columns 10-20, n vi: :1,$!cut -c 1-9,21-
on some systems: :1,$!colrm 10 20 ! Not shavano!

# From the book: Learning the vi Editory (Nutshell book)
- To replace current line: cc or S
- To replace from cursor to EOL: c
- To show all lines containing string: :g/string/p
- To show all lines NOT containing string: :g!/string/p
- To show TABs and EOLs: :set list
- To move to top of screen: H
- To move to bottom of screen: L
- To move to 1st char of next line: <return>
- To display file info: ^g
- To format a paragraph: !}fmt
- To specify all lines: :1,$ or :%
- Global change: :%s/old/new/g
- Selective change: :%s/old/new/gc
- Do :q! and then vi same file again: :e!
- Delete all blank lines: :g/^$/d
- Delete all blank OR white space lines: :g/^[ TAB]*$/d
- Reverse order of lines: :g /./mo0
- Read in results of command (e.g. date): :r !date
- Sort lines 10-20: :10,20 !sort

```

Appendix 4 - Information Sources

o Tools mentioned in this talk are available via anonymous ftp to:

ftp.ucar.edu:/cug/5c/

o rn (Reads Usenet newsgroups)

- comp.unix.questions comp.unix.shell
comp.sources.misc comp.sources.unix

o Internet: archie, gopher, www

- Federal Income Tax forms
- Perl Archive

o "UNIX Power Tools"

"UNIX Power Tools", by Peek, O'Reilly, & Loukides.
O'Reilly & Associates/Bantam Books
March, 1993

o Unix Books

anonymous ftp to rtfm.mit.edu:/pub/usenet/misc.books.technical and get file
"[misc.books.technical]__A_Concise_Guide_to_UNIX_Books".

o FAQs

To get FAQs, anonymous ftp to rtfm.mit.edu:/pub/usenet/.
There are directories for many usenet groups; look in the desired directory
for an FAQ.

E.g. in the directory "/pub/usenet/comp.unix.questions" you will see several
files that start with "Unix_-_Frequently_Asked_Questions"

Also, the directory /pub/usenet/news.answers/ contains many FAQs.

o IPT - vendor for Fortran-lint (flint)

Information Processing Techniques
1096 East Meadow Circle
Palo Alto, CA 94303
(415) 494-7500

o www - World Wide Web

- Frequently accessed via 'xmosaic', 'lynx', or 'gopher'.
- Federal Income Tax Forms: <http://www.scubed.com:8001/tax/fed/>
- Perl archive: <http://www.cis.ufl.edu:80/perl/>

o Author: Tom Parker, tparker@ncar.ucar.edu, (303) 497-1227

SHORT PAPERS

Xnewu: A Client-Server Based Application for Managing Cray User Accounts

Khalid Warraich and Victor Hazlewood

Texas A&M University
Supercomputer Center
College Station, TX 77843-3363

victor@tamymp.tamu.edu

ABSTRACT

Many tools are available on the UNICOS system for account administration and system management. However, none provide a graphical interface to account creation and/or management. A client-server based application was developed at Texas A&M to add new login ids and new accounts. The graphical interface provides point and click functionality to adding new UNICOS accounts. The information is validated and then sent to the server which resides on UNICOS. This server provides some additional data validation and then makes the appropriate calls to log the transaction, calls *udbgen* and other utilities to set up the account.

1. Background

The Texas A&M University Supercomputer Center (TAMUSC) was established in August 1989 to provide supercomputer resources to Texas A&M University faculty, students, and researchers. TAMUSC decided from the onset of the center to use the Cray System Accounting (CSA) package that is provided by UNICOS. A local database containing local accounting and user information was included as a supplement to CSA and is updated at the creation of each user login and user account (users may have multiple accounts per login name). This information is not stored in the User Data Base (UDB), therefore, it is not managed by the UNICOS utility *nu*. *nu* provides some flexibility but does not supply an X Window (X) graphical user interface nor the ability to add local fields and commands to the configuration file */etc/nu.cf60*. Over the last year three analysts at Texas A&M have created an X based client-server application called Xnewu which manages the creation of new logins and accounts for the Cray system at TAMUSC.

2. Xnewu objectives

The primary objectives for the TAMUSC Xnewu utility include the following:

- 1) provide client-server capability to manage accounts on the VAX and the Cray systems,
- 2) provide an easy to use X Window user interface on TAMUSC's SUN computer system,
- 3) update local accounting and user information databases on the Cray,
- 4) log each account creation transaction for disaster recovery.

When a user receives his first TAMUSC account, a login name and account are created on a local VAX system and on the Cray. The TAMUSC requires that the login name be the same on both systems for accounting and billing purposes. This requirement forces a coordination between the two systems. Once all the appropriate information is given for a new account in Xnewu (see Figure 1) a request is made for a new username to an authoritative database. When the username is determined, a request to create a new login is sent to the VAX and Cray.

To create subsequent accounts for a user, his existing username is determined, the appropriate information entered in Xnewu, and then an account creation request is issued. This account creation request is sent to the Cray only.

3. Xnewu Overview

We chose our SUN system as the point of user interaction with the Xnewu system for account creation on the VAX and the Cray systems over the network. The SUN system provides the graphical capability, as well as the networking ability needed to manage the accounts. The Xnewu program on the SUN is written using the Open Look Intrinsic Toolkit and acts as the client to the Cray servers. The interaction with VAX is done through an *expect* script running on the SUN.

The creation of a TAMUSC account starts with the approval of a researchers grant paperwork. Once approved the user information is enter into the Xnewu client on the SUN computer system. If this is the first account for a user a new login name is determined by requesting a new login name from the IBM authoritative database via the VAX through the *expect* script (see Figure 2). After the name is returned a request to create a new login name is sent to the VAX (using the *expect* script) and the Cray Xnewu servers (using a TCP connection, see Figure 2).

When the Cray server receives an account or login creation

request the server logs the transaction in a file for disaster recovery. Then it updates local accounting and information files with information regarding the user's identity, his department, college, grant type received, and the grant time. Next the server runs the *udbgen* program to update the necessary fields in the UDB. Finally, the server runs a script to create the users home directory (if necessary), the .forward file, an entry in the USCP slot file, and copies the "dot" files to the users home directory. Miscellaneous other tasks could simply be added to this script if necessary.

5. Implementation

(The focus of this section will be on the SUN and Cray portions of the Xnewu client-server application.)

The three main routines representing functions of Xnewu client on the SUN (implemented as callback routines and activated by appropriate buttons on Xnewu) are *check_callback*, *echo_callback*, and *create_callback*.

The *check_callback* routine checks all input data and checks its validity and consistency. If any required data is missing, out of range, or inconsistent with other data, the *check_callback* routine will warn the user and describe the problem in the status line at the bottom of the Xnewu form.

The *echo_callback* procedure first calls *check_callback* and then displays the current input data on standard output. This allows the user to inspect the data before it is transmitted to the VAX and/or the Cray.

The action of *create_callback* routine depends on the "Action" item selected in the Xnewu form. The two supported actions are "Create Login" and "Create Accounting Id."

If *Create Login* is selected at the time the *Perform Action* button is pressed (initiating the *create_callback* routine) then a login id is obtained from the IBM via the VAX, the VAX account created, and a request structure is built (see Table 1 below) and forwarded to the Xnewu server on the Cray for processing.

Table 1: Request Structure

Login id	Users login id
Account	Users accounting id
Password	Users password
Name	Users First, Middle, and Last Name
SSN	Users social security number
Dept	Users department name
College	Users college name
Email	Users default Email address

Table 1: Request Structure

PIname	Users Supervisor (Principal Investigator)
PIacid	PI's controlling accounting id
OrigSBU	Users Cray resource allocation
HomeDir	Users Home Directory
GrantType	Type of grant user is being allocated

If *Create Accounting ID* is selected at the time the *Perform Action* button is pressed a request structure is built and sent to the Cray for creation of a new account for an existing user.

Once the Xnewu server on the Cray receives either the *Create Login* or the *Create Accounting ID* request several pieces of the information are checked for accuracy and consistency (e.g., duplicate login ids, account creation for non-existent login. etc.). This second check on the Cray is more extensive than the one on the SUN since more information is available on the Cray than on the SUN. An entry is then made to a log file for disaster recovery. Next, a *udbgen* command is created and executed with the request structure information to created either the new login or the new account. The local TAMUSC information database is updated. Then the script to create user's home directory (for new login requests) and miscellaneous other tasks is executed. Finally, the server goes to sleep waiting for the next request.

6. Conclusion

The Xnewu client-server application provides an easy to use, easily modifiable interface to creating new logins and accounts for the Cray system at TAMU. This application could be easily modified to provide other sites with similar account administration needs a flexible account creation interface. The Xnewu application provides logging capabilities, update of local databases, update of the UNICOS UDB, and execution of a local script which can perform any number of miscellaneous tasks.

The Xnewu client-server application requires a SUN system with TCP/IP network access to the Cray system.

QEXEC: A TOOL FOR SUBMITTING A COMMAND TO NQS

Glenn Randers-Pehrson

U. S. Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

INTRODUCTION:

"qexec" is a shell script that can simplify the procedure for running a command in batch mode through the UNICOS Network Queueing System (NQS) (UNICOS is a trademark of Cray Research, Inc).

Using the existing NQS "qsub" procedure, you are normally required to write a shell script containing the command to be executed. For example, suppose you wish to run the command

```
cft77 -em x.f
```

in batch mode using 2000kW memory and 100 second limits. You would put "cft77 -em x.f" in a file called "cft77.sh" and then type:

```
qsub -eo -lm 2000kW -lM 2000kW \  
-lt 100 -lT 100 cft77.sh
```

Using "qexec" you would not have to bother with creating "cft77.sh"; you would simply type

```
qexec -eo -m 2000 -t 100 cft77 -em x.f
```

"qexec" passes its options to "qsub". In addition, it understands "-t sec" as shorthand for "-lt sec -lT sec" and "-m 1000" as shorthand for "-lm 1000kW -lM 1000kW".

THE .qexecrc FILE:

You may put default options for "qexec" in a `./qexecrc` or `$HOME/qexecrc` file. Options on the "qexec" directive take precedence over those mentioned in these files and options in `./qexecrc` take precedence over those in `$HOME/qexecrc`. For example, suppose `./qexecrc` does not exist and `$HOME/qexecrc` contains:

```
-eo -m 2000 -t 600
```

Then you would obtain the same effect as above by typing the following:

```
qexec -t 100 cft77 -em x.f
```

This will take "-eo" and the memory limit from `$HOME/.qexecrc`, and the time limit from the "qexec" command.

If for some reason you wish to submit a job with "qexec" entirely ignoring your `.qexecrc` and `$HOME/.qexecrc` files, then put a "-" at the beginning of the option list, e.g.

```
qexec - -m 2000 -t 100 cft77 -em x.f
```

CHANGING TO THE PRESENT WORKING DIRECTORY:

Like "qsub", "qexec" will run the job in your home directory if you have not included an appropriate "cd" command in your `.profile` or `.login` file. The following will cause your job to run in the directory from which "qexec" or "qsub" was issued:

"sh" users should include in `.profile`:

```
case ${ENVIRONMENT:=LOGIN} in  
  BATCH) cd $QSUB_WORKDIR;;  
  esac
```

"csh" users should include in `.login`:

```
if ($ENVIRONMENT == BATCH) then  
  cd $QSUB_WORKDIR  
endif
```

THE qexec.log FILE:

"qexec" reports its activity in a `./qexec.log` file, giving the time that the job was submitted to NQS, the qsub options, a copy of the command, and the time that the job actually began and finished execution. This information is also included in the standard output of the job, along with timing results obtained with `/bin/time`.

NQS JOB NAME AND PROCESS NAMES:

"qexec" makes a local copy of `/bin/time`, named `$QSUB_REQID`. This can help you identify your job in the "ps" or "top" displays; the process leader will have the same name as the NQS request ID. The first word of the command (e.g. "cft77") becomes the NQS job name.

THE qexec SCRIPT:

The "qexec" shell script follows. You may obtain a machine-readable copy by sending e-mail to <glennrp@arl.army.mil>.

```
#!/bin/sh
# qexec [-] [-m kw] [-t sec]
# [qsub_args] [--] cmd [cmd_args]

# creates a shell script containing a
# command and its arguments and submits
# the script for execution via qsub

# if no leading "-", uses any arguments
# in .qexecrc or $HOME/.qexecrc that are
# not specified on the command line

# makes an entry in ./qexec.log

# written by Glenn Randers-Pehrson
# U. S. Army Research Laboratory
# Aberdeen Proving Ground, MD
# <glennrp@arl.army.mil>

case $1 in
- ) qexec_rc="";;
*) qexec_rc=`cat -s .qexecrc $HOME/.qexecrc`;;
esac

argname=""; silent=no; qsub_args=""
have_lm=no; have_lM=no; have_r=no
have_lt=no; have_lT=no

for arg do
case $1 in
-a) # wait until after specified time
# change embedded blanks to commas
when=`echo $2|sed -e s/" , *"/,/g`
qsub_args="$qsub_args -a $when"
shift; shift;;
-lm) # memory limit
qsub_args="$qsub_args $1 $2"
have_lm=yes; shift; shift;;
-lM) # memory limit
qsub_args="$qsub_args $1 $2"
have_lM=yes; shift; shift;;
-lt) # time limit
qsub_args="$qsub_args $1 $2"
have_lt=yes; shift; shift;;
-lT) # time limit
qsub_args="$qsub_args $1 $2"
have_lT=yes; shift; shift;;
-m) # memory limit
case $2 in
*[0-9]) units=Kw;;
```

```
*) units="";;
esac
qsub_args="$qsub_args -lm $2$units"
qsub_args="$qsub_args -lM $2$units"
have_lm=yes; have_lM=yes
shift; shift;;
-r) argname="-r $2"
shift; shift;;
-t) # time limit
qsub_args="$qsub_args -lt $2 -lT $2"
have_lt=yes; have_lT=yes
shift; shift;;
-[eopqsCL]|-1?|-1??|-mu) # opts with args
qsub_args="$qsub_args $1 $2"
shift; shift;;
-ec|-[kmmr]?|1U??|-x) # opts w/o args
qsub_args="$qsub_args $1"
shift;;
-z) silent=yes; shift;;
-) shift;;
--) shift; break;; # optionally ends args
*) break;; # found beginning of command
esac
done

# append arguments from $HOME/qexec.rc
# if they haven't already been specified

task=find_next_arg
for rc in $qexec_rc
do
case $rc in
-m) task=parsing_m;;
-r)
case $have_r in
yes) task=skip;;
no) task=parsing_r;;
esac;;
-t) task=parsing_t;;
-lm) task=parsing_lm;;
-lM) task=parsing_lM;;
-lt) task=parsing_lt;;
-lT) task=parsing_lT;;
-[aeopqsCL]|-1?|-1??|-mu)
task=copy
for new_arg in $qsub_args
do
case $new_arg in
$rc) task=skip;;
esac
done
case $task in
copy)
qsub_args="$qsub_args $rc";;
esac;;
-ec|-[kmmr]?|1U??|-x)
task=copy
for new_arg in $qsub_args
```

```

do
  case $new_arg in
    $rc) task=skip;;
  esac
done
case $task in
  copy) qsub_args="$qsub_args $rc";;
  esac;;
-z) silent=yes;;
--) break;;
*) case $task in
  parsing_lm)
    case $rc in
      *[0-9]) units=Kw;;
      *) units="";;
    esac
    case $have_lm in
      no) have_lm=yes
        qsub_args="$qsub_args -lm $rc$units";;
    esac
    task=find_next_arg;;
  parsing_lm)
    case $rc in
      *[0-9]) units=Kw;;
      *) units="";;
    esac
    case $have_lm in
      no) have_lm=yes
        qsub_args="$qsub_args -lm $rc$units";;
    esac
    task=find_next_arg;;
  parsing_m)
    case $rc in
      *[0-9]) units=Kw;;
      *) units="";;
    esac
    case $have_lm in
      no) have_lm=yes
        qsub_args="$qsub_args -lm $rc$units";;
    esac
    case $have_lm in
      no) have_lm=yes
        qsub_args="$qsub_args -lm $rc$units";;
    esac
    task=find_next_arg;;
  parsing_r)
    argname="-r $rc"; have_r=yes
    task=find_next_arg;;
  parsing_lt)
    case $have_lt in
      no) have_lt=yes
        qsub_args="$qsub_args -lt $rc";;
    esac
    task=find_next_arg;;
  parsing_lT)
    case $have_lT in
      no) have_lT=yes
        qsub_args="$qsub_args -lT $rc";;
    esac
    task=find_next_arg;;
  parsing_t)
    case $have_lt in
      no) have_lt=yes
        qsub_args="$qsub_args -lt $rc";;
    esac
    case $have_lT in
      no) have_lT=yes
        qsub_args="$qsub_args -lT $rc";;
    esac
    task=find_next_arg;;
  copy)
    qsub_args="$qsub_args $rc";;
  skip)
    task=find_next_arg;;
  find_next_arg) ;;
  esac ;;
done

case $have_r in
  no) argname="-r $1";;
esac

# create a file for submission. At this
# point, "$*" contains the command line
# that remains after parsing the "qsub"
# arguments.

# cp /bin/time \${QSUB_REQID} causes the time
# process to have the name of the qsub request,
# allowing you to relate process id's to request
# id.

echo >> qexec.log
echo `date` >> qexec.log
echo "qsub $qsub_args $argname" >> qexec.log
echo $*>> qexec.log

qsub $qsub_args $argname >> qexec.log <<!
echo command: $*
echo "submitted thru qexec on " `date`
echo "with qsub $qsub_args $argname "
echo "beginning execution on " `date` \
cp /bin/time \${QSUB_REQID}
echo "\${QSUB_REQID} beginning execution\
on" `date`\`>>qexec.log
\${QSUB_REQID} $*
echo "\${QSUB_REQID} finishing execution\
on" `date`\`>>qexec.log
echo "end command execution on" `date` \
rm \${QSUB_REQID}
!

case $silent in
  no) tail -1 qexec.log;;
esac

```


ATTENDEE LIST

CRAY User Group Attendee List

San Diego, Spring 1994

Adair, Bill
Exxon Eutec
P.O. Box 4449
Houston TX 77210-4449
USA
Phone: 713 965-7638
Fax: 713 965-7477

Adamec, Steve
USAE Waterways Experiment
Station
Information Technology Laboratory
Attn: CEWES-IM-H
3909 Halls Ferry Road
Vicksburg MS 39180-6199
USA
Phone: 601 634-2901
Fax: 601 634-2331
Email: adamec@wes.army.mil

Adams, Jerry
Cray Research, Inc.
Customer Service
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Adeff, Sergio
University of Mississippi
Center for Supercomputing
Research
Powers Hall, Room 109
University MS 38677
USA
Phone: 601 232-7206
Fax: 601 232-7180

Aguila, Jordi
Cesca
Marketing
Avda. Diagonal 645
Barcelona 08028
Spain
Phone: 93 491 38 35
Fax: 93 490 46 35
Email: zdijav01@
puigmal.cesca.es

Allen, Portia
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Almond, James C.
University of Texas System CHPC
Center for High Performance
Computing
10100 Burnet Road, Commons
Building 1.154
Austin TX 78758-4497
USA
Phone: 512 471-2472
Fax: 512 471-2445
Email: j.almond@
hermes.chpc.utexas.edu

Amiot, Mary
Cray Research, Inc.
Marketing
655 A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3524
Fax: 612 683-3599
Email: mary.amiot@cray.com

Anderson, Alfred
University of Texas
CHPCm CMS 1.154
10300 Jollyville Road #633
Austin TX 78759
USA
Phone: 512 471-2463
Fax: 512 471-2445
Email: a.anderson@
chpc.utexas.edu

Anderson, Mike
Cray Research, Inc.
Corporate Marketing
655-A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Anderson, Paul
Department of Defense
9800 Savage Road
Ft. Meade MD 20755
USA
Phone: 301 688-9519

Audemard, Michele
IDRIS/CNRS
System Management
Bat. 506, BP #167
Orsay Cedex 91403
France
Phone: 33 69 82 41 27
Fax: 33 69 28 52 73
Email: audemard@idris.fr

Azar, Alex
Cray Research France
Marketing
18, Rue de Tilsitt
Paris 75017
France
Phone: 16 1 44 09 14 41
Fax: 16 1 44 09 1404
Email: aa@cray.com

Balog, Douglas
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh PA 15213
USA
Phone: 412 268-4960
Fax: 412 268-5832
Email: balog@cpwsca.psc.edu

Bamber, Gail
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Barnes, David
NASA/ Langley Research Center
ACD
MS 157B
Hampton VA 23681-0001
USA
Phone: 804 864-7389
Fax: 804 864-7604
Email: d.b.barnes@
express.larc.nasa.gov

Barriuso, Ray
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Baumbaugh, R. Suzette
University of Mississippi
Center for Supercomputing
Research
Powers Hall, Room 322
University MS 38677
USA
Phone: 601 232-7206
Fax: 601 232-7180

Beck, Susan
Cray Research, Inc.
P.O. Box 51703
Knoxville TN 37950
USA
Phone: 615 690-8869
Fax: 615 631-2224

Beckley, Donna
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Bedoll, Bob
Boeing Computer Service
Research and Technology
P.O. Box 24346, MS 7L-48
Seattle WA 98124-0346
USA
Fax: 206 865-2965
Email: rfb@sd.c.boeing.com

Belbot, Norm
Cray Research, Inc.
Customer Service
4041 Powder Mill Road
Calverton MO 20705
USA
Phone: 301 595-2605
Fax: 301 595-2637
Email: belbot@castrg1

Benwell, Peter R.
United Kingdom Meteorological
Office
Central Computing
London Road
Bracknell Berkshire RG12 2SZ
UK
Phone: 44 344-85 6097
Fax: 44 344-85 4412

Bergman, Larry A.
Jet Propulsion Laboratory
Section 341
4800 Oak Grove Drive, MS 300-
329
Pasadena CA 91109
USA
Phone: 818 354-4689
Fax: 818 393-4820
Email: larry@jpl.nasa.gov

Berkey, Chet
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Bernardi, Sergio
CINECA
Systems Management
Via Magnanelli 6/3
Casalecchio Di Reno (BO I-40033
Italy
Phone: 39 51 598411/ 6599411
Fax: 39 51 598472 /6599472
Email: abatu01@cinymp.cineca.it

Bernhard, Win
General Electric - Aircraft Engines
8700 Governor's Hill
Mail Stop A303
Cincinnati OH 45249
USA
Phone: 513 583-3687
Fax: 513 583-3652

Bielli, Dominique
METEO-France
SCEM/TTI/DEV
42 Avenue Coriolis
Toulouse Cedex 31057
France
Phone: 19 33 61 07 81 22
Fax: 19 33 61 07 81 09
Email: bielli@metro.fr

Blaskovich, David
Cray Research, Inc.
Corporate Marketing Suite 1331
North
1331 Pennsylvania Avenue, NW
Washington DC 20004
USA
Phone: 202 638-6000
Fax: 202 638-0820

Blay, Christopher
Lockheed Info Tech Co.
High Performance and Classified
Computing
1401 Del Norte Street
Denver CO 80221
USA
Phone: 303 430-2122
Fax: 303 430-2225
Email: cblay@
hpcc,litc.lockheed.com

Bodzin, Henry
Ford Motor Company
Engineering Computer Center MD-
1
20,000 Rotunda Drive, ECC
Building
Dearborn MICH 48121
USA
Phone: 313 845-1347
Fax: 313 390-4865
Email: bodzin@
pms625.pms.ford.com

Boland, Bob
Los Alamos National Laboratory
Distributed Computing
Environments MS B272
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 667-1729
Fax: 505 665-6333
Email: wrb@lanl.gov

Bongiorno, Vito
Cray Research, Inc.
Central Region
1440 Northland Drive
Mendota Heights MN 55120
USA
Phone: 612 683-3404
Fax: 612 683-7483
Email: vb@.cray.com

Boyle, Tom
Cray Research, Inc.
Americas Tech. - Support
655 F Lone Oak Dr.
Eagan MN 55121
USA
Phone: 612 683-5695
Fax: 612 683-5599
Email: boyle@hemlock.cray.com

Brady, Chris
Cray Research, Inc
Customer Service- NCAR
1850 Table Mesa Drive
Boulder CO 80307
USA
Phone: 303 497-1836
Fax: 303 494-4002

Brandt, Ruediger
Debis SH CCS
CCS-SW/HPC
Fasenenweg 9
Leinfeldten D-70771
Germany
Phone: 49 711 9722193
Fax: 49 711 9721955
Email: sde4220@str.daimler-
benz.com

Breinlinger, Helmut
Leibniz-Rechenzentrum Muenchen
Barer Str. 21
Muenchen D-80333
Germany
Phone: 49 89 21058788
Fax: 49 89 2809460
Email: breinlinger@
lrz-muenchen.de

Brewer, Kevin
Mobil-MEPTEC
Information Technology
P.O. Box 650232 B-2-236
Dallas TX 75265
USA
Phone: 214 951-3506
Fax: 214 951-3529
Email: krbrewer@dal.mobil.com

Brost, Jerry
Cray Research, Inc.
Engineering
900 Lowater Road
Chippewa Falls WI 54729
USA
Phone: 715 726-6508
Fax: 715 726-6713

Brown, Denice
U.S. Army Research Lab
Attn: AMSRL-CI-AC
Bldg 328, Room 3
Aberdeen Proving Ground MD
21005-5067
USA
Phone: 410 278-6269
Fax: 410 278-5077
Email: denice@arl.army.mil

Brown, Troy
Exxon Eutec
3616 Richmond Avenue
Houston TX 77046-3604
USA
Phone: 713 965-4660
Fax: 713 965-7310

Brunzell, Barb
Cray Research, Inc.
Customer Service
655-A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Buerger, Paul
Ohio Supercomputer Center
1224 Kinnear Rd
Columbus OH 43212-1154
USA
Phone: 614 292-8447
Fax: 614 292-7168
Email: kevin@osc.edu

Buisan, Marc
CNES
18 Av E. Belin
Toulouse 31055
France
Phone: 33 61 27 47 41
Fax: 33 61 28 16 62
Email: buisan@melies.cnes.fr

Busch, Hubert
ZIB Berlin
Computer Center
Heilbronner Str. 10
Berlin-Wilmersdorf D-10711
Germany
Phone: 49 30 89604 135
Fax: 49 30 89604 125
Email: busch@ZIB-Berlin.de

Campbell, Colin
Cray Research UK
Oldbury
Bracknell Berkshire RG124TQ
UK
Phone: 44-344 722190
Fax: 44-344 426319
Email: cjc@cray.com

Caputo, Gina
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Cardo, Nicholas P.
Sterling Software, Inc.
NASA/Ames Research Center
M/S 233-3
Moffett Field CA 94035-1000
USA
Phone: 415 604-4754
Fax: 415 964-1760
Email: npcardo@ames.arc.nasa.gov

Carll, Rich
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Carlson, John
Cray Research, Inc.
655 A Lone Oak Dr.
Eagan MN 55121
USA
Phone: 612-683-7120
Fax: 612-683-7199
Email: John.Carlson@cray.com

Carpenter, John
Cray Research, Inc.
Corporate Marketing
655-A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Carrillo, Alex
USAE Waterways Experiment
Station
Information Technology Laboratory
Attn: CEWES-IM-H
3909 Halls Ferry Road
Vicksburg MS 39180-6199
USA
Phone: 601 634-2588
Fax: 601 634-2331

Carruthers, Bob
Cray Research, Inc.
Corporate Marketing UK
Oldbury
Bracknell Berks RG12 4TQ
United Kingdom
Phone: 44 344 485971
Fax: 44 344 426319

Cave, Robert
Institute for Defense Analyses
Center for Communications
Research
Thanet Road
Princeton NJ 08540
USA
Phone: 609 924-4600
Fax: 609 924-3061
Email: bob@ccr-p.ida.org

Chahine, Roger
Hydro-Quebec
855 EST St. Catherine 20e
Montreal Quebec H2L 4P5
Canada
Phone: 514 840-3027
Fax: 514 840-4160
Email: chahine@cant.hydro.qc.ca

Charon, Elizabeth
CEA/ CEL V
DMA/SIE
94195 Villeneuve
Saint Georges Cedex
France
Phone: 33 1 45 95 61 84
Fax: 33 1 45 95 95 55

Cheung, Henry
CSE
719 Heron Road
Ottawa Ontario K1G 3Z4
Canada
Phone: 613 991-7173
Fax: 613 991-7323
Email: hccheun@manitou.cse.dnd.ca

Cho, YoungWook
SERI/ KIST
Software Engineering and System
Software
P.O. Box Yuseong
Taejeon 305-600
Korea
Phone: 82 042 869-1653
Fax: 82 042 869-1699
Email: ywcho@kumdori.seri.re.kr

Christoph, Gary
Los Alamos National Laboratory
Group C-1 MS B252
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 667-3709
Fax: 505 665-6333
Email: ggc@lanl.gov

Ciotti, Bob
NASA Ames Research Center
MS 258-5
NAS Facility
Moffett Field CA 94035-1000
USA
Phone: 415 604-4408
Fax: 415 604-4377
Email: ciotti@nas.nasa.gov

Ciuffini, Mary Ann
NCAR
P.O. Box 3000
Boulder CO 80307
USA
Phone: 303 497-1806
Fax: 303 497-1818
Email: mac@ncar.ucar.edu

Clark, Charlie
Cray Research, Inc.
European Region 4 Customer
Service
Oldbury
Bracknell Berks RG12 4TQ
United Kingdom
Phone: 44 344 722296
Fax: 44 344 722191

Clave, Salvador
Cesca
Projects
Avda. Diagonal 645
Barcelona 08028
Spain
Phone: 93 491 38 35
Fax: 93 490 46 35
Email: zdiscm01@puigmal.cesca.es

Cohen, Phil
Scripps Research Institute
10666 North Torrey Pines Road
MB200R
La Jolla CA 92037
USA
Phone: 619 554-9914
Fax: 619 554-6260
Email: phil@scripps.edu

Cole, John
U.S. Army Research Lab
Advanced Computing Division Attn:
AMSRL-CI-A
Bldg 328, Room 26.
Aberdeen Proving Ground MD
21005
USA
Phone: 410 278-9276
Fax: 410 278-5077
Email: cole@arl.army.mil

Collinet, Philippe
IDRIS/CNRS
Bat. 506, BP #167
Orsay Cedex 91403
France
Phone: 33 69 82 41 27
Fax: 33 69 28 52 73
Email: collinet@idris.fr

Copeland, Rene
Cray Research, Inc.
Supercomputer Operations
1440 Northland Drive
Mendota Heights MN 55120
USA
Phone: 612 683-5922
Fax: 612 683-7345

Corbett, Dorothy
Arctic Region Supercomputing
Center
P.O. Box 756020
Fairbanks AK 99775
USA
Phone: 907 474-5102
Fax: 907 474-5494
Email: dscorbett@acad5.alaska.edu

Councill, Carolyn
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh PA 15213
USA
Phone: 412 268-1556
Fax: 412 268-5832
Email: councillor@psc.edu

Crain, Sylvia
Cray Research, Inc.
Software Development
500 Montezuma; Suite 118
Santa Fe NM 87505
USA
Phone: 505 988-2468 x30
Fax: 505 984-1375

Craw, James
NASA Ames Research Center
MS 258-6
NAS Facility
Moffett Field CA 94035-1000
USA
Phone: 415 604-4606
Fax: 415 604-4377
Email: craw@nas.nasa.gov

Crawford, Dianna
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Crawford, Susan
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Curtis, Nick
Idaho National Engineering
Laboratory
Scientific Systems
P.O. Box 1625
Idaho Falls ID 83415-2603
USA
Phone: 208 526-9687
Fax: 208 526-9936
Email: nic@inel.gov

Dagitz, Roger
Cray Research, Inc.
Customer Service
890 Industrial Blvd.
Chippewa Falls WI 54729
USA
Phone: 715 726-5269
Fax: 715 723-4343
Email: roger.dagitz@cray.com

Dalmas, George
U.S. Govt C.I.A.
MS 2V29
Room 2V29, NHB
Washington DC 20505
USA
Phone: 703 874-2765
Fax: 703 883-9053

Das, Simanti
Exxon Eutec
CIT GP3-728233-Benmar
233 Benmar
Houston TX 77002
USA
Phone: 713 423-7236
Fax: 713 423-7801

Davey, Sandy
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Davis, Mike
Cray Research, Inc.
Customer Service
6565 Americas Pkwy NE
Albuquerque NM 87110
USA
Phone: 505 844-1034
Fax: 505 844-2067

Declerk, Michael
IDA
Thanet Road
Princeton NJ 08540
USA
Phone: 609 924-4600
Fax: 609 924-3061

Dent, David
European Centre for Medium
Range Weather Forecasts
Shinfield Park
Reading Berkshire RG2 9AX
UK
Phone: 44 734-499702
Fax: 44 734-869450
Email: ddent@ecmwf.co.uk

Dispen, Arve
University of Trondheim Computing
Centre
Sintef Runit
Trondheim N-7034
Norway
Phone: 47 73 592989
Fax: 47 73 591700
Email: dispen@cray.sintef.no

Doering, Don
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Dombrowski, Jay
San Diego Supercomputer Center
Eng.&Oper.
P.O. Box 85608
San Diego CA 92186-9784
USA
Phone: 1619 534 5023
Fax: 1619 534 5152
Email: dombrowhf@sdsc.edu

Dorgan, John
NIST
Computer Services Division
Bldg. 225, Room A27
Gaithersburg MD 20899-0001
USA
Phone: 301 975-2959
Email: dorgan@tiber.nist.gov

Douglas, Margaret
NSWCDD
K51
Navy/NSWC
Dahlgren VA 22448
USA
Phone: 703 663-7334
Fax: 703 663-7999
Email: mdougla@relay.nswc.navy.mil

Drobnis, Dan
San Diego Supercomputer Center
Engineering & Operations
P.O. Box 85608 SDSC-103
San Diego CA 92186-9784
USA
Phone: 619 534-5020
Fax: 619 534-5152
Email: Drobnisdd@sdsc.sds.edu

Dungworth, Mick
Cray Research, Inc.
V.P.- Customer Service
1440 Northland Drive
Mendota Heights MN 55120
USA
Phone: 612 683-7150
Fax: 612 683-7345

Dunn, Christopher
G.C.H.Q.
Computer Operations MS F/0302B
Priors Road, Oakley
Cheltenham Glos GL52 5AJ
United Kingdom
Phone: 44 242 221491 x2381
Fax: 44 242 226816

Dunn, Thomas
Naval Meteorology and
Oceanography Command
Mail Code 00C
1020 Balch Boulevard
Stennis Space Center MS 39529-
50
USA
Phone: 601 688-4189
Fax: 601 688-4880
Email: dunn@pops.navy.mil

Eacock, Nigel A.
Government Communications
Headquarters
F/1212
Priors Road
Cheltenham Gloucestershire
GL52 5AJ
UK
Phone: 44 242-221491
Fax: 44 242-226816

Edge, Curtis D.
North Carolina Supercomputing
Center
P.O. Box 12889
Research Triangle Park NC
27709
USA
Phone: 919 248-1148
Fax: 919 248-1101
Email: edge@ncsc.org

Eitgroth, Peter
Lawrence Livermore National
Laboratory
Physics MS L-294
P.O. Box 808
Livermore CA 94551
USA
Phone: 510 422-4096
Fax: 510 423-6961
Email: eitgroth@lanl.gov

Engel, James
NASA/Johnson Space Center
Grumman Data Systems
12000 Aerospace Ave.
Houston TX 77034
USA
Phone: 713 483-5894
Fax: 713 483-7044
Email: engel@sed.jsc.nasa.gov

Erickson, David
Cray Research, Inc.
Customer Service
890 Industrial Blvd,
Chippewa Falls WI 54729
USA
Phone: 715 726-5308
Fax: 715 726-4343

Escobar, Juan
IDRIS/CNRS
Users Support
Bat. 506, BP #167
Orsay Cedex 91403
France
Phone: 33 69 82 42 00
Fax: 33 69 28 52 73
Email: escobar@idris.fr

Evans, Roger
Rutherford Appleton Laboratory
Chilton
Didlot OX11 0QX
Great Britain
Phone: 44 235 445656
Fax: 44 235 446626
Email: rge@ib.ql.ac.uk

Eversole, Larry C.
Jet Propulsion Laboratory
Cal Tech MS 301-455
4800 Oak Grove Drive
Pasadena CA 91109
USA
Phone: 818 354-2786
Fax: 818 393-1187
Email: eversole@
voyager.jpl.nasa.gov

Ewald, Robert
Cray Research, Inc.
900 Lowater Road
Chippewa Falls WI 54729
USA
Phone: 715-726-6508
Fax: 715 726-6713
Email: Bob.Ewald@cray.com

Fagan, Mike
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Fairhurst, Helen
United Kingdom Meteorological
Office
London Road
Bracknell Berkshire RG12 2SZ
UK
Phone: 44 34 856097
Fax: 44 34 854412

Falcione, Dean
Westinghouse- Bettis
RT
P.O. Box 79
W. Mifflin PA 15122-0079
USA
Phone: 412 476-6535
Fax: 412 476-6924
Email: falcione@bettis.gov

Falde, Paul
Cray Research, Inc.
655F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599
Email: falde@cray.com

Falkenthal, John
DCIX
2100 Main Street
Huntington Beach CA 92648
USA

Ferber, Daniel
Cray Research, Inc.
665F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683 3522
Fax: 612 683 3599

Fichtel, Hartmut
Deutsches Klimarechenzentrum
GmbH
Systems Software
Bundesstrasse 55
D-20146 Hamburg 13
Germany
Phone: 49-40-41173-220
Fax: 49-40-41173-270
Email: fichtel@dkrz.d400.de

Fine, Martin
Merrill Lynch
Swaps
World Financial Center, North
Tower, 16th Floor
New York NY 10281-1316
USA
Phone: 212 444-6203
Fax: 212 444-6789
Email: fine@repo-dev.ml.com

Finn, Steven
Pacific-Sierra Research
2901 28th Street
Santa Monica CA 90405
USA
Phone: 310 314-2332
Fax: 310 314-2323
Email: sf@lanl.gov

Finney, Christopher
EMASS
Sales
51 Corporate Woods, 9393 W.
110th Street
Overland Park KS 66213
USA
Phone: 913 451-6989

Fischer, Ericka
University of Stuttgart Computing
Center
Operations
Allmandring 30
Stuttgart D-70550
Germany
Phone: 49 711 685-4515
Fax: 49 711 68 2357
Email: e.fischer@
rus.uni-stuttgart.de

Fischer, Uwe
Rechenzentrum der Universitaet
Stuttgart
Allmandring 30A
D-7000 Stuttgart 80
Germany
Phone: 49 711 685 5800
Fax: 49 711 682 357
Email: uwe.fischer@
rus.uni-stuttgart.de

Friedman, Karen
National Center for Atmospheric
Research
P.O. Box 3000
Boulder CO 80307
USA
Fax: 303 497-1298

Fry, Wes
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Frybarger, Jim
Los Alamos National Laboratory
Computer Operations and
Assurance Group MS B292
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 667-4584
Fax: 505 665-8816
Email: jaf@lanl.gov

Furtney, Mark
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Gaffey, Brian
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Gates, Kathy
University of Mississippi
Center for Supercomputing
Research
Powers Hall, Room 105
University MS 38677
USA
Phone: 601 232-7206
Fax: 601 232-7180

Gates, Leary
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Georgi, Gunter
Grumman
73 Irma Avenue
Port Washington NY 11050
USA
Phone: 516 883-2336
Email: georgi@cug.org

Gerard, Nicole
Electricite de France
DER-IMA
1 Avenue Du General De Gaulle
Clamart 92141
France
Phone: 33 1 47 65 45 53
Fax: 33 1 47 65 39 73
Email: nicole.gerard@der.edf.fr

Gerth, Mitch
Phillips Petroleum Company
Information Technology
1110 PLaza Office Building
Bartlesville OK 74004
USA
Phone: 918 661-3971
Fax: 918 661-5250
Email: jmgarth@ppco.com

Gigandet, Martine
CEA/CEL V
DMA/SIE
94195 Villeneuve
Saint Georges Cedex
France
Phone: 33 1 45 95 61 84
Fax: 33 1 45 95 95 55
Email: gigandet@limeil.cea.fr

Gordon, Daniel
IDA Center for Communications
Research
4320 Westerra Court
San Diego CA 92121
USA
Phone: 619 622-5431
Fax: 619 455-1327
Email: gordon@ccrwest.org

Gottschewski, Juergen
Konrad Zuse-Zentrum fur
Informationstechnik Berlin
Heilbronner Str. 10
Berlin-Wilmersdorf D-10711
Germany
Phone: 49-30-89604-130
Fax: 49-30-89604-125
Email: Gottschewski@sc.ZIB-
Berlin.de

Graffunder, Sara
Cray Research, Inc.
Applications
655-E Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Grasseau, Giller
IDRIS/CNRS
Users Support
Bat. 506, BP #167
Orsay Cedex 91403
France
Phone: 33 69 82 42 50
Fax: 33 69 28 52 73
Email: grasseau@idris.fr

Grassi, Charles
Cray Research, Inc.
Benchmarking
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Greenwade, Eric
INEL - EG&G Idaho, Inc.
1155 Foote Drive
P.O. Box 1625, MS 2608
Idaho Falls ID 83415
USA
Phone: 208 526-1276
Fax: 208 526-9936
Email: leg@inel.gov

Griffing, R. Bruce
Lawrence Livermore National
Laboratory
NERSC
P.O. Box 5509
Livermore CA 94551
USA
Phone: 510 422-4498
Fax: 510 422-1482
Email: griffing@nersc.gov

Grindle, Jim
Cray Research, Inc.
Software Development-UNICOS
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Guest, Clayton
Sterling Software (NASA Ames)
MS 258-6
NAS Facility
Moffett Field CA 94035-1000
USA
Phone: 415 604-1339
Fax: 415 964-1760
Email: cguest@
eagle.arc.nasa.gov

Guritz, Richard
Arctic Region Supercomputing
Center
P.O. Box 73685
Fairbanks AK 99775
USA
Phone: 907 474-6307
Fax: 907 474-5494
Email: rguritz@
iias.images.alaska.edu

Guzy, Christine
NCAR
SCD
1850 Table Mesa Drive
Boulder CO 80303
USA
Phone: 303 497-1826
Fax: 303 497-1814
Email: guzy@ncar.ucar.edu

Haerer, Sally
NCAR
SCD Consulting Group
P.O. Box 3000
Boulder CO 80307
USA
Phone: 303 497-1283
Fax: 303 497-1298
Email: haerer@ncar.ucar.edu

Hale, Cherie
Los Alamos National Laboratory
Group C6
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 667-2879
Fax: 505 665-5402
Email: cbh@lanl.gov

Hale, Gerald
Los Alamos National Laboratory
T-2 MS B243
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 667-7738
Fax: 505 667-9671
Email: ghale@lanl.gov

Hall, Bonnie
Exxon Eutec
ST-967
P.O. Box 4449
Houston TX 77210-4449
USA
Phone: 713 966-6031
Fax: 713 965-7477

Hampton, Mary
USAE Waterways Experiment
Station
Information Technology Laboratory
Attn: CEWES-IM-MI-C
3909 Halls Ferry Road
Vicksburg MS 39180-6199
USA
Phone: 601 634-3501
Fax: 601 634-2331
Email: hampton@wes.army.mil

Hanyzewski, Gary
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Hardesty, Carol Cannon
Lawrence Livermore National
Laboratory
NERSC
P.O. Box 5509
Livermore CA 94550
USA
Phone: 510 422-9037
Fax: 510 423-5951
Email: hardesty@nersc.gov

Harrell, Jim
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Haven, Will
British Aerospace
W302M, warton Aerodrome
Preston Lancashire PR5 6AC
UK
Phone: 44 772 852230
Fax: 44 772 852787

Haxby, Andy
Shell Common Information
Services
Cray and Unix Support
Rowlandsway House, Rowlands
Way PS/ 23
Wythenshawe Manchester M22
5SB
UK
Phone: 44 61 499 4910
Fax: 44 61 499 4914
Email: andy@trumpet.demon.co.uk

Hazlewood, Victor
Texas A&M University
C15
012 Teague 3142
College Station TX 77840-3142
USA
Phone: 409 862-4121
Fax: 409 847-8643
Email: victor@tamymp.tamu.edu

Heib, Michael
Debis SH CCS
CCS-SW/HPC
Fasenenweg 9
Leinfeld D-70771
Germany
Phone: 49 711 9722180
Fax: 49 711 9721955
Email: sde4220@str.daimler-benz.com

Heinzel, Stefan
Max Planck Institut fuer
Plasmaphysik
Boltzmannstrasse 2
Garching 85748
Germany
Phone: 049 89 3299 1340
Fax: 049 89 3299 2200
Email: sth@ipp-garching.mpg.de

Henese, Mike
Cray Research, Inc.
N.E. District Manager
655-A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Henken, Gerrit
Deutsches Klimarechenzentrum
GmbH
Communication
Bundesstr. 55
Hamburg D-21035
Germany
Phone: 49 40 41173-330
Fax: 49 40 41173-270
Email: henken@dkrz.d400.de

Henquel, Patrick
CNES
18 Au Edouard Belin
Toulouse Cedex 31055
France
Phone: 33 61 28 20 48

Henry, Olivier
CEA/CEV-M
DMA/AMA
Boite Postale 7
Courty 77181
France
Phone: 33 1 49 36 89 12

Hessler, Gerd
University of Kiel
Computer Center
Olshausenstrasse 40
Kiel D-24118
Germany
Phone: 49 431-8802770
Fax: 49 431-8801523
Email: hessler@rz.uni-kiel.d400.de

Hilberg, Claus
European Centre for Medium
Range Weather Forecasts
Shinfield Park
Reading Berkshire RG2 9Ax
UK
Phone: 44 734-499000
Fax: 44 734-869450
Email: claus.hilberg@ecmwf.co.uk

Hines, Gary
Cray Research, Inc.
Corporate Marketing
655-A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Hinrichs, Emil
Prakla-Seismos GmbH
Geophysical Data Center
Bucholzer Str. 100
D-30655 Hannover 51
Germany
Phone: 49 511-642-4227
Fax: 49 511-647-6860
Email: hinrichs@annover.sgp.slb.com

Hochlenert, Juergen
Cray Research GmbH
Service
Riesstrasse 25
Munchen D-80992
Germany
Phone: 49 89 14903 130
Fax: 49 89 14903 149
Email: jho@crmunich0.cray.com.de

Holzmaier, Walter
Cray Research GmbH
Service
Riesstrasse 25
Munchen D-80992
Germany
Phone: 49 89 14903 134
Fax: 49 89 14903 149
Email: wgh@crmunich0.cray.com.de

Horner-Miller, Barbara
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena CA 91355
USA
Phone: 818 354-3434
Fax: 818 393-1187
Email: horner@cosmos.jpl.nasa.gov

Hung, Howard
National Institute of Standards and
Technology
Scientific Computing Environments
Division
MS B146, Bldg 225
Gaithersburg MD 20899
USA
Phone: 301 975-2890
Fax: 301 963-9137
Email: hung@cam.nist.gov

Hutton, Tom
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152
Email: Hutton@sds.sdsc.edu

Jamgochian, Linda
EMASS
Marketing MS 58100
2260 Merritt Drive
Dallas TX 75266
USA
Phone: 214 205-7762
Email: lindaj@emass.esy.com

Jastremski, Bruce
Cray Research, Inc.
Local Sales Management
222 N. Sepulveda Blvd. Suite 1406
El Segundo CA 90245
USA
Phone: 310 640-8402
Fax: 310 640-8442

Jaunin, Michel
Ecole Polytechnique Federale de
Lausanne
Service-Informatique Central-SE
CP121 Ecublens
CH-1015 Lausanne
Switzerland
Phone: 41-21 693 22 11
Fax: 41-21 693 22 20
Email: jaunin@sic.epfl.ch

Jennings, David
NSWCDD
Navy/NSWC
Dahlgren VA 22448
USA
Phone: 703 663-7334
Fax: 703 663-7999
Email: djennin@relay.nswc.navy.mil

Jensen, Gary
National Center for Atmospheric Research
P.O. Box 3000
Boulder CO 80307
USA
Phone: 303 497-1289
Fax: 303 497-1298
Email: guido@niwot.ucar.edu

Jensen, Nancy
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Jette, Morris
Lawrence Livermore National Laboratory
NERSC L-561
P.O. Box 808
Livermore CA 94550
USA
Phone: 510 423-4856
Email: jette@nsl.gov

Jin, SangWon
Samsung Advanced Institute of Technology
Department of Supercomputer Applications
P.O. Box 111
Suwon 440-600
Korea
Phone: 82 2 744 0011 x 9161
Fax: 82 331 280 9158

Johnson, Frederick
National Institute of Standards and Technology
Bldg. 225/Rm. B122
Gaithersburg MD 20899
USA
Phone: 301 975-2700
Fax: 301 963-9137
Email: fjohnson@nist.gov

Johnson, Steve
Cray Research, Inc.
1050 Lowater Rd
Chippewa Falls WI 54729
USA
Phone: 715 726-8227
Fax: 715 726-6715
Email: sjj@artgate.cray.com

Jones, Kelly
Cray Research, Inc.
890 Industrial Blvd
Chippewa Falls WI 54729
USA
Phone: 715 726-5224
Fax: 715 726-4343
Email: krj@techops.cray.com

Jones, Terry
Grumman Data System
Bldg 1001, Room 101
Stennis Space Center MS 39522
USA
Phone: 601 688-5289
Fax: 601 689-0400

Kadomatsu, Gary
Cray Research, Inc.
2100 Main Street
Huntington Beach CA 92648
USA
Phone: 714 960-7611
Fax: 714 969-6472
Email: garyk@craywr.cray.com

Kaler, Joe
Cray Research, Inc.
Customer Service
890 Industrial Blvd,
Chippewa Falls WI 54729
USA
Phone: 715 726-5334
Fax: 715 726-4343

Kamrath, Anke
San Diego Supercomputer Center
User Services
P.O.Box 85608
San Diego CA 92186-9784
USA
Phone: 619 534-5140
Fax: 619 534-5117
Email: kamratha@sdsc.edu

Keller, Jayne
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Kikuchi, Mitutaka
Nippon Telegraph and Telephone Corporation
Research and Development
Information and Patent Center
3-9-11 Midori-Cho
Tokyo Musashino-shi 180
Japan
Phone: 81-422 59-3699
Fax: 81-422 60-7480
Email: kikuchi@superm.ntt.jp

Kirchhof, Cal
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Klein, Rosie
Cray Research, Inc.
Corporate Marketing
1440 Northland Drive
Mendota Heights MN 55120
USA
Phone: 612 683-7521
Fax: 612 683-7345

Klepzig, Floyd
University of Mississippi
Center for Supercomputing Research
Supercomputer Building, Room 100A
University MS 38677
USA
Phone: 601 232-7206
Fax: 601 232-7180

Knaak, David
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Koeninger, Kent
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599
Email: kentk@cray.com

Krantz, Daniel
Los Alamos National Laboratory
Computing, Information and Communications Div. MS B294
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 665-4932
Fax: 505 665-6333
Email: dwk@lanl.gov

Krause, Lisa
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Krecji, Fred
Cray Research, Inc.
Customer Service-West
2100 Main Street
Huntington Beach CA 92648
USA
Phone: 714 960-7611
Fax: 714 969-6472

Kuehn, Jeff
NCAR
SCD Consulting Group
P.O. Box 3000
Boulder CO 80307
USA
Phone: 303 497-1311
Fax: 303 497-1814

Kulsrud, Helene
Institute for Defense Analyses
Center for Communications Research
Thanet Road
Princeton NJ 08540
USA
Phone: 609 279-6243
Fax: 609 924-3061
Email: laney@ccr.p.ida.org

Kyriopoulos, Marj
Cray Research, Inc.
Customer Service
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

LaCroix, Suzanne
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Lapeyre, Alain
Centre National d'Etudes Spatiale
18 Au Edouard Belin
Toulouse Cedex 31055
France
Phone: 33 61 28 20 51
Fax: 33 61 28 18 93
Email: lapeyre@sc2000.cnes.fr

Larson, Julie
Cray Research, Inc.
Customer Service
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Lasher, Bill
EMASS
2260 Merritt Drive
Garland TX 75042
USA
Phone: 214 205-8555
Fax: 214 205-7200
Email: bill@emass.esy.com

Lebens, Janet
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Lecoeuvre, Claude
Commissariat a l'Energie Atomique
Cel-V
94195 Villeneuve
St. Georges Cedex
France
Phone: 33 1 45 95 61 85
Fax: 33 1 45 95 95 55
Email: lecoeuvr@limeil.cea.fr

Lee, Yong Woo
SERI/ KIST
Software Engineering and System
Software
P.O. Box Yuseong
Taejeon 305-600
Korea
Phone: 82 042 869-1674
Fax: 82 042 869-1699
Email: ywill@garam.kreonet.re.kr

Lemaire, Frederic
Cray Research France
Service
18, Rue de Tilsitt
Paris 75017
France
Phone: 16 1 44 09 14 41
Fax: 16 1 44 09 1404

Levine, Michael
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh PA 15213
USA
Phone: 412 268-4960
Fax: 412 268-5832
Email: Levine@a.psc.edu

Lin, Carlos
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Litteer, Gerald
INEL-EG&G Idaho, Inc.
1155 Foote Drive ISC G06
P.O. Box 1625
Idaho Falls ID 83415
USA
Phone: 208 526-9117
Fax: 208 526-9936
Email: gll@inel.gov

Lloyd, Harold
National Meteorological Center
Automation Division
DOC/NOAA/NWS FOB-4, Room
2334
Suitland MD 20746
USA
Phone: 301 763-2616
Fax: 301 763-3479

Lovato, Frank
Naval Oceanographic Office
Mail Code NSC
1002 Balch Boulevard
Stennis Space Center MS 39529-
50
USA
Phone: 601 688-5091
Fax: 601 689-0400
Email: lovato@
pops.navo.navy.mil

Mack, Dieter
Rechenzentrum der Universitaet
Stuttgart
Systems
Allmandring 30
Stuttgart D-70550
Germany
Phone: 49 711 685-5788
Fax: 45 711 68 2357
Email: mack@rus.uni-stuttgart.de

Mandell, Jeff
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Mandt, Hans
Boeing Advanced Systems
Laboratory
P.O. Box 24346
MS 7L-48
Seattle WA 98124
USA
Phone: 206 866-3505
Fax: 206 865-2965
Email: hans@skipper.boeing.co
m

Mantock, Gregg
Eli Lilly and Company
Lilly Corporate Center
Scientific Information Systems
Indianapolis IN 46285
USA
Phone: 317 276-4269
Fax: 317 276-4127

Marsden, Yuki
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Martin, Stuart
Rutherford Appleton Laboratory
Atlas Centre
Chilton Didcot Oxon OX11 0QX
England
Phone: 44 235 446780
Fax: 44 235 446626
Email: sjm4@ib.rl.ac.uk

Mascarenas, Art
LANL
MS B294
Los Alamos NM 87545
USA
Phone: 505 667-7191
Email: adm@lanl.gov

Mason, Ange
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186-9784
USA
Phone: 619 534-8333
Fax: 619 534-5152
Email: masona@sdscc.edu

Mason, Delores
National Energy Research
Supercomputer Computer Center
NERSC
P.O. Box 808 L-560
Livermore CA 94550
USA
Phone: 510 422-9325
Fax: 510 423-8744
Email: mason@nersc.gov

Mason, Don
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Matthews, Kevin
Cray Research, Inc.
655F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-5422
Fax: 612 683-5201
Email: kcm@cray.com

McLaughlin, Dan
Cray Research, Inc.
Customer Service
890 Industrial Blvd.
Chippewa Falls WI 54729
USA
Phone: 715 726-5067
Fax: 715 726-4343

Melendez, Jerry
Los Alamos National Laboratory
Computer Systems
P.O. Box 1663, MS B294
Los Alamos NM 87545
USA
Phone: 505 667-5243
Fax: 505 665-6333
Email: kjm@lanl.gov

Mengel, Don
Cray Research, Inc.
Customer Service
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Meys, Tony
Cray Research, Inc.
Applications
655-E Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Miller, Raymond
Los Alamos National Laboratory
C-2
P.O. Box 1663 MS B294
Los Alamos NM 87545
USA
Phone: 505 665-3222
Fax: 505 665-6333

Miller, Robin
Mississippi Center for Computing
Research
205 Supercomputing Building
University of Mississippi
University MS 38677
USA
Phone: 601 232-7206
Fax: 601 232-7180
Email: robin@vm.cc.olemiss.edu

Milosevich, Sam
Eli Lilly
MC7R7
Lilly Corporate Center, Drop 1513
Indianapolis IN 46285
USA
Phone: 317 276-9118
Fax: 317 276-5431
Email: sam@ncsa.uiuc.edu;
sam@lilly.com

Minto, Bill
Cray Research, Inc.
Corporate Marketing/ T3D
655-A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Mitchell, John Cameron
Government Comm.Headquarters
C34C Room F/1211
Priors Road
Cheltenham Gloucestershire
GL525AJ
UK
Phone: 044 242 221491 ext. 3793
Fax: 44 242 251725

Moore, Reagan W.
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186-9784
USA
Phone: 619 534-5073
Fax: 619 534-5152
Email: moore@sdsc.edu

Morin, Michele
Electricite de France
DER-IMA
1 Avenue Du General De Gaulle
Clamart 92141
France
Phone: 33 1 47 65 51 15
Fax: 33 1 47 65 41 18
Email: michele.morin@der.edf.fr

Morreale, Peter
National Center for Atmospheric
Research
SCD Consulting
1850 Table Mesa Drive
Boulder CO 80307
USA
Phone: 303 497-1293
Email: morreale@ncar.ucar.gov

Morrow, Dennis
NASA/Goddard
SDCD
Mail Code 932
Greenbelt MD 20771
USA
Phone: 301 286-2829
Fax: 301 286-1634
Email: morrow@calgary.gsfc.nas
a.gov

Muehling, Eric
Arctic Region Supercomputing
Center
P.O. Box 756020
Fairbanks AK 99775
USA
Phone: 907 474-5149
Fax: 907 474-5494
Email: fnerm@arsc.alaska.edu

Nadrchal, Jaroslav
Czech Academy of Sciences
Institute of Physics
Cukrovarnicka 10
162 00 Praha 6
Czech Republic
Phone: +42 2355 500
Fax: +42 2312 3184
Email: nadrchal@fzu.cs

Nagy, Nicholas
Los Alamos National Laboratory
CIC-DO B260
Los Alamos Nat'l Laboratory P.O.
Box 1663
Los Alamos NM 87544
USA
Phone: 505 667-6164
Fax: 505 665-4361
Email: Nagy@lanl.gov

Nason, John
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Naud, Alain
IDRIS/CNRS
System/Data Management
Bat. 506, BP #167
Orsay Cedex 91403
France
Phone: 33 69 82 41 27
Fax: 33 69 28 52 73
Email: naud@idris.fr

Novotny, Robert
EMASS
Sales
10809 W. 128th Terrace
Overland Park KS 66213
USA
Phone: 913 897-6944

O'Connor, Bill
EMASS
Sales
7716 E. Minnezona Avenue
Scottsdale AZ 85251
USA
Phone: 602 990-3202

O'Neill, Michael
Cray Research U.K.
Applications Support Group
Oldbury
Bracknell Berkshire RG12 4TQ
UK
Phone: 44 344 8485971
Fax: 44 344 57234
Email: mlon@cray.com

Ogawa, Susumu
Cray Research Japan Ltd.
East Japan Sales District
Ichibancho Eight-One Building
64 Ichiban-cho Chiyoda-KU Tokyo
102
Japan
Phone: 81 3 3239 0710
Fax: 81 3 3239 0955
Email: sog@sol.crj.cray.com

Ogno, Anton
Exxon Eutec
10806 Atwell Drive
Houston TX 77096
USA
Phone: 713 965-7308
Fax: 713 965-7310

Ohmura, Kyukichi
CRC Research Institute, Inc.
1-3-D16, Nakase, Mihama-ku
Chiba-shi 261-01
Japan
Phone: +81 43 274 7180
Fax: +81 43 298 1863
Email: k-oomura@crc.co.jp

Olias, Uwe
University of Kiel
Computer Center
Olshausenstrasse 40
Kiel D-24118
Germany
Phone: 49 4318 802 770
Fax: 49 4318 801 523
Email: olias@rz.uni-kiel.d400.de

Olson, Denny
Cray Research, Inc.
Applications
655-E Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Olson, James
Phillips Petroleum Company
1110 PLaza Office Building
Bartlesville OK 74004
USA
Phone: 918 661-3042
Fax: 918 661-9345
Email: jjo@ppco.com

Oner, Fatma
Power Computing Company
1930 Hi Line Drive
Dallas TX 75207
USA
Phone: 214 655-8618
Fax: 214 655-8836

Opalko, Bob
Mississippi Center for
Supercomputing Research
Computer Center
Powers Hall, Room 315
University MS 38677
USA
Phone: 601 232-7206
Fax: 601 232-7180

Oura, Shuhei
Cray Research, Inc.
Yokogawa Electric Corp.
Shinjuku Center Building (50F)
1-25-1 Nishi-shinjukuShinjuku-Ku
Tokyo 163-60
Japan
Phone: 81 3 3349 0617
Fax: 81 3 3349 0697
Email: t_tanaka@
eng.yokogawa.co.jp

Owen, R.
NASA Ames Research Center
MS 258-6
NAS Facility
Moffett Field CA 94035-1000
USA
Phone: 415 604-5935
Fax: 415 964-1760
Email: rkowen@nas.nasa.gov

Oyanagi, Steven
Minnesota Supercomputer Center
1200 Washington Avenue South
Minneapolis MN 55415
USA
Phone: 612 337-3527
Fax: 612 337-3400
Email: sho@msc.edu

Pack, Jeff
Grumman Data Systems
7 Grace Hopper Avenue
Monterey CA 93943-5005
USA
Phone: 408 656-4647
Fax: 408 656-4648
Email: jpack@fnoc.navy.mil

Palm, Joan
Cray Research, Inc.
Corporate Marketing
655 A Lone Oak Dr.
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Parker, Tom
National Center for Atmospheric
Research
SCD Consulting Group
P.O. Box 3000
Boulder CO 80307
USA
Phone: 303 497-1227
Fax: 303 497-1814
Email: tparker@ncar.ucar.edu

Parks, Cathy
Sterling Software (NASA Ames)
MS 258-6
NAS Facility
Moffett Field CA 94035-1000
USA
Phone: 415 604-4768
Fax: 415 604-4377
Email: cparks@nas.nasa.gov

Patella, Rick
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Pellegrino, Fran
Westinghouse Electric Corp
PO Box 355
Computer Services
Pittsburgh PA 15230-0355
USA
Phone: 412 374-4281
Fax: 412 374-4909
Email: pellegrino@b.psc.edu

Perry, Steve
Cray Research, Inc.
Sales
200 Westpark Drive, Suite 270
Peachtree City GA 30269
USA
Phone: 404 631-2235
Fax: 404 631-2224

Peterson, Anthony
EMASS
MS 57230
2260 Merritt Drive
Garland TX 75042
USA
Phone: 214 205-8320
Email: anthony@
emass.esy.com

Petty, James
Grumman Data System
Security
Bldg 1001, Room 101
Stennis Space Center MS 39522
USA
Phone: 601 688-4514
Fax: 601 689-0400

Pfaff, Bruce
NASA Goddard Space Flight
Center
Mail Code 931
Greenbelt MD 20771
USA
Phone: 301 286-8587
Fax: 301 286-1634
Email: k3bep@
charney.gsfc.nasa.gov

Pfaff, Dale
Naval Research Lab
Mail Code 5594
4555 Overlook Avenue, SW
Washington DC 20375-5000
USA
Phone: 202 767-3190
Fax: 202 404-7402
Email: pfaff@ccf.nrl.navy.mil

Polterock, Josh
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Powers, Alan
Sterling Software (NASA Ames)
MS 258-6
NAS Facility
Moffett Field CA 94035-1000
USA
Phone: 415 604-3991
Fax: 415 604-4377
Email: powers@nas.nasa.gov

Price, Robert
Westinghouse Electric Corporation
Westinghouse Corporate Computer
Services
P.O. Box 355, WEC-W204C
Pittsburgh PA 15146
USA
Phone: 412 374-5826
Fax: 412 374-6924
Email: price@a.psc.edu

Puzio, Sylvie
IDRIS/CNRS
Users Graphic Support
Bat. 506, BP #167
Orsay Cedex 91403
France
Phone: 33 69 82 42 00
Fax: 33 69 28 52 73
Email: puzio@idris.fr

Qualters, Irene
Cray Research, Inc.
Software Development
655 F Lone Oak Dr.
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599
Email: imq@cray.com

Raith, Dieter
RUS Rechenzentrum Uni Stuttgart
Computer Center
Allmandring 30
D-70550 Stuttgart
Germany
Phone: 49-711-68545167
Fax: 49-711-682357
Email: raith@rus.uni-stuttgart.de

Raymond, Richard
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh PA 15213
USA
Phone: 412 268-4960
Fax: 412 268-5832
Email: raymond@psc.edu

Reinhardt, Steve
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Rew, Juli
National Center for Atmospheric
Research
SCD
1850 Table Mesa Drive
Boulder CO 80303
USA
Phone: 303 497-1830
Fax: 303 497-1814
Email: juliana@ncar.ucar.edu

Reynolds, John
Lawrence Livermore National
Laboratory
NERSC L-561
P.O. Box 5509
Livermore CA 94550
USA
Phone: 510 422-8350
Fax: 510 422-0435
Email: reynolds@nslsc.gov

Rhodes, Hannah
Cray Research, Inc.
Applications
655-D Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Richmond, Barbara
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Ride, Sally
University of California, San Diego
CAL SPC IN
La Jolla CA 92093
USA

Rittenhouse, Virgil
Cray Research, Inc.
Customer Service-West
2100 Main Street
Huntington Beach CA 92648
USA
Phone: 714 960-7611
Fax: 714 969-6472

Roach, David
University of Mississippi
Center for Supercomputing
Research
Powers Hall, Room 305
University MS 38677
USA
Phone: 601 232-7206
Fax: 601 232-7180

Robb, Derek
Cray Research, Inc.
Corporate Marketing
655-A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Robertson, Michael
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Rohwer, David
Arctic Region Supercomputing
Center
Computer Network
303 Tanana Drive
Fairbanks AK 99775
USA
Phone: 907 474-6319
Fax: 907 474-7127
Email: scdar@orca.alaska.edu

Roiger, Wayne
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Rokkum, Leif Reidar
University of Trondheim Computing
Centre
Industrial Mathematics
University of Trondheim
Trondheim N-7034
Norway
Phone: 47 73 592033
Fax: 47 73 592971
Email: leifreidar.rokkum@
sima.sintef.no

Romberg, Mathilde
KFA Juelich
Postfach 1913
Zentralinstitut fur Angewandte
Mathematik
D-52425 Juelich
Germany
Phone: 49 2461-613631
Fax: 49 2461 616656
Email: m.romberg@kfa-juelich.de

Rosenberg, Robert
United States Navy
Research Computation Division
4555 Overlook Ave., S.W.
Washington DC 20375
USA
Phone: 202 767-3884
Fax: 202 404-7402
Email: Rosenberg2@
ccf.nrl.navy.mil

Rutherford, Paul
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Saffioti, Joe
Communications Security
Establishment
719 Heron Road
Ottawa Ontario K1G 3Z4
Canada
Phone: 613 991-7302
Fax: 613 991-7323

Saini, Subhash
NASA Ames Research Center
MS 258-6
NAS Facility
Moffett Field CA 94035-1000
USA

Saye, Louis
Cray Research, Inc.
925 1st Avenue
Chippewa Falls WI 54729
USA
Phone: 715 723-5501
Fax: 715 723-4980
Email: %00%CHPI@mpls2

Scarbnick, Carl
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Schafer, Hans-Ulrich
Cray Research GmbH
Sales and Marketing Support
Riesstrasse 25
Munchen D-80992
Germany
Phone: 49 89 149030
Fax: 49 89 1409075
Email: hus@cray.com

Schardt, Thomas
NASA Goddard Space Flight
Center
Code 931
Greenbelt MD 20771
USA
Phone: 301 286-9155
Fax: 301 286-1634
Email: k3tds@charney.gsfc.nasa.gov

Schilder, Sandi
Cray Research, Inc.
CCN
655-E Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Schroeder, Wayne
San Diego Supercomputer Center
P. O. Box 85608
San Diego CA 92186-9784
USA
Phone: 619 534-5065
Fax: 619 534-5152
Email: schroeder@sdsc.edu

Schroeder, William
GE CRD
Computer Graphics & Systems MS
KWC-211
1177 Highland Park Road
Schenectady NY 12309
USA
Phone: 518 387-5106
Fax: 518 387-6560
Email: schroeder@crd.ge.com

Schultz, Richard
IDA Center for Communications
Research
4320 Westerra Court
San Diego CA 92121
USA
Phone: 619 622-5420
Fax: 619 455-1327
Email: rich@ccrwest.org

Shaginaw, Richard
Bristol-Myers Squibb
SIS
P.O. Box 4000
Princeton NJ 08543-4000
USA
Phone: 609 252-5184
Fax: 609 252-6163
Email: shaginaw@bms.com

Sharp, Barry
Boeing Computer Technical
Service
Technical Services
P.O. Box 24346 MS 7A-35
Seattle WA 98124-0346
USA
Phone: 206 865-6411
Fax: 206 865-2007
Email: bxs@sdsc.cs.boeing.com

Sheddon, Mark
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186-9784
USA
Phone: 619 534-5130
Fax: 619 534-5152
Email: sheddon@sdsc.edu

Sherman, Tom
Cray Research, Inc.
Applications
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Sheroke, Robert
U.S. Army Research Lab
Bldg 328, Room 3
Aberdeen Proving Ground MD
21005
USA
Phone: 410 278-2064
Fax: 410 278-5077
Email: rsheroke@arl.army.mil

Shorrel, Gary
Cray Research, Inc.
Engineering
1050 Lowwater Road
Chippewa Falls WI 54729
USA
Phone: 715 726-8223
Fax: 715 726-7615

Shukla, Suresh
Boeing Computer Service
Service Management MS 7A-26
P.O. Box 24346
Seattle WA 98124-0346
USA
Phone: 206 865-3482
Fax: 206 865-2007

Shuler, Jean
National Energy Research
Supercomputer Computer Center
NERSC L-561
P.O. Box 5509
Livermore CA 94550
USA
Phone: 510 423-1909
Fax: 510 422-0435
Email: shuler@nersc.gov

Sides, Stephanie
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186-9784
USA
Phone: 619 534-5131
Fax: 619 534-5152
Email: sides@sdsc.edu

Silvia, Robert J.
North Carolina Supercomputing
Center
3021 Cornwallis Road
Research Triangle Park NC
27709
USA
Phone: 919 248-1132
Fax: 919 248-1101
Email: rjs@ncsc.org

Simmons, Margaret W.
Los Alamos National Laboratory
MS B265
P. O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 667-1749
Fax: 505 665-5220
Email: mls@lanl.gov

Sinco, Russ
222 N. Sepulveda Blvd.
El Segundo CA 90245
USA
Email: Sindo@sunnyla.cray.com

Skaug, Dallas
Cray Research, Inc.
Corporate Marketing
655-A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Slocomb, Charles
LANL
CIC-DO
Los Alamos Nat'l Laboratory P.O.
Box 1663
Los Alamos NM 87544
USA
Phone: 505 667-6164
Fax: 505 665-4361
Email: cas@lanl.gov

Slowinski, David
Cray Research, Inc.
Development
900 Lowwater Road
Chippewa Falls WI 54729
USA
Phone: 715 726-6000
Fax: 715 726-6713

Smart, Charlotte
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Smetana, Andrew
Westinghouse Savannah River
Scientific Computations
Bldg 773-42A Rm.127
Aiken SC 29808
USA
Phone: 803 725-4192
Fax: 803 725-4704

Smith, Edward
Apple Computer
20740 Valley Green Drive
M/S #32 E
Cupertino CA 95014
USA
Phone: 408 996-1010
Fax: 408 974-3103

Smith, Liz
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Sotler, Virginia A.
United States Army Corps of
Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg MS 39180-6199
USA
Phone: 601 634-4418
Fax: 601 634-2331
Email: sotler@wes.army.mil

Sova, Mary
Cray Research, Inc.
Customer Service-West
2100 Main Street
Huntington Beach CA 92648
USA
Phone: 714 960-7611
Fax: 714 969-6472

Spencer, Leo
Lawrence Livermore National
Laboratory
Livermore Computing L-67
7000 East Avenue
Livermore CA 94550
USA
Phone: 510 422-0484
Fax: 510 423-6961

Spiller, Jake
Merrill Lynch
Swaps
World Financial Center, North
Tower
New York NY 10281-1316
USA
Phone: 212 449-0056
Fax: 212 449-2724
Email: jake@swaps-ny.ml.com

Spitzmiller, Ted
Los Alamos National Laboratory
C-6 MS B295
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 667-7298
Fax: 505 665-5402
Email: ths@lanl.gov

Spragg, Douglas
Exxon Upstream Technical
Computing Company
P.O. Box 4449
Houston TX 77210
USA
Phone: 713 965-4804
Fax: 713 965-7310

Springmann, Jeanne
NIST
Bldg. 225, Room B146
Gaithersburg MD 20899-0001
USA
Phone: 301 975-3805

St. Charles, Neil
Ford Motor Company
Engineering Computer Center
PO Box 2053
Dearborn MI 48121-2053
USA
Phone: 313 845-5471
Fax: 313 390-4865
Email: stcharle@
pms007.pms.ford.com

St. John, Wally
Los Alamos National Laboratory
C-5 MS B255
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 665-3666
Fax: 505 665-7793
Email: jwb@lanl.gov

Steidel, Jon
Cray Research, Inc.
Software Development
655 F Lone Oak Dr.
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599
Email: jls@bedlam.cray.com

Stern, Stuart
Boeing Advanced Systems
Laboratory
Research and Technology
P.O. Box 24346 MS 7L-48
Seattle WA 98124-0346
USA
Phone: 206 866-3504
Fax: 206 865-2965

Storer, Neil
European Centre for Medium
Range Weather Forecasts
Operations Department
Shinfield Park
Reading Berkshire RG2 9AX
UK
Phone: 011-44-734-499353
Fax: 011-44-734-869450
Email: neil.storer@ecmwf.co.uk

Strand, Brad
Cray Research, Inc.
655F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599
Email: bstrand.cray.com

Strande, Shawn
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Stringer, Ben
COADOD
P.O. Box 4924
Kingston 2604 ACT
Australia
Phone: 61 6 265 0431
Fax: 61 6 265 0485
Email: ben@defcen.gov.au

Swanson, Robert
Martin Marietta Services Group
135 Washington Avenue
Bay City MI 48708
USA
Phone: 517 894-7600
Fax: 517 894-7676
Email: rto@nesc.epa.gov

Swanson, Sandra
Martin Marietta Services Group
135 Washington Avenue
Bay City MI 48708
USA
Fax: 517 894-7600
Email: saz@nesc.epa.gov

Sydow, Pete
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Tanaka, Toshiro
Cray Research, Inc.
Yokogawa Electric Corp.
Shinjuku Center Building (50F)
1-25-1 Nishi-shinjuku
Shinjuku-Ku Tokyo 163-60
Japan
Phone: 81 3 3349 0617
Fax: 81 3 3349 0697
Email: t_tanaka@eng.yokogawa.
co.jp

Thelliez, Eric
Electricite de France
DER-IMA
1 Avenue Du General De Gaulle
Clamart 92141
France
Phone: 33 1 47 65 39 49
Fax: 33 1 47 65 39 73
Email: eric.thelliez@der.edf.fr

Therre, Marie Helene
Cray Research France
Service
18 Rue de Tilsitt
Paris 75017
France
Phone: 16 1 44 09 14 41
Fax: 16 1 44 09 1404

Tiemann, Jochen
DEBIS Systemhaus GmbH
RZ S/S MS HPC G300
Mercedesstrasse 136
Stuttgart D-70322
Germany
Phone: 49 711 1722836
Fax: 49 711 1756764
Email: tiemann@str.daimler-
benz.com

Tovo, Pat
Cray Research, Inc.
655F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599
Email: tovo@mahogany

Tracy, Daniel
Department of Defense
9800 Savage Road
Ft. Meade MD 20755-6000
USA
Phone: 301 688-7206

Umscheid, Lou
Geophysical Fluid Dynamics
Laboratory
Room 161
P.O. Box 308, Princeton University
Princeton NJ 08542
USA
Phone: 609 452-6591
Fax: 609 987-5063
Email: lu@gfdl.gov

Valenzuela, Felipe
Swiss Federal Institute of
Technology
SIC-SE
CH-1015 Lausanne
Switzerland
Phone: 41 21 693 2256
Fax: 41 21 693 2220
Email: valenzuela@sic.epfl.ch

Vandevender, Walter
Sandia National Laboratories
Organization 1943
P.O. Box 5800
Albuquerque NM 87185-5800
USA
Phone: 505 844-4802
Fax: 505 844-2067
Email: whvande@sandia.gov

Vann, Leon
Cray Research, Inc.
655A Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683 3522
Fax: 612 683 3599

Verdier, Francesca
NASA Goddard Space Flight
Center
Mail Code 931
Greenbelt MD 20771
USA
Phone: 301 286-8153
Fax: 301 286-1634
Email: xfrv@calvin.gsfc.nasa.gov

Vernon, Krista
Mississippi Center for
Supercomputing Research
Center for Supercomputing
Research
Powers Hall, Room 303
University MS 38677
USA
Phone: 601 232-7206
Fax: 601 232-7180
Email: cckrista@magnolia.mcsr.olemiss.edu

Vigil, Manuel
Los Alamos National Laboratory
MS B294
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 667-5243
Fax: 505 665-6333

Vildibill, Michael
San Diego Supercomputer Center
P. O. Box 85608
San Diego CA 92186-9784
USA
Phone: 619 534-5074
Fax: 619 534-5152
Email: mikev@sdsc.edu

Vizino, Chad
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh PA 15213
USA
Phone: 412 268-5868
Fax: 412 268-5832
Email: vizino@psc.edu

Vogel, Mary C.
Rockwell International
MS SE25 P.O. Box 2515
2201 Seal Beach Boulevard
Seal Beach CA 90740
USA
Phone: 310 797-2565
Fax: 310 797-3511

Wasserman, Harvey
Los Alamos National Laboratory
Computer Research Group MS
B265
P.O. Box 1663
Los Alamos NM 87545
USA
Phone: 505 667-2136
Fax: 505 665-5220
Email: hjw@lanl.gov

Watson, Jr., Robert
Lockheed Info Tech Company
High Performance Computing
1401 Del Norte Street
Denver CO 80221
USA
Phone: 303 430-2122
Fax: 303 430-2225
Email: bwatson@litc.lockheed.com

Weening, Joseph
IDA Center for Communications
Research
4320 Westerra Court
San Diego CA 92121
USA
Phone: 619 622-5429
Fax: 619 455-1327
Email: jweening@ccrwest.org

Wehinger, Walter
Rechenzentrum der Univ. Stuttgart
Allmandring 30
D-70550 Stuttgart 80
Germany
Phone: 49-711-685-2513
Fax: 49-711-682-357
Email: wehinger@rus.uni-stuttgart.de

Weidl, Ingeborg
Max Planck Institut fuer
Plasmaphysik
Boltzmanstrasse 2
Garching D-85748
Germany
Phone: +49 89 3299 218
Fax: +49 893299 2191
Email: sth@uts.ipp-garching.mpg.de

Weinberger, Howard
Technology Applications
P O Box 40190
Albuquerque NM 87196
USA
Phone: 505 266-6740
Fax: 505 266-6931
Email: weinberger@plk.af.mil

Wenes, Geert
Cray Research, Inc.
Applications
5 Post Oak Park, Suite 2020
Houston TX 77027
USA
Phone: 703 297-7896
Fax: 703 968-1620

Wenger, Doug
United States Navy
Fleet Numerical Oceanography
Center
Airport Road
Monterey CA 93943-5005
USA
Phone: 408 656-4445
Fax: 408 656-4489
Email: dwenger@fnoc.navy.mil

West, Sam
Cray Research, Inc.
Customer Service-West Box
454028
4505 S. Maryland Parkway
Las Vegas NV 89154-4028
USA
Phone: 702 895-4499
Fax: 702 895-4156

Whiting, Don
Cray Research, Inc.
890 Industrial Blvd
Chippewa Falls WI 54729
USA
Phone: 715 726-5014
Fax: 715 726-6713

Wicks, Thomas
Boeing Computer Service
Research and Technology MS 7L-
48
P.O. Box 24346
Seattle WA 98124-0346
USA
Phone: 206 866-3953
Fax: 206 865-2965

Wielinga, Paul
Sara
Kruislaan 415
Amsterdam 1098 SJ
The Netherlands
Phone: 31-20-592 3000
Fax: 31-20-668 3167
Email: wielinga@sara.nl

Wienke, Bruce
Los Alamos National Laboratory
Advanced Computing Laboratory
P.O. Box 1663, CDO/ACL MS
B287
Los Alamos NM 87545
USA
Phone: 505 667-1358
Fax: 505 665-4939
Email: brw@lanl.gov

Wilkins-Diehr, Nancy
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Williams, Mark
United States Army Research
Center
SECAD, SLCBR-SE-A
Bldg. 394, Room 231
Aberdeen Proving Ground MD
21005-5067
USA
Phone: 410 278-6320
Fax: 301 278-5077
Email: mrwil@arl.army.mil

Williams, Winnie
Cray Research, Inc.
Software Development - T3D
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Wilson, John
Cray Research, Inc.
Software Development
655-F Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Wimberly, Frank
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh PA 15213
USA
Phone: 412 268-4960
Fax: 412 268-5832
Email: wimberly@psc.edu

Windoffer, Cris
Jet Propulsion Laboratory
4800 Oak Grove Drive
MS 301-455
Pasadena CA 91109
USA
Phone: 818 354-7357
Fax: 818 393-1187
Email: cris@voyager.jpl.nasa.gov

Winget, Karen
Fine Point Editorial Services
1011 Ridge Valley Court
Shepherdstown WV 25443
USA
Phone: 304 876-1618
Fax: 304 876-1814
Email: kwinget@mcimail.com

Wohlever, Kevin
Martin Marietta Services Group
135 Washington Avenue
Bay City MI 48708
USA
Phone: 517 894-7685
Fax: 517 894-7600
Email: kwy@nesc.epa.gov

Wojewocki, Diane
San Diego Supercomputer Center
P.O. Box 85608
San Diego CA 92186
USA
Phone: 619 534-5000
Fax: 619 534-5152

Wood, Robert
Lawrence Livermore National
Laboratory
Computation Organization/LC
p.o. Box 808
Livermore CA 94550
USA
Phone: 510 423-5077
Email: bwood@llnl.gov

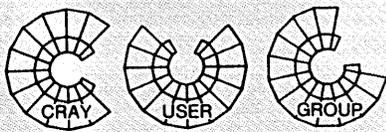
Worlton, Jack
Cray Research, Inc.
Corporate Marketing
2231 E. 3980 South
Salt Lake City UT 84124
USA
Phone: 801 272-0494
Fax: 801 272-0495

Yoo, Sang Seop
Samsung Advanced Institute of
Technology
Department of Supercomputer
Applications
P.O. Box 111
Suwon 440-600
Korea
Phone: 82 2 744 0011 x 9160
Fax: 82 331 280 9158

Young, Bing
Lawrence Livermore National
Laboratory
LC MS L-061
P.O. Box 808
Livermore CA 94551
USA
Phone: 510 423-5077
Email: young@lanl.gov

Zais, Jeff
Cray Research, Inc.
Applications
655-E Lone Oak Drive
Eagan MN 55121
USA
Phone: 612 683-3522
Fax: 612 683-3599

Zumach, Bill
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh PA 15213
USA
Phone: 412 268-4960
Fax: 412 268-5832
Email: zumach@psc.edu



INCORPORATED

Cover Photo:
James Blank/San Diego Convention & Visitors Bureau