Dynamic address translation equipment is key to the design of System/370 central processing units, and dynamic relocation is key to the design of Operating System/Virtual Storage 1.

Discussed are the significance and implementation of these key facilities in the supervisor and job scheduler functions of virtual storage operating system.

Within the supervisor are presented system initiation, page management, input/output supervisor, and storage management. Within the job scheduler are discussed queue management, the job entry subsystems, and remote job entry services.

OS/VS1 concepts and philosophies

by T. F. Wheeler, Jr.

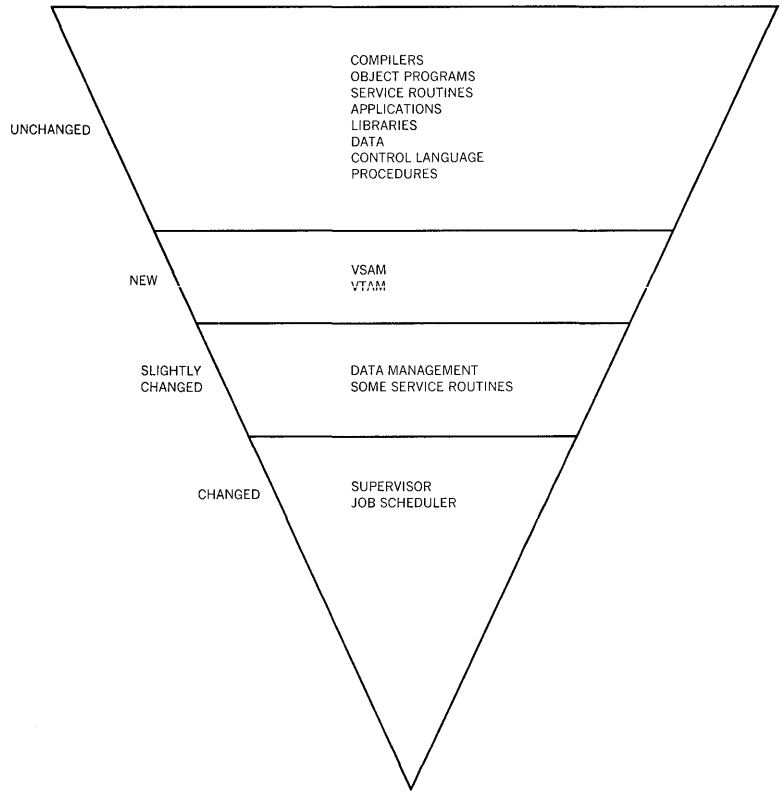
During the past two years, a number of users have been introduced to the dynamic relocation function in the IBM Operating System/Virtual Storage 1 (OS/VS1). Therefore, it is appropriate to investigate in this paper many of the design concepts that have proved fundamental to the structure of the OS/VS1 operating system.[1]

In a significant earlier resolution of storage management problems, the designers of the Ferrante-Atlas computer incorporated dynamic address translation and a mechanism for expanding fixed storage capacity.[2,3] Additional innovations in storage technology throughout the industry made possible variations and extensions of system design. The introduction of multiple address spaces[4,5] and segmentation[6,7] also advanced the state of the art.

With this background available to them, the designers of OS/VS1 used virtual storage concepts to significantly extend the capabilities of the Operating System/Multiprogramming with a Fixed Number of Tasks (OS/MFT). In addition to the extended capabilities themselves, a salient philosophy of the design and structuring of the system was to cause the user of OS/VS1 the least possible disruption in their use of these capabilities. This objective has been achieved by stabilizing system interfaces and by reducing the extent of the changed areas as shown in Figure 1.

The use of the enhanced capabilities requires that System/370 central processing units have a facility for Dynamic Address

Figure 1 OS/VS1 viewed as modifications of OS/MFT



Translation (DAT). The DAT is a hardware device that automatically makes address adjustments that permit all references to storage to be made to the virtual range of storage. In effect, the DAT provides a mapping between a virtual address and the current physical storage location independently of and transparent to the operating system. The DAT hardware passes control to the operating system when the data or instructions addressed are on secondarystorage and not in real storage.

Reflecting a significant change to storage management, the job scheduling mchanisms have been modified to use a technique of dynamic relocation.[8] The concept of job management in OS/MFT has been broadened to become resource management in OS/VS1. Perhaps the single most important addition is the incorporation into the system of the Job Entry Subsystem 1 (JES1) with Remote Entry Services, which simplify the control paths through the system.[9]

Changes have been made to portions of the OS/MFT data management routines to implement the I/O capability in the virtual partitions. Also portions of data routines that handle input and

output (SYSIN and SYSOUT) have been modified to ease job entry interface transparency.

New data and terminal access methods have also been introduced into the system to make use of the dynamic relocation capabilities. Because both the Virtual Storage Access Method (VSAM)[10] and the Virtual Telecommunications Access Method (VTAM) require special treatment, they have not been included in this article.

Supervisor

Virtual storage provides an expansion of address space, thereby making the total address space appear to be larger than that of real storage. In OS/VS1, the total address space can be as great as 16,777,216 bytes, which contain the control program, data, and normal application jobs within partitions. Virtual storage addresses are not mapped directly to real storage addresses, but both are broken down into 2,048-byte sections called *pages*[11] in virtual storage which, in turn, are stored in *page frames* in real storage. A collection of pages is called a *segment*, and, in OS/VS1, segments are 64K bytes in length.

A study was made to determine the optimum page size for an OS/VS1 environment to be used with a range of real storage sizes greater than 160K bytes. Involved in the study was a determination of the effective CPU time for instructions and data within a page, measurements of interpage and intrapage reference activities, and the time required to move a page from real storage to secondary storage. The 2K-byte page size was found to be an optimum balance among these considerations.

Similarly, the Direct Access Storage Device (DASD) mapping algorithm was considered to be a critical parameter for achieving both the extended capabilities and performance in the midrange of System/370 computer systems. The direct one-to-one mapping of virtual storage space into secondary storage space was found to greatly simplify the movement of data from real to secondary storage and to reduce the logic size of the paging input/output routines. The DASD paging space has its origin at the upper boundary of the resident supervisor nucleus, and the virtual space has its origin at zero.

In the following sections we discuss the implementations of algorithms in the principle supervisor nucleus subcomponents that are resident in main storage.

system initialization

The *Nucleus Initialization Program* (NIP) performs many of the basic housekeeping requirements for the OS/VS1 system, which

include the calculation of initial values for the paging tables (such as the real storage page tables) and the completion of resident parameter lists. Information for NIP processing comes largely from system generation options and from variations of those options that are entered by the operator.

Early in the design of the system, it was decided to make more effective use of parameter entries during the execution of the NIP. This has made it possible to assign addresses for many of dynamic relocation and JES1 related options after the Initial Program Loading (IPL). Thus it is also possible for modifications to be made to the tailored system during the early stages of initialization. Also, the automation of the initialization program reduces the role of the operator, thereby speeding up the initialization process.

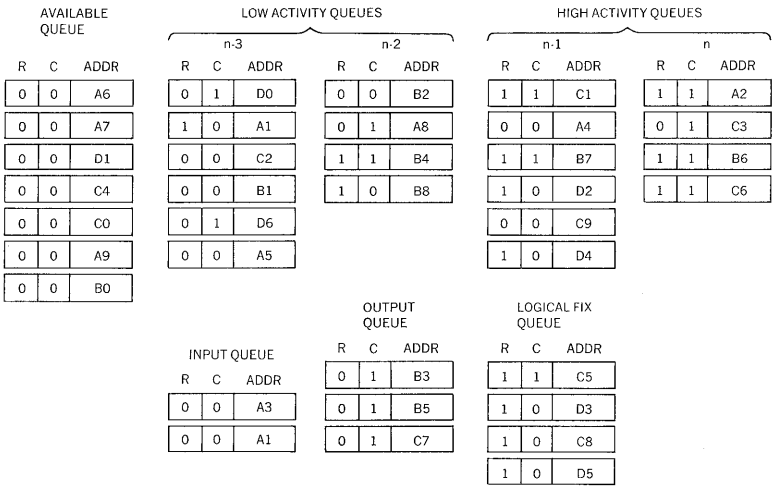
page management

Computer configurations that use OS/VS1 as their operating system are termed *demand paging systems* in the sense that new pages are read into real storage when the DAT detects that a referenced item is not present in real storage. Page management is the key component in the management of storage in a demand paging system. The page management component is accessed directly by the System/370 hardware when a *page exception* (or *page fault*) occurs. A page exception occurs when the Dynamic Address Translation (DAT) feature is unable to resolve a virtual address to a real storage location. At this point of the page exception procedure, page management assumes responsibility for further storage management operations. A key objective of a paging system is to reduce page exceptions (faults) to a minimum level, and simultaneously to optimize the use of real storage.

OS/VS1 uses an algorithm[12] that maintains a list of page frames that are available for replacement by a demanded page. To do this, the system uses several pointer queues to manage a least-recently-used page replacement algorithm. The pointer queues also regulate the flow of pages to and from external page storage. Storage management of this kind implicitly requires the capability of predicting the hit ratio[13] of the system, i.e., the probability of finding the referenced frame. We summarize the capabilities of the following four pointer queues that are involved in carrying out the page replacement algorithm.

In-use queues contain the addresses of currently active page frames.[14] These queues are arranged in the order of ascending activity of frame references to the frames located in each queue. The number of in-use queues is a variable that depends on the number of active partitions and active tasks, including system tasks. Included among the in-use queues are the following types:

Figure 2 Page supervisor queues



available (for replacement), activity (ranged from low to high), input, output, and logical fix. Figure 2 shows these in-use queues, with the queue containing the highest activity frames in the system at a given time.

Available page frame queue contains addresses of frames that are available for program replacement when a page exception occurs. During execution of the NIP, all Real Storage Page Table Entries (RSPTEs) that represent real storage block addresses that are greater than the fixed supervisor nucleus are entered into this queue. As the NIP is executed, the Available Page Queue is maintained at a threshold that is just sufficient to minimize the conditions of no replacements possible (*lockout*) and excessive replacements (*thrashing*).[15]

Page input/output device queues contain the addresses of frames that are being used for page I/O. The input queue represents the list of frame addresses that are currently being filled from external page storage (SYS1.PAGE). This process has been called *pulling* (or "creation"). The output contains the addresses of the least referenced pages that are about to be stored on external page storage (SYS1.PAGE), and has been called *pushing* (or "annhilation"). The management of I/O page frames in a timely fashion reduces the backing storage delay referred to by Joseph.[3]

Logical fix queues contain the addresses of both short-term fixed page frames and long-term fixed page frames, and derive from the fact that some components cannot tolerate a page exception. Therefore some pages are fixed or locked in real storage with a duration based upon the content of the pages and the time requirements of the system. With the availability of Release 3 of

OS/VS1, the user can force some of his application pages on this queue by means of the applicable macroinstructions.[16]

Fundamental to page frame management are the *change bit* and the *reference bit* in the page and segment mapping tables. Both bits are originally set by hardware, and they are reset in the process of paging by the page management routines. The change bit indicates whether the contents of a given page frame have been modified since the page was brought into real storage. This bit is reset only when the page is moved to the external page file. The reference bit is turned on when reference is made to the contents of a page frame, and it remains on until the bit is reset as a result of a new page measurement process.

At periodic intervals, based on task switches or a set time value, the status of the in-use queue frames is adjusted. This process involves the migration of all unreferenced page frames to the next lower queue and all referenced frames to the highest level queue. Page frame migration enables the low reference level frames to move to the lowest level queue and eventually enables their replacement. OS/VS1 uses a single stepwise downward bubbling mechanism to the lowest level of the queue, and a direct-jump upward mechanism to the highest level queue for all referenced pages. Because this mechanism keeps the referenced frame on the queue for a longer period of time, the critical effects of input/output operations on paging performance are reduced.

When a referenced page is not contained in real storage, the DAT hardware facility turns control over to page management. Page management immedicately attempts to free the necessary frames from the available queue. A request to page management is frequently for a number of frames. If an adequate number of free frames is available, the request is immediately satisfied. If there are not enough frames to satisfy the request and to maintain an adequate threshold, the page replacement routine is entered. The page-frame-release request formula is applied as follows:

$$R = A + H - C$$

where

R is the release request amount

A is the page allocation request

H is the high threshold on available page queue (The formula uses the high threshold to reduce the number of entries into the page replacement algorithm.)

and

C is the available page frame count

This calculation provides the number of additional page frames to be released to maintain the available queue at an acceptable level. The page replacement routine interrogates the low usage queues to determine the frames that may be freed. Page frames with their reference bits turned off can be released to the available queue. If a change bit is turned on, the frame must first be moved to the output queue, where it is placed on the external page storage. Unchanged frames are moved directly to the available queue.

Refering again to Figure 2 and Reference 14, an entry in the page measurement routine moves all frame addresses to the next lower level queue, where $n - 2$ is the lowest possible level. All frames that have been referenced are moved to or remain on reference level n in the queue structure. (This includes frames A1, B4, B8, C1, B7, D2, D4, A2, B6, C6). The reference bits are reset to 0 on all frame indicators on the n queue, so as to maintain the stepwise downward bubbling movement. The change bit is not modified at this time, nor is the reference bit pattern on the logical fix queue altered.

Since the lowest activity queue contains the Least Recently Used (LRU) frames, the process of page release concentrates on the low-activity queues by moving in a right-hand scan in the example, from the lowest to the highest queue. Once again, Figure 2 shows that frames D0, C2, B1, D6, A5 are currently available on queue $n - 3$ and are thus available for release if required. If we establish the low threshold as 3 and the high threshold as 6 for the available queue, any request that causes the number of available frames to fall within the threshold range causes the *page release routines* to be entered. The frames on the output queue are moved directly to the available queue when an output completion is returned, and the change bit is reset to 0.

task deactivation

A detailed study of the hit ratio function during the development of OS/VS1 has enabled the developers to build a number of mechanisms into the system to resolve potential performance problems. Of these mechanisms, task deactivation is the most dramatic, and, therefore, it should be discussed. The page release routines normally scan only the low activity queues. If an inadequate number of frames can be obtained from the low activity queues, then the *task deactivation routine* is entered. Partitions are then deactivated one at a time to make their frames available for additional page requests. Since partition pages are scattered throughout real storage, task deactivation frees up frames throughout real storage. Partitions are deactivated from low to high order of priority,[16] as defined by the installation system programmer.

Deactivation controls excessive paging rates known as *thrashing*, a vexing problem in paging systems that is caused by hypercon-

tention for available real storage. The end result of thrashing is a very high page I/O rate. Thrashing often occurs when a program runs with reduced system capability. Deactivation reduces contention by reducing the number of active tasks when a task threshold level is detected. Severe contention is thus eliminated and performance is maintained at an adequate level for a minimally reduced number of tasks. As a guide to understanding paging behavior, Denning discusses a three-way relationship among program behavior, paging algorithms, and system hardware configurations.[15]

task reactivation

The opposite performance problem is that of an insufficient number of active partitions. In this case, *task reactivation* routines must be entered in time to permit a properly balanced range of CPU loading. Periodic checks are made to determine the availability of resources for task reactivation and to maintain a proper CPU load balance.

These checks are based on analyses of available pages versus minimum required page partition activation. Deactivated partitions are reactivated in order of decreasing priority (with highest priority first) when a task switch occurs. OS/VS1 has expanded the facility by which a user installation monitors and controls some of the deactivation parameters that enable an operator to force the activation of particular partitions. PAGETUNE is the command that allows a system programmer to control certain values used in the paging algorithm, including the following categories:

- Alteration or suspension of the deactivation function.
- Alteration or suspension of page measurement functions.
- Alteration of the timing and paging criteria used by task deactivation.
- Display alterable values.

Excluded from deactivation are the following categories:

- System functions necessary for continuous execution, since their deactivation stops the system.
- Jobs executing in the virtual-equals-real mode, since the required real storage is defined as not available to paging.
- The last active user job, since this is the object of system execution.
- Pages currently in a locked condition.

In summary, the page management routines play a vital role in the achievement of an installation's goals. The adequate allotment of storage to these routines is therefore extremely important.[17]

input/output supervisor

Automatic address translation is not performed on channel command word addresses of the System/370 channels. Since the ad-

dresses are virtual addresses, they must be converted to real addresses before program execution can take place. In OS/VS1, the input/output supervisor performs the additional address translation. Moreover, certain information must be fixed in real storage to avoid page contention during an operation.

Thus, in the normal execution of an I/O request, the I/O supervisor first fixes the frames that contain fields for tables, buffers, and work areas. Since short term fixing is part of the I/O execution, the I/O data fields need not be totally contained within main storage, but are brought into main storage by the interaction between page management and the I/O supervisor. When this information is fixed in storage, the real addresses are placed in the appropriate locations in the channel control word. The START I/O instruction is issued to a single channel command word or to a chain that contains real addresses that are located in protected system storage. Upon the completion of the I/O operation, the fixed frames are unfixed and returned to the normal processing queues. Since the address of the real channel control word chain is different from the virtual address (that has been built by Data Management), self-modifying channel command words do not execute in a virtual partition.

virtual = real

To provide a capability to run real-time or self-modifying channel programs, OS/VS1 provides a mode of operation known as *virtual = real* (V = R). In this mode, the address space that is assigned to a job step is placed in contiguous real locations below the user-designated V = R line.[18] For a given program, the size of the V = R area is specified on the REGION parameter of the Job Control Language, and it represents the actual size of the program to be executed. Since the V = R area must be totally contiguous, the job step execution waits for an unused contiguous space to be freed. As soon as, the job execution is initiated, the address space is not available for paging, and the real job is not deactivated.

Although the DAT feature is in use during the execution of the V = R job, the address translation is the identity translation for the CPU program. Channel programs are not translated.

storage management

The virtual = real address space permits the execution of highly time-dependent programs and self-modifying programs. In addition, certain high I/O activity job steps may be run in V = R mode to avoid channel command word translation. It is apparent, however, that the effect of V = R on real storage may be to adversely affect other areas of the system.

The storage management algorithms of OS/MFT have been modified to use virtual address space, by having a page supervisor

assume the role of real storage management. It was recognized early in the design of OS/VS1 that the relatively large addressing capability of virtual storage could be used to make many of the options of OS/MFT resident in virtual storage. Figure 3 shows a number supervisor options that have been made resident through the paging capability. Conversely, many formerly main-storage resident routines can now be loaded into pageable system modules, thereby reducing contention on critical real storage and still permitting ready access to the routines. As a result, portions of the control program as well as many critical control blocks have been placed in protected portions of virtual storage, thereby making OS/VS1 more secure than its predecessors.

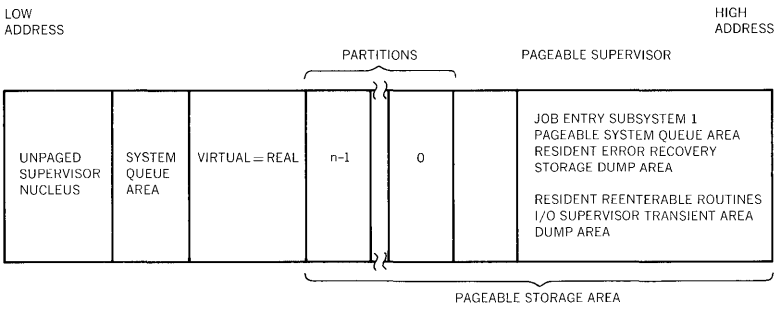
protection

OS/VS1 has a protection system that is based on keys that provide security of one partition from another. The keys are transparent to the user and are maintained by the storage management portion of the supervisor. The addition of the Authorized Program Facility (APF) further enhances the system security capabilities by controlling access to the system and to user functions. A number of services thus fall under the protection of installation management on a job step level. Similarly, the System Queue Space (SQS), which had become a critical resource in OS/MFT, has been broken into the following four portions that depend on the area of required information:

- *Fixed System Queue Area* (SQA) is a permanently fixed table space (adjacent to fixed resident supervisor space) and is used for the execution of system functions. The size of SQA is initially established during system generation, but it can be extended or contracted, depending upon its level of usage. Examples of SQA usage include translation areas for real channel command words and tables oriented to task execution, such as, enqueue and dequeue (ENQ/DEQ) control blocks.
- *Pageable system queue area* is an area used by system tasks for their pageable storage requirements.
- *Fixed Partition Queue Area* (FPQA) is a permanently fixed area, generally of 2K bytes or less, that is used primarily for partition page tables and for other tables that are used for partition management. Such tables cannot be paged for reasons of reliability and integrity.
- *Pageable Partition Queue Area* (PPQA) is a protected portion of each partition that contains tables that are used for partition management, and its paging has little impact on system performance.

There is a main storage resident nucleus of the OS/VS1 supervisor that is brought permanently into real storage during execution of the Initial Program Load (IPL) to perform the normal control program functions. Strict control should be exercised over the generation of such a nucleus in small systems so that an adequate

Figure 3 OS/VS1 storage map



amount of real storage remains for the paging process. Caution should be used to avoid the generation of unnecessary resident options.

A maximum of fifteen user partitions may be defined in OS/VS1 with each partition in 64K byte increments of virtual storage. In addition, up to thirty-seven system task partitions may be defined. Priority is determined by the partition in which each task resides, wherein partition priority is entered into the system by the CLASS parameter on the JOB card. Normally, system support modules, such as data management, are located in the user's partition. A user may define a resident re-entrant load module in a pageable resident re-entrant routine area for space and performance considerations, and he may similarly use the partition definition as a means of controlling performance.

We have discussed the major areas of change made necessary by the dynamic job relocation function. Dynamic task dispatching has also changed the system task dispatching techniques, so as to prevent a CPU dominant task from overriding I/O task dispatching. The calculation of dynamic dispatching priorities has aided performance in some cases by using a time slice algorithm to classify and order tasks.

The OS/VS1 scheduler, as the earliest implementation in the OS/VS1 relocation environment, has been especially packaged in certain portions to optimize the programming of virtual storage. The value of reducing the number of branches and of clustering the subroutines in the job scheduler has a demonstrable effect on optimization.

Job scheduler

We now investigate the major areas of change in the job scheduler. Based on the need to support expanded supervisor functions and the ability to take advantage of dynamic relocation, a

number of design decisions have changed much of the OS/MFT job scheduler. These decisions include those to repackage modules and to make major algorithmic changes within the job scheduler framework. Many of the changes improve user accessiblity to the system and remove performance bottlenecks. Other decisions, such as that of rewriting the job initiator, are intended to provide performance improvements. Individual module repackaging attacks local performance problems; I/O load balancing provides performance improvement to specified areas. The end result is a faster and more usable job scheduler, a strong base for the total system.

The support scope of the dynamic relocation function could have limited the scheduler changes to control card modifications and some internal changes in the Program Status Word and Set System Mask areas. It was recognized, however, that additional benefits could derive from tailoring the OS/VS1 job scheduler to make use of dynamic relocation. The OS/MFT job scheduler uses the two basic options of small partition scheduling and normal job scheduling to provide program scheduling. Investigation demonstrated that performance and maintainability improvements would result from changing the scheduler to execute in a 64K byte virtual partition. Dynamic relocation thus alleviated the need for small partition scheduling, and, as a result, jobs can be scheduled into available partitions of their requested class.

Portions of the OS/MFT job scheduler, such as termination routines, were in part repackaged and in part reprogrammed to better support dynamic relocation. This was done by moving high-usage subroutines in line so as to avoid excessive paging activity. In addition, a regrouping of tables that is based on reference rate and location of reference has further reduced paging activity.

The initiation portion of the OS/VS1 job scheduler has also been rewritten to provide faster and consolidated job initiation. Moreover, the initiator has been tied to the interpreter instead of the reader as a part of the Job Entry Subsystem 1 work.[19]

These changes do not affect the basic execution order of the job scheduler. However, other enhancements have modified the functional structure of the OS/VS1 job scheduler, although the outward interface has been maintained.

central queue manager

We now consider some of these major enhancements to the OS/VS1 job scheduler in detail. An early analysis of job queue usage indicated the need for a redefinition of the contents and structure of the central queue manager. The OS/MFT Job Queue Data Set (SYS1.SYSJOBQE) contains various forms of job control information in addition to the actual job queue. Access to this queue is

spread through a number of in-line routines to 176-byte chained records. As a better reflection of virtual storage OS/VS1 has retained the 176-byte queue records, but has broken the job queue into a number of specialized data sets that include the following.

Job Queue Data Set (SYS1.SYSJOBQE) retains the name of the OS/MFT data set, but it is much smaller in size. The SYS1.SYSJOBQE format is specified when the system is generated and is altered during system start if desired. Relevant information created by the reader, interpreter, and initiator is stored in this data set. Disk entry records and accounting records are placed on the data set according to class and priority. When jobs terminate, an entry is made for SYSOUT information, according to class. The jobe queue information is deleted, following the processing of last SYSOUT record. The Job Queue Data set is dynamically extendable in OS/VS1 Release 3.

Scheduler Work Area Data Set (SWADS) is created when an initiator is started on a partition basis. A SWADS contains the scheduler work tables that are created and maintained throughout the scheduling routines. Since the SWADS are allocated on a partition basis, the file can be accessed in parallel by each partition. In OS/VS1 Release 3, SWA can reside in virtual storage.

Spooling Data Set (SYS1.SYSPOOL) contains the Job Control Information, commands, and input data from the JES1 input reader. On the output side, the Spool Data Set contains output and messages related to each job execution.

pageable job entry subsystem (JES1)

We now discuss the relationship of the spooling data set to the Job Entry Subsystem. It is apparent that the division of queue information into a number of parts has reduced contention problems. The incorporation of an embedded spooling capability into OS/VS1 is one of the broadest functional changes to the OS/VS1 job scheduler. The Job Entry Sybsystem 1 (JES1) incorporates a high-speed spooling mechanism into a pageable centralized routine for scheduler usage. JES1 is so structured that apart from a page that is fixed in real storage for a long time after the Nucleus Initialization Program, JES1 is pageable, all or in part, depending on its frequency of use.

The normal introduction of a job stream to OS/VS1 is through the JES1 input reader. The Job Entry Peripheral Services (reader and writer) handles all the system input (SYSIN) and output (SYSOUT). The JES1 reader is designed to read job control statements and data, which are passed immediately to the appropriate data sets owned by the Spool Manager. This action changes the sequence of interpretation so that the Job Control Language (JCL) interpreter in OS/VS1 runs as a subroutine of the initiator. Delayed interpretation can be prevented by entering a new parameter

TYPRUN = SCAN on the job card. In this case, a diagnostic error scan is performed as the job passes through the reader. When the errors have been corrected, the job must be resubmitted.

A minimum of input scanning is done by the JES1 reader before the input is submitted to the Job Entry Central Service routines. An internal job identifier is assigned at this time, which is a combination of the user job name plus a unique system number. The central service routines separate the input from the JCL and write both to the SYS1.SYSPOOL data set. Similarly, in-line procedures and entries from the procedure library (SYS1.PROCLIB) are placed in a special procedure SPOOL area.

Division into separate areas enables the JES1 routines to minimize disk-access contention and improves system performance. JES1 maintains an information directory to allow rapid retrieval from all areas of the spool file.

The JES1 reader for card devices does not terminate at the end of file as in OS/MFT. This facility has become known as a *hot reader* facility because it reduces operator intervention and reads data into the computer more quickly. Another advantage of the JES1 readers and writers is the single re-entrant copy that is maintained in the pageable system area to assure user access to all partitions.

The JES1 output writers reside in the JES1 portion of the pageable supervisor. They write the output data sets created by problem programs as well as the messages created during the initiation/termination tasks. The user may specify the number of writers, each of which may hold up to eight classes of data. In addition to the output of JES1 writers, the user may decide to send his output directly to output devices by means of the direct system output writer. Unlike the JES1 writers, the direct system output writers reside in a problem-program partition.

To provide greater flexibility of use, the JES1 options and default values (parameters) are stored in the JESPARM member of the parameter library (SYS1·PARMLIB). JES1 options and default values may be modified during the Initial Program Load (IPL) process. This capability is used for modifying the number and size of the JES1 buffers, and it greatly reduces the need for new system generation.

remote entry services

An important extension to the job entry environment is that of Remote Entry Services (RES), which provide a logical terminal extension of job entry by using the Remote Terminal Access Method (RTAM) to drive batch terminal devices. RES permits jobs to be routed from remote work stations, and permits the output to be returned to the same or different work stations.

Since RES is a true extension to JES1, RES commands are identical to those used by the local JES1 support. Additional optional data sets have been added that permit the proper authorization and storage of broadcast information. RTAM is the only portion of RES that is not present unless specifically generated.

Since the RES design is totally integrated into the JES1 structure, RES is treated as a logical extension of the JES1 reader and writer, and communicates to the system in the same manner as the reader and writer.

Concluding remarks

One of the major concerns of users of virtual storage systems has also been a major objective of system designers, that is, ease of moving programs from OS/MFT systems to OS/VS1 systems. Discussed in this article have been supervisor and job scheduling operations that make such a move possible, whereby programs remain wholly or largely intact. The major innovation of virtual storage systems is that of dynamic relocation. Effects of dynamic job relocation on the supervisor and the job scheduler have been discussed.

Some functional aspects of the OS/VS1 system generation (SYSGEN) process have also been related to the supervisor and job scheduling functions. SYSGEN in OS/VS1 is a simplified derivitive of that in OS/MFT.

As in the case of SYSGEN, the operator interface to OS/VS1 is substantially the same as that to OS/MFT.

Aspects of the OS/VS1 control program have been discussed. Many of the OS/MFT optional control program functions have been incorporated into the OS/VS1 control program portion of the pageable supervisor space, and have thus been removed from the option process.

Because of the extended directly addressable storage (virtual storage), the movement of programs to OS/VS1 systems is simplified. Virtual storage largely removes problems of storage management and the overlaying of application program substructures in moving from OS/MFT to OS/VS1. When designing and coding new applications, dynamic relocation provides system enhancements for improving performance in those cases.

CITED REFERENCES AND FOOTNOTES

1. S. J. Shields, "How one company went to VS," *Computer World*, January 23, 1974 and January 30, 1974.

2. J. Fotheringham, "Dynamic storage allocation in the Atlas computer, including an automatic use of a backing store," *Communications of the ACM* **4,** 10, 435–436 (November, 1961).
3. M. Joseph, "An analysis of paging and program behavior," *Computer Journal* **13,** 48–54 (February 1970).
4. R. J. Adair, B. V. Bayles, L. W. Comeau and R. J. Greasy, *A Virtual Machine System for the S/360/40*, TR 320–2007, IBM Scientific Center Report, May 1966. May be obtained from the IBM Scientific Center, 545 Technology Square, Cambridge, Massachusetts 02139.
5. G. E. Hoernes and L. Hellerman, "An experimental S/360/40 for time sharing," *Datamation* **14,** 39–42 (April 1968).
6. S. E. Gluck, "Impact of scratchpads in design: Multifunctional scratchpad memories in Burroughs B8500," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **27,** 661–666 (1965).
7. F. B. MacKenzie, "Automated secondary storage management," *Datamation* **11,** 24–28 (November 1965).
8. In the early design of OS/VS1, it was decided to treat the scheduler as the first user of dynamic relocation and to optimize critical routines accordingly. Some useful references to optimization are the following:
   - G. S. Shedler and S. C. Yang, "Simulation of a model of paging system performance," *IBM System Journal* **10,** 2, 113–128, (1971).
   - L. W. Comeau, "A study of user program optimization in a paging system," *ACM Symposium on Operating System Principles*, Gatlingburg, Tennessee (October 1967).
   - J. E. Morrison, "User performance in virtual storage system," *IBM Systems Journal* **12,** 3, 216–237 (1973).
   - D. R. Slutz and I. L. Traiger, *A Note on the Calculation of Average Working Set Size*, IBM Technical Report, RJ 1209, April 27, 1973, may be obtained from the IBM T. J. Watson Research Center, Yorktown Heights, New York 10598. (To be published in *Communications of the ACM.*)
9. Discussed in the article by J. G. Baily, J. A. Howard, and T. J. Szczygielski in this issue.
10. Discussed in the article by D. G. Keehn and J. O. Lacy in this issue.
11. OS/VS1 and DOS/VS use a 2.048-byte page and page frame size. TSS/360, OS/VS2, and VM/370 and a 4,096-byte page and page frame size.
12. L. A. Belady groups replacement algorithms into the following three classes in his article "A study of replacement algorithms for a virtual storage computer," *IBM Systems Journal* **5,** 2, 78–101 (1966):
    Class 1 for storage blocks that are equally referenced but with no algoithmic basis in storage usage; essentially a type of first-in-first out algorithm.
    Class 2 for storage blocks that are classed by their most recent history; status bits serve as prime movers.
    Class 3 for storage blocks that are classed by an intensive history of pages.
13. The hit ratio function applies to the optimal behavior of a page referencing pattern. A high hit ratio indicates a high availability of the referenced frames.
14. T. F. Wheeler, "IBM OS/VS1–An Evolutionary Growth System" *Proceedings of the National Computer Conference*, 395–400 (1973).
15. P. J. Denning, "Thrashing: its causes and prevention," *AFIPS Conference Proceedings, Fall Joint Computer Conference* 33, Part 1, 915–922, (1968).
16. PGFIX fixes virtual storage contents in page frames.
    PGFREE frees page frames.
    PGLOAD loads virtual storage into page frames.
    PGRLSE releases page frame contents.
    For a more complete description see *IBM Systems/370, OS/VS1 Planning and Use. Guide VS1 Release 3*, Form GC24-5090-2, IBM Corporation, Data Processing Division, White Plains, New York 10604.
17. In an attempt to identify performance characteristics, Morrison (in Reference 8) identifies a number of interrelationships between working set and parachor curves. Morrison defines a *parachor curve* as a graph of the total

number of page exceptions that a program causes to access when paging against itself in a fixed amount of real storage, versus the amount of storage available to it for execution.

18. The virtual = real line defines the upper limit of address space available for contiguous allocation in blocks. The system default for V = R is the smaller of either 512K bytes or the real main storage size of the computer mode. The actual amount of storage available for the virtual = real mode varies with the options chosen for the system.
19. Job Entry Subsystem is based to a large degree on technology that has evolved from the Houston Automatic Spooling Package system (HASP), which was developed for OS/MFT and OS/MVT. JES1 incorporated many of the mechanisms of HASP directly.