Desktop Batch Processing

Jim Gray and Chris Nyberg

San Francisco Systems Center
Digital Equipment Corporation
455 Market Street, San Francisco, CA 94015
{Gray, Nyberg} @ SFbay.enet.dec.com

Abstract: *Today, online transaction processing applications can downsize from mainframes to microprocessors. Commodity database systems, operating systems, and hardware came of age in 1993., -- they surpassed the online transaction processing performance of proprietary solutions.*

There are lingering doubts about downsizing batch transaction processing applications. The doubts center on the ability of microprocessor hardware to handle the high IO bandwidth required by batch processing, and on doubts that microprocessor systems offer the software services and utilities key to batch processing applications.

This paper reviews the impressive progress of made by commodity software and hardware in processing OLTP workloads. The discussion is quantitative because the Transaction Processing Performance Council defined a set of benchmarks that characterize OLTP and that quantify price and performance.

Discussion then turns to batch transaction processing. There is less consensus on the characteristics of batch transaction processing. Consequently, much of the discussion focuses on requirements. The discussion ends with some performance measurements of utilities running on DEC Alpha AXP microprocessors and on commodity disks. These results indicate that microprocessors today have the capacity to process batch workloads at mainframe speeds. We predict that over the next few years, batch-processing software, exploiting parallel processing will emerge. This, combined with commodity hardware will provide both superior performance and price performance.

1. Client/Server Economics

Downsizing and rightsizing are driven by economics: in particular the economy of scale. There are 100,000,000 microprocessors in use while there are at most 50,000 mainframes in use. This creates a diseconomy of scale. The fixed engineering costs associated with mainframes must be amortized across a few thousand units. These costs, in excess of a billion dollars, drive unit costs into the millions. The benefits of mainframes do not justify these huge fixed costs.

C. Gordon Bell observes that there are seven computers classes ranked by price [1]:

Type	Example	Population
Less than 10$:	Wristwatch	10^9
Less than 100$:	Pocket calculator	10^8
Less than 1,000$:	PC/Notebook/cellular	10^8
Less than 10,000$:	Workstation	10^7
Less than 100,000$:	Departmental server	10^6
Less than 1,000,000$:	Mainframe server	10^4
Less than 10,000,000$:	Supercomputer	10^2

The small populations (right-hand column) have large fixed costs spread over a few units. These fixed costs make "big" machines disproportionately expensive. To make these arguments concrete, consider the following prices and volumes.

	Micro	Mainframe	Ratio
$/SPECint	100$/SPECint	10,000$/SPECint	100:1
$/RAM megabyte	50$/MB	1,000$/MB	20:1
$/Disk Gigabyte	500$/GB	5,000$/GB	10:1

The high mainframe prices reflect multi-billion dollar engineering costs amortized across a few thousand units.

Similar arguments apply to software. Bill Joy observed that one should not write software for a platform with less than 100,000 licenses because the economics are terrible: The engineering cost is spread across only a few units and so is prohibitive. When Joy formulated this rule, commodity meant 100,000 units. Today, commodity means one million or ten million units. Today *one should not write software for a platform with less than a million or ten million units*.

To make the argument concrete, consider the database systems used for OLTP. IBM's DB2 database system costs over 100,000$ as an initial license fee for the complete product. There are about 10,000 such systems. Microsoft Access costs about 100$ and has several million licenses. Both systems generate 300M$ in annual revenue and both can sustain a comparable engineering organization. Digital's Rdb database system has about 100,000 licenses and averages about 30,000$/license, giving it a comparable business. Several other database vendors are operating in

this range. Oracle is able to generate 1.5B$ annual revenue through a combination of higher volumes and premium prices.

The message here is clear: the high-volume producers have low unit-costs. This will eventually drive the market to a few producers in each area. This is happening for microprocessors, disks, printers, displays, and operating systems. It is also happening to layered software -- graphical user interfaces, programming environments, class libraries, and database systems. It is happening to generic applications like word processors, spreadsheets, mail, workflow, general ledger, inventory control, MRP, etc. Each of these will have to sell millions of units to be competitive.

The platform numbers are:

Platform	Units
DOS	75,000,000
MS/Windows	25,000,000
Mac	5,000,000
X/Windows (= UNIX)	2,000,000
Presentation Manager (=Mainframes)	50,000

These numbers are dynamic, Windows is quickly penetrating the DOS base. Microsoft's NT operating system has been out for only three months, but already outsells UNIX 3:1.

There is an important distinction between client and server software. Client software can sustain unit prices of about 100$ while server software can sustain unit prices of about 1,000$ -- about 100$/client. Hundred-dollar products can afford at most 10$ of engineering expense and 25$ of marketing and support expense. Since product engineering routinely costs in excess of a million dollars, client products must sell 100,000 units per year to be viable. For thousand-dollar server products the engineering expense can be 100$ per unit and the company need only sell 10,000 per year to be viable. If, as is typical for many software products, the engineering expense is in the tens of millions of dollars, then the company must sell a million clients or hundreds of thousands of servers per year to be viable.

These simple economics ignore the support and marketing issues -- large suppliers can spread their marketing and support costs among more units and so have much better visibility. Oracle and Microsoft have huge marketing and support organizations. These economies of scale, and the benefits of standardizing on a single code base make it difficult for others to compete on product features and engineering excellence.

This can be summarize by Mike Stonebraker's view that we are going from class 5 software to class 3 software. Stonebraker classifies software by the number of trailing zeros in the price: a 100$ package is class 2 and a million dollar package is class 6. The database server business is currently driven by class 5 products -- both in the UNIX, VMS, and MVS space. Recent price cuts and the packaging of Sybase with NT and Oracle with NetWare have already moved us to a class 4 price point. Stonebraker predicts we will be at the Class 3 price point by 1995.

2. The Commoditization of OLTP

For most of the 1980's, the mainframe vendors and want-to-be mainframe vendors had a goal to deliver SQL-based transaction processing systems able to process 1,000 transactions per second -- 1Ktps. One side effect of this effort was consensus on the definition of a transaction per second. In 1988, essentially all the DB and TP vendors formed a consortium called the Transaction Processing Performance Council's (TPC). The TPC's goal was to reduce the bench-marketing hype and smoke by defining a level playing field on which all vendors could compete and be measured. In 1989, these efforts bore their first fruit with the TPC-A benchmark [2]. TPC-A defined metrics for performance (tps) and price/performance ($/tps). TPC-A was followed with a more realistic OLTP TPC-C benchmark. The TPC is now defining decision support, client/server, and mainframe benchmarks.

From 1989 to 1992, the performance and price-performance metrics showed that proprietary systems had the best peak performance and best price performance. For a while Digital's VAX and Tandem's Cyclone/CLX had the best peak performance and price performance. HP's best performance was registered by its Allbase product. IBM's AS/400 line also had impressive performance and price performance -- substantially better than its RS/6000-AIX offering. Significantly, IBM's DB2 mainframe system never published results. Certainly, DB2 had excellent performance (estimated in the hundreds of transactions per second), but it ran on expensive mainframes. We conjecture that IBM did not want to quantify the diseconomy of its mainframes by publishing TPC-A results for them. The only mainframe vendor to publish results, Unisys, came in at about 45k$/tps. At the time, this was twice the average price of its competitors.

Between 1989 and 1993, the commodity operating systems (SCO UNIX, NetWare, NT), the commodity databases (Oracle, Informix, Sybase, Ingres), and the commodity transaction monitors (Tuxedo, VIS/TP, Encina) dramatically improved their performance on simple transactions.

In 1993, UNIX, Oracle, and Tuxedo became the price-performance leaders. Oracle, Tuxedo, and Sequent's Dynix operating system running on Intel 486 processors were the first to break the 1ktps barrier that had stood for over a decade. Using six Digital Alpha AXP processors on VMS, both Rdb and Oracle broke the 1ktps barrier with slightly

better price performance. The peak performance and price per transaction continue to improve rapidly. Currently Compaq-SCO/UNIX-Oracle is the price performance leader. Digital, HP and Sun have higher-performance but higher-priced solutions. As of January 1994, the leaders in each performance band are [3]:

Performance band	Leader	$/tps
under 250 tps-A	Compaq/Oracle	5k
under 1000 tps-A	Sun/Oracle	6k
over 1000 tps-A	Digital/Oracle	7k

A few years ago you could fault Compaq for having machines with no parity on the memory or processor, relatively unreliable discs, and no OLTP software. Their machines were simply not competitors. Today, the story is completely changed. Compaq is the world's largest supplier of RAID5 disk arrays. The "enterprise" versions of their products have extensive built-in diagnostics, remote maintenance, integral UPS, and limited ability for one node to fail-over to another. The SCO-UNIX offering, combined with Tuxedo and Oracle or Informix is well respected in the industry. The NetWare and NT offerings from Novell-Oracle and Microsoft-Sybase are also getting good reviews from users.

These commodity systems do not cluster at present. *Clustering* allows a pool of processors provide service to clients. Clusters provide a uniform programming and management interface to the resource pools of the cluster. Clustering is needed for scale up to really large configurations containing dozens of disks and thousands of clients. It is also needed for high availability. In clusters other devices quickly switch in to provide access to a replica of the server or data when a device or processor fails.

Today, robust clustering technology is restricted to Tandem's Guardian operating system, Teradata's DBC/1024, and to Digital's VMS operating system. However, every vendor offers an early version of their clustering on UNIX and NT. We expect that this cluster software to take a few years to mature, but there is no question that it will be robust by 1996.

In addition, the new software is substantially easier to use. For example NT/Sybase provides a uniform naming and security domain, a graphical interface to administration and operations, and modern development tools. SQL stored procedures, application generators like PowerBuilder, SQLwindows, Windows 4GL, and others make it relatively easy to build TP-lite client-server applications supporting as many as a hundred users per server. Scaling to larger user communities, requires partitioning the task into multiple smaller servers or using conventional transaction processing monitors like Tuxedo, Encina, ACMSxp, or CICS. Software to automate this client-server split is offered by tools like Ellipse and Forte.

So, times have changed. The OLTP business has been commoditized. Downsizing from mainframe solutions to commodity technology is in full swing. Commodity software has set new price points.

3. The Next Step: Commodity Batch Processing

Most users agree with what has been said so far. For them the only question is how to move and how quickly to move from the mainframe. In these discussions, there is one recurring theme: *what about batch?* Users believe they can now move their online transaction processing workload to a microprocessor. The TPC-A results demonstrate that the performance is there and that the software is there. But, what about their batch workload?

Many users assume that their batch workload cannot move off the mainframe to small servers. They point to hardware and software limitations. In our view, concerns about hardware are outdated -- modern commodity systems have impressive performance and reliability. As explained below, there *are* valid concerns about the absence of batch processing software on commodity computers. Much of this software is being ported from mainframes to micros, and should be robust in a few years.

We discuss the hardware issue first, and then the software issues.

3.1. Hardware Is Not The Problem

The Teradata DBC/1024 should dispel the notion that microprocessors cannot drive large disk farms. Some of the largest mainframes are just front-ends for a Teradata cluster of a few hundred Intel 486 processors driving a few thousand SCSI disks. Many large retailers use such multi-terabyte disk farms to track their sales activity and to optimize their inventory. These systems provide excellent performance on data-intensive decision support workloads.

Today, the performance of the commodity Intel and RISC processors is close to the performance of the fastest mainframes. RISC clock rates are faster (300MZ), and the overall performance on standard benchmarks are comparable.

Consider the disk IO issue. In the PC space, systems were hampered by compatibility with the PC-AT bus which limited IO traffic to a few megabytes a second -- less than the speed of a single modern disk. Today, with the MicroChannel at 50MB/s and the PCI bus at 200MB/s, Intel-based and DEC-Alpha-based servers can deliver

100MB/s from the disk to the application. This has been demonstrated for both NetWare and for VMS.

Disc architectures available for Intel and DEC-Alpha systems have excellent performance. Compaq is the largest supplier of RAID5 disk arrays. Small Fast-Wide-Differential SCSI disks are delivering 7MB/s today, and arrays of these discs have been measured at over 60MB/s. Modern SCSI discs are as reliable as their mainframe brethren, but are about 2x faster and about 10x less expensive. These disks have large and sophisticated caching mechanisms built into the drive controller. These caches make it relatively easy to read and write the disc at device speed.

3.2. PC and UNIX File Systems are Improving

On the software side, UNIX and MS/DOS file systems were not designed for high-performance disk IO. The more modern systems, NetWare and NT, do not suffer these limitations. UNIX offers raw disk interfaces, and competition from NT is finally forcing the UNIX purists to offer asynchronous and unbuffered (no extra copies) IO.

The original structure of the UNIX file system prevented high speed sequential IO -- UNIX tends to map the data to disc as a tree of small blocks, rather than using an extent-based file system. Traditional UNIX file systems do small writes, tend to copy the data at least twice (as it moves through the buffer pool), and UNIX traditionally performs all IO operations as synchronous requests. In addition, UNIX tends to generate many spurious IOs to maintain the file system directory.

The UNIX directory IO problem has been "solved" by using non-volatile RAM (Prestoserve), or by exploiting transaction processing logging techniques to track directory updates., or by using a log-structured file system.

More aggressive designs have compromised pure UNIX semantics by providing a "traditional" file system modeled after IBM's OS/360. These file systems, using extent-based file allocation, have no extra data moves, and provide an asynchronous IO interface. Cray and Sequent give good examples of this UNIX adaptation.

To satisfy the needs of IO intensive applications, almost all UNIX systems provide a raw disk interface. The system manager can designate zones of disc to be dedicated to an application (these zones look like files). The application can then do direct Get_Block() and Put_Block() reads and writes to these files. This interface has low overhead. Most database systems and high-performance applications use these raw-disk interfaces rather than the "cooked" file systems.

In addition, a variety of disk striping and disc mirroring packages are appearing as integral parts of UNIX file systems.

The NT file system is modeled on the VMS file system. It includes an extent-based file system, direct and asynchronous IO, disk mirroring, and disk striping. All indications are that it provides excellent IO performance.

To summarize the IO story. Traditionally, the DOS, NetWare, NT, and UNIX systems were hampered by low-performance hardware IO subsystems. Modern commodity cpu, bus, and disc subsystems have very impressive IO performance --- typically in excess of 50MB/s.

3.3. The Software Barrier

Traditional and modern batch processing systems depend on software seldom found on commodity systems. Each online transaction generates data that is processed by an end-of-day, end-of-week, or end-of-month batch transaction. These batch transactions perform operations such as account reconciliation, transaction settlement, monthly invoicing, billing, task and equipment scheduling, materials resource planning, or reporting.

Some argue, as we have, that corporations should be re-engineered to have no batch steps: all batch operations should be done online as mini-batch jobs during normal processing. For example, billing and invoicing could be done on the fly. Each transaction could update the corresponding bill. Then, the billing cycle would consist of the mini-batch job of printing or EDI-posting the bill to the customer. Similarly, when a new order arrived, the order entry job could initiate a mini-batch job to schedule the manufacture and delivery of that order. Any changes to work schedules and inventory planning would be part of this task.

It may well be that we are right, mini-batch is the correct way to engineer new applications. But, the fact is that no large corporation works this way today. Banks, stock exchanges, manufacturers, distributors, retailers, governments, and all other large organizations have large batch workloads. These workloads run on mainframes today.

What will it take to downsize these batch workloads to commodity systems? We believe the following software services are prerequisite to a batch downsizing effort:

Tools (COBOL, RPG, MRP, SQL, Sort, ...): Downsizing to a commodity platform is easier if the new computer system supports the traditional programming languages and tools. The good news is that most of the UNIX and

PC systems support GUI interfaces to high function versions of the traditional tools.

Job Control and Scheduling: Batch jobs are described in a job control language. A simple workflow language that says do this step, then this one. The language has simple procedure steps, simple flow control and some access to the process, job, and system environmental variables. IBM's JCL and REX languages, and UNIX's shell language are typical.

The job control program can be executed interactively, by invoking it from a command shell. More commonly, the job placed in a batch execution queue and is executed by a job scheduler. The scheduler advances the job through its various steps. If a job fails due to external events, the scheduler restarts it. The scheduler performs load control by dispatching at most one job from each queue at a time. Users want to track and account for jobs rather than job steps --- the scheduler accumulates the job step costs and generate accounting reports. Today, the job schedulers on UNIX, NetWare, and NT are rather primitive, but they will evolve quickly as companies downsize to such platforms.

Spool and Print Services: Batch jobs typically generate reports, bills, payrolls, statements and task lists. These high-volume print jobs often use special forms. The print spooler manages a pool of high speed printers and their forms, typically driving them off forms-oriented queues. If the printer runs out of forms or if some are damaged, the job resumes where it left off -- rather than reprinting the entire output. VMS, UNIX, NetWare, and NT each have spoolers with these capabilities.

Tape Handling and a Tape Librarian: Magnetic tape is used for archiving old data, for shelving infrequently used data, and for making a backup copy of data in case of disaster. Large shops accumulate tens of thousands of tapes. Magnetic tapes are the standard form of high-speed data interchange among corporations. Online networking is increasingly common, but even today, many organizations support only tape interchange.

It is rare to find good software to read ANSI labeled tapes (no kidding) -- the mainframes have it but the minis and micros have been slow to implement these standards. The tape library systems common to mainframes are rare on commodity operating systems. The promising news is that third party tape-handling systems are being ported from the mainframes to UNIX, VMS, and NT.

System Managed Storage (SMS) is sometimes called Hierarchical Storage Management (HSM)**.** Batch jobs tend to read old files and create new ones (old-master new-master). Over time the number of files can become extraordinarily large. Rather than have a human being manage file archival and retrieval, customers expect the computer system to migrate old files to tape and retrieve them on demand. SMS took a long time to implement on the mainframe. Third party systems are being ported to commodity platforms as part of the downsizing movement.

Generation data sets: Batch programs often take an old-master new-master approach to their data. They expect the underlying operating system to manage versions of data, discarding very old versions. Few commodity operating systems have a versioning scheme built into the system (UNIX, NetWare, and NT do not, VMS does). This means that the application must manage file versions.

Exotic Reader and Printer Support: Some batch jobs have special IO requirements like magnetic check readers, optical, mark-sense readers, microfiche printers (COM), or high-speed laser printers. These devices traditionally had exotic interfaces suited to IBM 370, UNIVAC, Burroughs, or NCR mainframes. Increasingly they have Ethernet or FDDI interfaces -- but still, interfacing such paper-handling or film-handling devices to commodity systems is a difficult task.

Checkpoint-Restart: Batch transaction processing involves a few big jobs. The traditional logging-undo-redo protocols of OLTP are inappropriate for batch transactions. Rather, batch processing uses an old-master new-master scheme for coarse grained recovery, and a checkpoint-restart scheme for fine grained recovery. Checkpoint-restart interacts with job scheduling (the failed job is rescheduled at the restart point), printing (the spool output is resumed at the checkpoint), tape handling (the tapes are repositioned to the checkpoint) and so on. Checkpoint/Restart is an integral part of a batch-processing operating system. It is difficult to retrofit. The NetWare, UNIX, and NT operating systems are unlikely to provide a restart service -- rather applications will have to rely on the coarser old-master new-master recovery combined with application-level (user designed) checkpoint-restart.

High Speed Sequential IO: Batch programs read bulk data, reorganize i, consolidate it and then produce a variant of the data. These operations are designed to be sequential so that they exploit the high-speed sequential performance of disks and tapes. As argued earlier, commodity systems used to have poor sequential performance -- both due to hardware and software limitations. But today, the non-UNIX systems have good performance, and most UNIX systems are

beginning to provide high-performance asynchronous unbuffered IO.

Dusty Deck: Virtually every mainframe shop has some programs that no one understands. The programs were written long ago and just work. Often, the source for the program (the deck of punched cards) has been lost and there was never any documentation. There is no easy way to migrate dusty decks. They just have to be executed in emulation mode on the new platform or re-implemented.

In summary, customer concern about downsizing batch jobs to commodity platforms is legitimate. The hardware is capable, but some key software components are missing. Many software components are already present, and most other components will appear in the next year or two. The work we are doing in our laboratory is part of that trend.

4. The AlphaSort experience.

Our laboratory is focused on using parallelism to quickly process very large databases. We assume that future high-end severs will be built from hundreds of processors and thousands of disks. The challenge is to fined ways to program such an array of processors and disks [4].

In dealing with large databases, almost all operations are batch operations. Loading the data, organizing it, reorganizing it, and searching it all involve billions of records. Such problems require clusters of processors and farms of disks. Today we are using VMS clusters built on DEC Alpha AXP processors and disks. Our techniques are portable to other operating systems and processors.

We built a high-speed parallel sort utility as part of our effort. It uses parallel IO from conventional SCSI disks, and it uses shared-memory multiprocessors when possible.

Using a single DEC Alpha AXP 7000 processor (200Mhz), we were able to read an array of 36 "slow" SCSI disks at 64MB/s using only 20% processor utilization. These were "old" RZ26 drives capable of 2.3MB/s each -- today 1-year old is "old". By using modern RZ28 drives capable of 4MB/s, the processor could easily drive this array at 100MB/s. Next yea's disks will nearly twice as fast.

Using the "slow" array with a new memory-intensive algorithm called AlphaSort, we were able to sort records very quickly. On the Datamation Sort benchmark [5], AlphaSort sorted a million records in 7 seconds and sorted 10 million records (1 gigabyte) in less than a minute [6]. These times are four times faster than a Cray YMP, and eight times faster than an 32-processor-32-disk Intel Hypercube. They also beats the best IBM DFsort and SyncSort times by a wide margin. In fairness, neither DFsort nor SyncSort are able to use file striping since it is not part of MVS operating system services.

In addition to this sorting work, we are building a parallel execution environment the quickly load large databases. Our current goal is to load and index a terabyte database in a day.

Another Digital laboratory reports backing up and restoring Rdb databases at rates in excess of 30 GB/hour using DEC Alpha AXP processors -- that is substantially faster then the best reported mainframe backup/restore speeds. By partitioning the database and by using parallelism, one can multiply these rates to hundreds of gigabytes per hour. This software exists on VMS today, and is being ported to OSF/1 and NT.
These results convince us that Alpha AXP systems have excellent hardware and software support for high-speed IO. Batch oriented utilities are present on VMS today and are coming to UNIX and NT soon.

5. Summary

Commodity hardware and software can deliver online transaction processing today. The products are mature and deliver superior performance and price performance.

Commodity hardware can support the large databases and demanding workloads required for batch transaction processing. Unfortunately, the corresponding batch transaction processing software is either immature or non-existent.

The good news is that many third-party batch software infrastructure is being ported to commodity platforms as part of the downsizing trend.

6. References

[1] Bell, C.G., J.E. MacNamara, *High Tech Ventures*, Addison Wesley, Reading, MA., 1981.

[2] "TPC Benchmark A", Chapter 2 of *The Benchmark Handbook for Database and Transaction Processing Systems*, 2nd ed. Morgan Kaufmann, San Mateo, CA 1992.

[3] TPC Quarterly Report, Issue 8, Shanley Public Relations, San Jose, CA., 15 Jan. 1993.

[4] DeWitt, D. W., Gray, J.N., "Parallel Database Systems: the Future of High Performance Database Systems", CACM, **35**(6), June 1992.

[5] Anon Et Al., "A Measure of Transaction Processing Power." *Datamation*. **31**(7): 112-118., 1 April 1998.

[6] Nyberg, C.T., Barclay, T.D., Cvetanovic, Z.Z., Gray, J.N., Lomet, D.L., "AlphaSort - A RISC-machine Sort", Submitted to SIGMOD 94.