The design of IBM OS/VS2 release 2

by A. L. SCHERR

International Business Machines Corporation
Poughkeepsie, New York

INTRODUCTION

The purpose of this paper is to give some insight into the design of IBM OS/VS2, rather than cover individual features of the release. Included are the overall objectives for the design, some of the system's key architectural features, and how these relate to the environments that the system is intended to be used in. The major objective is to show how the design of the system fits together and to provide an insight into the rationale of the design.

OBJECTIVES

Release 2 represents a major revision of OS to provide a new base for future application areas. The key thrust is to provide a new SCP base with increased orientation toward DB/DC applications and the additional requirements placed on an operating system because of them. Another key goal of the system is to support multiple applications concurrently in a single complex. This complex may include multiple CPUs, loosely or tightly coupled. The system must dynamically adjust itself to the changing loads in the various environments that it supports, as well as provide increased security and greater insulation from errors.

Maintaining a high level of compatibility continues to be a major objective for VS2. Extending the system, adding function, and changing its internal structure, while at the same time considering compatibility, represented a significant challenge to the designers of Release 2. Over the last few years we have learned a lot about the needs of our users, and during this time, the state-of-the-art in software design has moved forward. The system has been reoriented to incorporate these things into the system.

USE OF VIRTUAL STORAGE

The incorporation of virtual storage into OS has allowed the system to support programs whose size is larger than available real main storage. There are operational advantages; however, significant additional benefits can be realized. Using virtual storage to provide for an extremely large address space, allows program structures to be simpler, intermediate data files to be eliminated, and real main storage to be used to hold data that in the past was resident on a direct access device. This latter use can result in a significant performance advantage which will be discussed later.

MULTIPLE ADDRESS SPACES

Perhaps the most obvious new feature of Release 2 is the support of multiple address spaces. Each job step, TSO user, and operator STARTed program in the system has a private address space that is 16 million bytes, less the space taken by the operating system. Figure 1 is a comparison between Release 1 and Release 2 storage maps. Both maps extend from 0 to 16 million bytes. Release 1 and MVT actually look alike, with the only difference being that MVT's address space is limited to the size of real storage.

The Release 1 map, shows two TSO regions with several users in each. Users A and B, for example, cannot be in execution at the same time because only one of these users can occupy the region at a time. The others are swapped out. The transition from Release 1 to Release 2 can be understood very simply by considering the Release 1 system with a single TSO region the size of the total available virtual storage. What has been done in Release 2 is to remove the multiprogramming restriction between the users of the TSO region. On the other hand, Release 2 does not allow two jobs to share the same address space. One of the first implications of this design is that it is no longer necessary for the operator to get storage maps printed at the console so that he can manage main storage.

To show the effect of multiple address spaces on certain control program functions, TCAM will be used as an example. In Release 1, terminal input is read through a channel directly into the TCAM region. There it undergoes some processing and is then moved to the user's region or to the TSO control region. In Release 2, the input is read into a common system area buffer at the top of the map, and from there is transmitted under TCAM's control to the user. To programs that have done inter-
region communication in previous systems, this new storage map represents a major difference.

In Release 2 V=R jobs no longer affect the virtual address space of V=V jobs. Since each job is assigned a 16 million byte address range, V=R jobs only affect the amount of real storage available. (See Figure 2).

STORAGE MAP

Figure 3 shows the storage map seen by a single job. This corresponds to an MVT or Release 1 region. At the top of the map in areas which are commonly addressable by all of the address spaces is the System Queue Area containing system control blocks, the pageable Link Pack Area, and the Common System Area for use in communicating between users. This area is used, for example, by TCAM and IMS for inter-region communication. At the bottom of the map is the Nucleus and that part of Link Pack Area which is to remain permanently in main storage.

The area in between is the private address space for each user. User requests for storage in all subpools are allocated from the bottom of this private address space. Requests for Local Supervisor Queue Area and the Scheduler Work Area storage are satisfied from the top.

COMPATIBILITY

Compatibility is a major objective in Release 2. Object code and load modules from MVT and VS2 Release 1, not dependent on the internal structure of the system, will run with Release 2. JCL compatibility is maintained, and the data sets and access methods of previous releases apply, as well as EXCP. SMF is compatible as well. However, it must be recognized that in moving from a non-virtual to a virtual environment, the usefulness of some of the measurements has changed; and in order to account completely for usage, it may be necessary to make some use of the new measurements that are provided. Internal interfaces are the area of greatest concern because, in some cases, such interfaces have been extensively used. Generally, our approach has been to evaluate every change of this type to see what the effect is on the user community as well as our program products. Several proposed changes were not made because of their potential impact; but, on the other hand, some change is
required to make progress, and thus we have had to con­
sider many difficult trade-offs.

The major differences that affect compatibility include
the system catalog, which is now a VSAM based data set
and requires conversion from the catalogs of existing sys­
tems. Forward and backward conversion utilities have
been provided, as well as compatibility interfaces allowing
the use of the original OS catalog macros. As men­
tioned earlier, the new storage map, will impact programs
that have done inter-region communications. Also, lOS
appendages run enabled in Release 2 and must use a new
synchronization mechanism. Therefore, there is likely to
be impact to user-written lOS appendages.

PARALLELISM

One of our major design goals in the system was to
provide for as much parallelism of operation as possible.
The reduction of software bottlenecks that prevented effi­
cient multiprogramming is the major technique that we
used. Listed are five of the main areas that we worked in.
Each of these areas will be discussed.

- Job Queue
- Allocation
- Catalog
- TSO Region
- MP65 Disable Lock

Experienced OS users will recognize these as areas with a
high potential for improvement.

JOB QUEUE

We have eliminated the Job Queue data set that OS has
used since its beginning. With HASP or ASP in an OS
system, there were really two job queues—the one kept by
the support system relating primarily to information
required to schedule jobs and the printing of output, and
the OS job queue which contains similar information as
well as information pertaining only to a job in execution.
One type of information is really for inter-region commu­
ication between various parts of the scheduling functions
of the system; the other, for intra-region communication
between the scheduling components and data manage­
ment in behalf of the executing job.

The inter-region information has now been placed
entirely in the job queue maintained by the job entry
subsystem, either JES2 or JES3. The intra-region informa­
tion has been segmented and placed into the individ­
ual job’s address space. In this way, the portions of the
original OS job queue having the highest usage are now in
each job’s private address space. The less frequently used
information relating to communication between various
components of the scheduling function is now in the JES
job queue. Thus, all of these elements of the job queue
can be accessed in parallel. The JES job queue is also
used to journal information required to restart jobs dur­ing
warmstart or from a checkpoint. (See Figure 4.)

ALLOCATION

The component of the system that does data set and
device allocation has been completely redesigned. Both
batch and dynamic allocation are now supported by the
same code and provide essentially the same function. The
design is oriented toward virtual storage—no overlays are
used, and all work areas are in virtual storage. Allocation
of data sets to storage or public volumes can be done
completely in parallel, regardless of other allocation
activity.

This type of allocation represents probably the most
common form in most installations, and, in general, the
design of the new allocation provides shorter paths for
these generally simpler cases. When it is necessary to
allocate a device and perform volume mounting, these
requests are serialized by device group. Therefore, a
request for a disk need not be held up because another
job is waiting for a card reader. Other improvements in
this area include the ability to prevent a job from holding
devices until its entire requirement can be met, and the
ability to cancel a job waiting for devices.

CATALOG

The catalog has been converted to an indexed VSAM
data set, primarily to allow for faster access to a large
catalog. The curves in Figure 5 give the general idea of
how access time should relate to catalog size with this new
structure.
TSO REGION

As previously stated, in MVT or Release 1, TSO users sharing the same region cannot be concurrently in execution. This restriction is eliminated in Release 2. Therefore, the situation shifts from one where each region serves a given set of users, to one where the entire system serves all of the users. Thus, any potential imbalance between regions is eliminated. (See Figure 6.)

Moreover, previous support placed a limit on the level of multiprogramming for TSO at the number of TSO regions. In Release 2, the level of multiprogramming can vary and is dependent upon the load placed on the system.

LOCKS

In a tightly-coupled multiprocessing system, it is highly desirable from a performance point of view to allow the control program to be executed simultaneously on both CPU's. However, some means is then required to synchronize or serialize the use of control information used by the control program.

System code in MVT disabled for interrupts prior to the updating or use of this type of control information; and when the operation was completed, the system was enabled. The MVT technique used for Model 65 multiprocessing was to use one lock which prevented both CPU's from being disabled at the same time. In environments with heavy usage of control program services, this lock becomes a significant performance bottleneck. (See Figure 7.)

In the Release 2 support of MP, we have used instead a number of specific locks, each relating to a particular function. Generally, the program obtains the appropriate lock relating to the data that it is going to update or use, performs the operation, and then frees the lock. Whether or not the system is disabled during this operation depends on whether or not interrupts can be handled.

The locks that are used include one per address space, a dispatcher lock, multiple IOS locks, a lock for real storage management, locks for global supervisor services, and locks for virtual storage management. This means that, for example, a GETMAIN can be performed in a user's private address space at the same time that another GETMAIN is being done in another user's space, or an interrupt is handled by IOS. The net result is that the system is enabled for interrupts more often and more elements of the control program can execute in parallel. The primary advantages here are to a tightly-coupled multiprocessing system, but some of these carry over into other environments. (See Figure 8.)

MAIN STORAGE EXPLOITATION

Because of recent changes in the relative costs of various hardware components, the trade-off between main storage usage and other activity in the system has changed. In Release 2, our goal has been to exploit main storage by trading it for CPU and I/O activity wherever possible.
In MVT and VS2 Release 1, data sets are generally placed on a device and all access to this data must go to that device. Main storage content is limited to the images of programs.

Certainly, in many environments there is data whose usage is high enough to warrant at least a part of it being resident on a higher speed device or perhaps in main storage. In fact, there are environments where some blocks of data receive higher usage than some of the pages of the program, and ideally should receive preference for main storage occupancy. In Release 2, we have attempted to move in the direction of allowing data to be placed in the storage hierarchy dynamically, according to its usage. Therefore, certain data can be resident in main storage or on a higher speed device if it has high enough frequency of usage.

The whole idea is to allow more data, more information, to be resident in main storage. Thus, access times are better for this information when it is required for allocation, termination, or OPEN/CLOSE-End of Volume processing. If usage is high enough, this information would be resident in main storage with the same advantages as with virtual I/O.

VIRTUAL I/O

Virtual I/O provides for placing data sets in the paging hierarchy. The net result of this is that if a page of data is resident in main storage, there is a reduction in I/O and CPU time. The CPU time is reduced because of the elimination of I/O interrupt handling, channel scheduling, and task switching. Because blocking is done automatically at 4K, greater efficiency may result. When I/O is done, it is performed by the paging mechanism, with generally more efficiency than with the conventional techniques.

An additional advantage of virtual I/O is that no direct access device space management is required, and therefore allocation time is faster. Because space is allocated in 4K blocks as needed, space utilization is also more efficient.

In Release 2, temporary data sets are supported for virtual I/O in a compatible way. No JCL or program changes are required for SAM, PAM, DAM, XDAP, and the equivalent operations in EXCP. Any program dependencies on direct access device characteristics are handled in a transparent way.

SCHEDULER WORK AREA

The Scheduler Work Area allows a job's job queue information to be contained in its own virtual storage. Thus access times are better for this information when it is required for allocation, termination, or OPEN/CLOSE-End of Volume processing. If usage is high enough, this information would be resident in main storage with the same advantages as with virtual I/O.

LARGE ADDRESS SPACES

The use of large address spaces to achieve greater performance has been described exhaustively in other places, however, several techniques which have been incorporated into portions of the control program should be highlighted. Overlay structures have been eliminated, and the use of the Overlay Supervisor, LINK, and XCTL services has been removed with a decrease in I/O activity as well as CPU time. Spill files have been eliminated; instead, large work areas in virtual storage have been used. The allocation redesign makes use of both of these techniques.

RESOURCE MANAGEMENT

In the resource management area, our goal has been to centralize all of the major resource control algorithms. The objective here is to achieve better coordination than is possible with decentralized algorithms. With a decentralized design, two uncoordinated algorithms can sometimes work at cross purposes. By having a centralized set of algorithms, more opportunity exists for optimization.

The system resource manager in Release 2 replaces the TSO driver, the I/O load balancing algorithm of Release 1, and HASP's heuristic dispatching. Further, it provides a new algorithm to control paging and prevent thrashing.
by dynamically adjusting the level of multiprogramming. The rate at which users get service is controlled by the Workload Manager in accordance with installation specified parameters.

WORKLOAD MANAGEMENT

Priorities for this Workload Manager are not absolute, but rather are expressed in terms of a rate of service for each job. This allows a departure from the usual situation where a lower priority job gets only what is left over after the higher priority jobs have received all of the service they can get. In Release 2, two jobs can proceed at a relative rate of progress that can be set by the installation. These service rates are specified for different system loads so that the relative rate of service received by two jobs can change as the overall system load shifts. Finally, service rates can be specified for a given set of users or jobs, where a set can include as few as one user.

Figure 9 shows a sample of how this is done. There are five sets of users, A through E; and service rates varying from 0 to 1,000 service units per second. Service is expressed in terms of a linear combination of CPU time, I/O services, and main storage use. The number 1 curve, which might be considered for a light load, shows the users in groups A and B receiving high service rates, users in groups C and D slightly less service, and E even less. User sets A and B might be two types of TSO users, C and D, high turnaround requirement batch jobs; and E the rest of the batch jobs.

As the load gets heavier, the installation has specified that they would like more of the degradation to apply to the users in Sets D and E, and the least degradation to apply to sets A and B. Curve 4 represents the heaviest load where users in set A get significantly better service than anyone else, and users in sets C through E receive only what is left. The system attempts to operate on the lowest numbered curve; however, as the load gets heavier, it degrades the service seen by each of the sets of users proportionally to the way shown by the curves. That is, in going from curve 1 to curve 2, it reduces the service seen by users in category C more than for category A.

A set of reports is produced which the installation can use to determine the response time or turnaround time and throughput that is being produced by the system for each user set. Should an adjustment be required, a higher rate of service specified for a set of users will yield better response time or turnaround time. Our objective here is to provide a relatively simple way to achieve discrimination between users and to provide the right level of service to each group of users.

RECOVERY

Recovery mechanisms in the system have also been overhauled in a major way. A significant amount of work has been done in this area. Our goal is to contain errors to the lowest possible level, and either to recover from an error so that the system can proceed as if the error never occurred, or at least to clean up so that the effect of the error is not felt outside of the function in which it occurred. In this area we have really recognized that it is not enough to have code with a minimum number of bugs, but rather to have a system that minimizes the effect of the failures that do occur.

The same approach for minimizing software failures is used for hardware error recovery as well, especially in the multiprocessing environment. Generally, the method is to provide specialized recovery routines that operate as a part of the main line functions, and which receive control whenever an error is detected by the system. There are approximately 500 such recovery routines in Release 2.

INTEGRITY

In Release 2 we have closed all of the known integrity loopholes in VS2. This means that unauthorized access or use of system facilities and data or user data, is prevented, both for accidental as well as intentional actions, and we will now accept APARs for integrity failures. Integrity is a prerequisite for adequate security, where security is defined as an authorization mechanism to distinguish between what various users can do. Moreover, integrity should also provide for an increased level of reliability.

SERVICE MANAGER

In Release 2, we have provided a new transaction-oriented dispatching mechanism which allows the efficient creation of new units of multiprogramming. Our goal here was to increase performance by trading off function. This new unit of multiprogramming differs from the OS task in that it is not a unit of resource ownership or recovery. The new facility, called the Service Manager, is used by the Release 2 supervisor, JES3, IOS, VTAM, and the version of IMS for use with VS2 Release 2. This mechanism can also be used by appropriately authorized user programs. For example, a VTAM application.
RELEASE 2 PERFORMANCE

Summarizing what has been done in Release 2 from a performance standpoint the following points are noteworthy. Because of the reduction in serialization and the tradeoffs that can be made between I/O activity and main storage, the system can better utilize the CPU.

Figure 10 shows conceptually the CPU and I/O overlap for an MVT job. The wait state time is comprised of I/O wait plus other waits caused by serialization on system resources. These wait times are reduced as a result of virtual I/O, scheduler work area, the new catalog, allocation, etc. However, this wait time may be extended due to paging. This is typically rather small, especially in a large main storage environment.

On the other hand, CPU time generally will be reduced as a result of virtual I/O activity since fewer interrupts are handled, etc. Other overhead is also reduced because the reduction in I/O and wait time generally allows the CPU to be fully utilized at a lower level of multiprogramming. On the negative side is degradation due to extra instructions required in Release 2 because of enhanced recovery, integrity, and new function. The overall effect is that individual jobs tend to look more CPU bound.

The general performance characteristics of Release 2 are significantly different than previous OS systems. The system now is generally more responsive, in that there is better consistency, with fewer long responses caused by the processing requirements of other jobs and the operator. Because fewer initiators can be used, and because of the reduction in bottlenecks, batch turnaround time can be improved. And, with the System Resource Manager, the installation has more control over the service seen by an individual user or job.

VS2 RELEASE 2 ENVIRONMENTS

The following summary shows how the features of VS2 Release 2 apply to various environments, such as

- Multiple large applications,
- Data base/data communications,
- Time sharing.

Batch, Multiprocessing, and finally, Operations.

MULTIPLE APPLICATIONS

One of our major goals is to allow multiple applications to operate effectively in a single system complex. This is theoretically highly desirable, but previous operating systems have had insufficient capabilities to shift resources dynamically from one application to another as the load changed. Perhaps even more important, failures in one application often brought down other applications, or even the entire system. There was also insufficient separation of applications from a security point of view. Release 2 provides both better isolation and integrity to address these problems. With virtual storage and other facilities in Release 2, more dynamic control and use of resources is also possible.

TELEPROCESSING

In the teleprocessing area, Release 2 is intended to be a base for high performance data base/data communications applications. VSAM, VTAM, Service Manager, Virtual I/O, Large Address Spaces, and the new Allocation all provide tools for such applications.

TIME SHARING (TSO)

For time sharing, a number of performance improvements have been made: SWA, the Catalog, etc. Compatibility between TSO and other areas of the system is more complete, primarily because the rest of the system is now more like TSO. Dynamic device allocation with volume mounting represents a new facility for TSO users that are authorized by the installation. SYSOUT data can be routed through JES2 or JES3 to a remote high speed work station to provide bulk output for a TSO user. Finally, large address spaces have been classically considered a time sharing function.

BATCH PROGRAMS

In the batch area there are a number of performance improvements as well. Dynamic data set and device allocation is provided for the first time for the batch programs. Among other things, this allows the ability to start printing SYSOUT data sets dynamically prior to the end of the job. This can be done with a minimal change to the JCL and with no programming change. Remote job entry is provided through the JES2 and the JES3 packages.

MULTIPROCESSING

Multiprocessing has traditionally placed a heavy emphasis both on reliability and availability as well as
performance. In the reliability area, a number of hardware improvements have been made. Certainly the increased integrity, both between the operating system and the user, as well as between the various parts of the control program, provides the potential for better reliability. Most important are the new recovery mechanisms in Release 2.

In the performance area, the complexity of a multiprogramming system is generally increased in the MP environment; however, the facilities for increased multiprogramming efficiency in Release 2 go a long way toward achieving good performance on MP systems. The exploitation of main storage is also important, since most MP systems are configured with large amounts of main storage. The multiple locks of Release 2 are aimed directly at minimizing contention for control program usage of the CPU’s in a tightly coupled multiprocessing system.

OPERATIONAL CHARACTERISTICS

On the operational side of the system, our goal has been to have less dependence on the operator for performance. Generally, the system is significantly less serialized on the operators and their activities. The system, we feel, is simpler to operate. Tuning should be significantly easier as well. There are fewer bottlenecks to balance, fewer parameters to specify, and the system is more self-tuning. Moreover, the system can allow for more control over its own operation with the new integrity facilities, the System Resource Manager, and so on.

CONCLUSION

The purpose of this paper has been to provide some insight into how we arrived at the design of Release 2. Our objective was to provide some very significant increases in function and availability, with improved performance characteristics, and with a high degree of compatibility to previous OS systems.

We think that the system has done a good job of meeting these often conflicting objectives. OS/VS2 Release 2 represents a major step forward, but it is only a first step, since it provides the base on which we will build total support for advanced applications in the 1970’s.