IBM OS/VS1—An evolutionary growth system

by T. F. WHEELER, JR.

International Business Machines Corporation
Endicott, New York

INTRODUCTION

A brief study of IBM OS/VS1 (Operating System/Virtual Storage 1) will reveal a system providing many faceted growth capabilities at all levels of user-system interaction. Additional meaningful function is provided on a stabilized base to assure this growth capability. It can be further seen that installation growth is achieved through new application work and not by a continual rework of existing programs. To assure the users ability to move to new work almost immediately, OS/VS1 is built on an IBM OS/MFT (Operating System/Multiprogramming with a Fixed Number of Tasks) base. Compatibility is defined to extend to most object programs, source programs, data and libraries from OS/MFT to OS/VS1, thus assuring a normal movement of existing programs to the virtual environment. Figure 1 graphically represents the areas of change between MFT and VS1.

In like manner, the transitional problem of education is greatly reduced for the programmer and operator alike. VS1 uses the MFT languages in all areas of programmer/operator contact to the system, and from the system generation procedure to the operator control language, VS1 incorporates, improves and extends the existing MFT language to support the virtual function.

As an OS compatible system VS1 becomes a vital part of the IBM family of new virtual systems which includes DOS/VS (Disk Operating System/Virtual Storage), OS/VS1, OS/VS2 and VM/370 (Virtual Machine/370). Each is based upon its predecessor system but each expands the horizon of support with virtual memory.

The virtual storage facility is the single most important new characteristic of VS1. It offers significantly longer address space for both application partitions and system functions by providing, with adequate equipment, a 16 million-byte addressing capability.

To provide this enhanced capability, OS/VS1 requires a System/370 central processing unit with the dynamic address translation facility. VS1 supports this facility on the System/370 Models 135, 145, 158, 168, and those 155's and 165's which have the DAT facility field installed. In addition to the hardware facility, significant changes were made to control programs code which I shall discuss later in this paper.

Significant enhancement was made to the job scheduling algorithms. The single most important addition has been the incorporation of Job Entry Subsystem and Remote Entry Services into the Release 2 scheduler. These functions provide efficient job entry from both local and remote users, providing a transparency of operation that enhances remote capabilities. I will also investigate these changes in detail at a later point in the paper.

Finally, VS1 will contain a new data management function—Virtual Storage Access Method (VSAM). This function and its new data set organization has been added, as an ISAM (Index Sequential Access Method) replacement, to better support more sophisticated and online applications. Significant improvements in the exploitation of relocate, data integrity and recovery, device independent addressing, and migration ability help to make VSAM an important base for data base development. Since VSAM is a topic in itself, it will not be discussed in this paper.

RELOCATE

General discussion

Virtual storage separates address space and real storage and then expands the address space to make it larger than real storage. In VS1, address space can be up to 16,777,216 bytes containing the control program, data, and normal application jobs within partitions. Virtual storage addresses are not related to real storage addresses, but both are broken into 2048-byte sections called in virtual storage, pages, and in real storage, page frames. A great deal of study went into determining the optimal page size for a VS1 environment. Involved in this study was a determination of the effective CPU time for instructions and data within a page and the time taken to move the page to secondary storage from real storage. The page size of 2K balances these considerations for optimal performance.

In like manner, the DASD (Direct Access Storage Device) mapping algorithm was considered critical in achieving both medium entry and performance in that entry level. The direct mapping of virtual to secondary space greatly simplifies the movement of data from real to secondary storage and reduces the logic size of the page input/output routines.

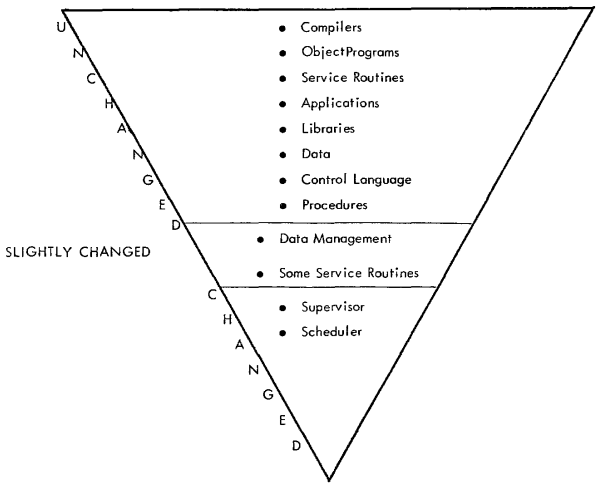

Figure 1—Areas of change between OS/MFT and OS/VS1

Page management

The key component in the management of virtual storage is page measurement. Page measurement is accessed directly by the System/370 hardware when a page exception occurs. A page exception occurs when the address translation feature is unable to resolve a virtual address to a real storage location. At the point of the exception, page management assumes responsibility for ensuring the addressability of the initial storage contents.

OS/VS1 uses a number of pointer queues to manage its least recently used page replacement algorithm and regulate the flow of pages to and from the external page storage. Some of these queues include:

1. In-Use Queues—The addresses in these queues point to locations of currently active page frames. These frames contain the most recently executed code and the most recently used data and tables. The number of in-use queues is a variable dependent upon the number of active partitions and active tasks including system tasks. Figure 2 shows four such in-use queues.
2. Available Page Queues—This queue contains the frames that are available for program servicing when a page fault occurs. At the initial program load, all RSPTE's (real storage page table entries) representing real storage blocks above the fixed nucleus appear on this queue. As execution occurs, this queue is maintained at a minimum threshold to minimize both lockout and thrashing possibilities.
3. Page Input/Output Device Queues—These queues are 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). The output queue contains the addresses of the least referenced pages that are about to be stored on external page storage (SYS1.PAGE).
4. Logical Fix Queue—This queue contains the addresses of both short-term fixed page frames and long-term page frames.

Keys to page frame arrangement are the change and reference bits. Both bits are set by hardware and reset in the process of paging activity 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.

At periodic intervals (between 3 and 9 task switches in Release 1), the status of the in-use queues page frames is adjusted. This process involves the migration of all unreferenced frames to the next lower queue and all referenced frames to the highest level queue. This migration enables the low reference level frames to move to the lowest level queue and eventually permit their replacement.

As we have noted before, when a referenced page is not contained in real storage, the hardware facility turns control over to page management. Page management immediately looks to the available queue to satisfy the request. If an adequate number of frames is available, the request is immediately satisfied. If there is an inadequate number to satisfy the request, the page replacement routine is entered. The page-frame release request formula is applied as follows:

$$A + HTV - APC = \text{Release Request Amount}$$

where:
A = Page Allocation Request
HTV = High Threshold on Available Page Queue
APC = Available Page Frame Count

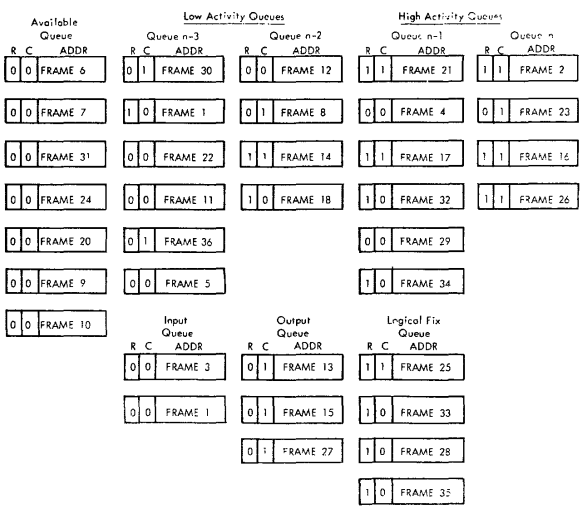

Figure 2—Four in-use queues

This calculation will indicate how many additional page frames should be released to maintain the available queue at an acceptable level. The page replacement routine will begin a scan of the low usage queues to determine what frames may be freed. Page frames that have the reference bit 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. Frames that have not been changed are moved directly to the available queue.

Let us again refer to Figure 2. An entry to the page measurement routine would move all frame addresses to the next lower level queue where N-3 is the lowest possible level. In like manner, all frames that have been referenced are moved to reference level N in the queue structure. (This includes frames 1, 14, 18, 21, 17, 32, 34, 2, 16, 26.) The reference bits are reset on all frame indicators on the N-queue. The change bit is not modified at this time nor is the reference bit pattern on the logical fix queue.

Similarly, the process of page release concentrates on the low activity queues moving in a right hand scan from the lowest to the highest queue. Once again referencing Figure 2, frames 30, 22, 11, 36, 5 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 at 6 for the available queue, any request for five or more pages would force the release routines to be entered. The frames on the output queue will be moved directly to the available queue when the interrupt is returned.

The page release routines will scan only the low activity queues. If an inadequate number of frames can be obtained from the low level queues, then partition deactivation is entered. Deactivation will deactivate a partition at a time to make available their frames for additional page requests. Partitions are deactivated from low order of priority to high order of priority.

Deactivation controls excessive paging rates known as thrashing. Thrashing, a classical problem in paging systems, is caused by a hyper-contention for available real storage. Deactivation reduces the contention by reducing the number of active tasks. Consequently, the severe contention is eliminated and performance is maintained at an adequate level.

The opposite performance problem is an insufficient number of active partitions. Reactivation must be entered in adequate time to permit a properly balanced range of CPU time. Periodic checks are made to determine the availability of resources for reactivation. Deactivated partitions are reactivated in order of highest priority when a task switch occurs. Release 2 of VS1 has expanded the facility for a user installation to monitor and control some of the deactivation parameters.

Excluded from the deactivation category are:

1. System functions that are necessary for continual execution.
2. Certain system tasks.
3. Jobs that are executing in virtual = real mode.
4. The last active user job.

In summary, the page management routines play a vital role in maintaining system performance at an acceptable level. Key in the achievement of these goals is the allotment of an adequate level of resources to the paging routines.

Input/output supervisor

Channels on the System/370 do not perform address translation on channel control word addresses. Since these are virtual addresses, they must be converted to real addresses for proper program execution; the input/output supervisor of VS1 must do the additional translation. In addition, certain information must be fixed in real storage to avoid a page exception in the middle of an I/O operation.

In the normal execution of an I/O request, therefore, the I/O Supervisor must first fix the frames that contain tables, buffers, and work areas. Once this information is fixed in storage, the real addresses will be placed in the appropriate locations in the channel control word. The Start I/O is issued to a chain that contains real addresses, and is thus referred to as a real channel program. Upon the return of the I/O interrupt, the frames are freed and returned to the normal processing queue. Programs that create self-modifying channel control word chains cannot be handled by the normal I/O Supervisor routines and should be run in a virtual = real mode.

OS/VS1 provides a function known as virtual = real mode. The address space assigned to the job step is placed in a contiguous real location below the V = R line. The size of the V = R area is specified on the REGION parameter and represents the actual size of the program to be executed. Since the V = R area must be contiguous, the job step execution must wait for a free contiguous space to be freed. In addition, the V = R job step is not deactivated during the entire job step execution.

Although the DAT feature is in use during the execution of the V = R job, no translation takes place. In like manner, software translation of the channel control word is avoided.

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

Storage management

The advent of relocate served to modify the storage management algorithms of the operating system. Portions of the control program that were either optional or resi-

dent in the nucleus could be repackaged into pageable system modules, thus reducing the contention on critical real storage.

Similarly, the System Queue Space which became a critical resource in OS/MFT was broken into three positions dependent upon the area of required information.

1. The System Queue Area (SQA) was designated as a permanently fixed area in real storage that could be dynamically extended or contracted dependent upon its usage. The SQA is used for channel control word translation and for relevant system-oriented tables.
2. The Fixed Portion Queue Area (FPQA) is a permanently fixed area used primarily for partition page tables.
3. The Pageable Partition Queue Area (PPQA) is a protected portion of each partition that contains partition-relevant tables.

The fixed nucleus in VS1 contains the normal control program functions and is permanently in real storage. Strict control should be exercised over the nucleus in small systems to provide an adequate amount of real storage for the paging process.

User partitions must be defined in VS1 in 64K increments of virtual storage. The user may define up to fifteen user partitions and fifty-one system task partitions. Normally the supporting system modules such as data management would be located in the users partition. A user may define resident access methods in the pageable resident access method area for space and performance considerations.

We have discussed the major areas of change made necessary by the relocate function. Dynamic dispatching affected a change in the dispatching techniques of the system in Release 2. Additional changes were made to portions of Data Management and the Scheduler to reflect the hardware and supervisor changes. The Scheduler became the first user of relocate and, as such, repackaged certain portions to reduce branches and move subroutines in line. We will next investigate the major areas of change in Job Scheduling.

JOB SCHEDULING

A number of fundamental design decisions changed much of the VS1 Scheduler. These decisions ranged from the simple packaging choice for modules to the complex inclusion of spooling algorithms within the scheduler framework. Many of the changes were intended to improve user accessibility to the system, while others removed potential bottlenecks to improve performance. Some, such as I/O Load Balancing, are intended to perform total performance improvement as I/O utilization is spread intelligently across channels. The end result was a faster, cleaner component that provides a growth step to the seventies.

Basically the support scope of the relocate function could have limited the scheduler changes to those control card modifications and some internal changes in the Program Status Word and Set System Mask areas. It was recognized, however, that additional benefit could be derived from tailoring the scheduler to make use of relocate. The original base scheduler in MFT used two basic options to schedule jobs. The first was intended for any program with a partition in excess of 44K bytes. The second option was intended for small partition scheduling executed in a linear fashion in the 2K transient area. Investigation demonstrated that a performance and maintainability gain would result by changing the Scheduler to always execute in a 64K virtual partition. The need for a small partition scheduling algorithm is eliminated since the Scheduler can execute in a minimum number of real page frames.

In like manner, portions of the Scheduler such as termination routines were in part repackaged and in part recoded to better support relocate. This was done by moving high incident subroutines in line to avoid excessive paging activity. In addition, a regrouping of tables based upon reference rate and location of reference reduced the paging activity.

These changes did not affect the basic execution order for the Scheduler; however, other enhancements modified the functional structure of the component while maintaining the outward interface.

We will now look in detail at some of these major enhancements to the VS/1 Scheduler.

Central queue manager

An early analysis of the Job Queue usage indicated a need for a redefinition of the contents and structure. The MFT Job Queue Data Set (SYS1.SYSJOBQE) contained various forms of job control information including the actual job queue. Access to this queue was spread through a number of in-line routines to the 176-byte chained records. VS1 has broken the job queue into a number of specialized data sets to reduce the bottlenecks of the old structure. These data sets include:

1. Job Queue (SYS1.SYSJOBQE) retains the name of the MFT data set but it is only a fraction of the size. Relevant information to job queuing is stored on 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 job queue information is deleted following the processing of the last SYSOUT record.
2. Scheduler Work Area Data Set (SWADS)—this data set 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.

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

We will discuss in the following part of this paper the relevance of the spooling data set. It is apparent that the dichotomy of the queue information into a number of parts has reduced contention problems.

Job entry subsystem (JES)

One of the broadest functional changes to the VS1 Scheduler was the incorporation of the Job Entry Subsystem. JES incorporates a high-speed spooling mechanism into a pageable centralized routine for Scheduler usage. JES is so structured that apart from a minimum resident response routine, it is pageable in all or in part depending upon the frequency of usage.

The first external introduction to JES is through the input reader. The Job Entry Peripheral Services (reader and writer) handles all the system input (SYSIN) and output (SYSOUT). The JES reader is designed to read and immediately pass the input data to the Spool Manager. This changes the sequence of interpretation so that Job Control Language (JCL) interpreter now runs as a subroutine of the Initiator. This delayed interpretation can be prevented by entering a new parameter: TYPRUN = SCAN on the job card. In this case, a simple error scan is performed and the job is flushed through the reader. Once the errors are corrected, the job must be resubmitted.

The input from the JES reader is submitted to the Job Entry Central Service routines. An internal job name has been assigned at this time which is a combination of the user job name plus a unique system number. The central service routines will separate the input from the JCL and write both to the SYS1.SYSPOOL data set. In like manner, in-line procedures and entries from the procedure library (SYS1.PROCLIB) are placed in a special procedure SPOOL area. The division into separate areas enables the JES routines to minimize disk-access contention and thus improve performance. JES maintains an information directory to allow rapid retrieval from all areas of the spool file.

The JES reader for card devices does not terminate at the end of file as in MFT. This facility has become known as a "hot" reader facility. Another advantage of the JES readers and writers is the single reentrant copy maintained in the pageable system area assuring user access to all partitions.

In order to provide greater flexibility of use, the JES parameters are stored in the parameter library (SYS1.PARMLIB) and may be modified during the Initial Program Load (IPL) process. This facility is useful in modifying the number and size of the JES buffers and greatly reduces the need for a new system generation.

Remote entry services

An important adjunct to the Job Entry environment is Remote Entry Services (RES). RES is a logical terminal extension of Job Entry using the Remote Terminal Access Method (RTAM) to drive the terminal devices. Just as JES is intended as an incorporation of HASP techniques into the center of the system, RES is based on HASP Remote Job Entry routines and in fact uses modified HASP work-station support.

RES supports Point-to-Point (leased and dial-up) and the Two-Wire and Four-Wire Half Duplex lines. As with HASP, RES supports multi-leaving which is synchronized, two-directional transmission between two processing units. RES, like most of JES, uses relocate facilities and is therefore pageable.

The RES design is totally integrated into the JES structure so that RES is treated as a logical extension of the JES reader and writer and thus communicates to the system in the same manner.

MIGRATION

One of the first and fundamental questions that arises about the virtual systems is the ease, or lack of ease, of moving current programs to the new system. As I indicated in the introduction to the paper, many of the existing capabilities of current Operating System programs were moved to the relocate counterpart totally or largely intact. We should look at some specific areas to get a better comprehension of the differences.

The first introduction to a new operating system is the vast body of reference information intended to introduce the different competence levels to the software. In the case of VS1, this begins with the IBM System/370 System Summary (GA22-7001) which is intended as a general introduction to the system providing an overview of new and expanded capabilities. In like manner, the OS/VS1 Planning and Use Guide (GC24-5090), A Guide to the IBM System/370 Model 145 (GC20-1734) and the OS/VS1 Features Supplement (GC20-1752) are intended to introduce the reader to a more specialized and in-depth treatment of relevant system components. The general mode of the documentation has been very favorably received since it embodies a strong technical content with an easy to use style. The documentation has been based on the existing operating system documentation but style changes have had a favorable effect on the readers.

The documentation is presented in specialized packages that attempt to match relevance of information with the specialized need to know. Thus an operator does not have to dig through as much generalized information to find the fundamentals of operation.

The system generation process (SYSGEN) has been simplified by the reduction of options. As I indicated earlier, many of the former optional control program func-

tions have been incorporated into the virtual control program portion of the pageable space and thus removed from the option process. In like manner, the use of JESPARM feature represents a considerable advantage when the specific JES parameters should be modified.

The operator interface to the operating system is largely identical to MFT. Some additional work is done during the Master Scheduler Initialization routines causing some additional messages and perhaps responses on the part of the operator. In like manner, dumps of virtual memory will initially cause the operator to speculate as to what is happening, but by and large the operators to the VS1 system have not found the system more difficult to operate or for that matter really any different from their current procedures.

The migration of the programmers to virtual memory is largely straightforward. The programmers are faced with a removal of concern for memory management and overlay in their application substructure. Once again the virtual dump is usually larger but comparable to a large degree to the MFT dumps. Initially, experience has shown that some time is spent in locating some of the familiar tables that have been separated in the protected sections of the system tables. But in the long run, the job seems simplified by the grouping of partition-relevant information in a single place—the partition.

The adjustment to VS1 has been very rapid. Many users have compared the ease of migration to VS1 with the ease of moving to a new release of MFT. In like manner there has been genuine surprise at the increase in functional ability and the increased volume of job throughput.