*A procedure of observation and correction is presented for the coarse tuning of an MVS system by analyzing it into its software and hardware components and increasing their efficiency successively. The method involves adjustments to swapping, the input-output load, the CPU load, main storage, and the system resources manager. Also discussed are performance measures necessary to characterize a system, tools to tune a system, and various aspects of data gathering and the effects of adjustments on system parameters. Two illustrative case histories are also given.*

# Performance tuning in OS/VS2 MVS

**by T. Beretvas**

Probably as technologically and practically significant as bringing virtual storage into the world of commerce and industry has been the introduction of Multiple Virtual Storage (MVS). The earlier virtual storage systems are often termed "single virtual storage" systems because a number of users share concurrently the total available virtual storage. Single virtual system products are exemplified by the IBM Disk Operating System (DOS/VS) and OS/VS1 and OS/VS2 Release 1. Later releases of OS/VS2 (i.e., Releases 2 and 3 and enhancements thereof) are known as Multiple Virtual Storage or simply MVS. In MVS, each user has all the virtual storage (16 megabytes) permitted by the addressing structure available to him. The design enforces complete separation of one user's programs and data from those of another.

MVS is an operating system that is significantly more complex than its predecessors, and requires a large share of the computer's resources (particularly in terms of main storage). This fact and the cost-consciousness of computer users necessitate the tuning of the computer system to exploit its resources to the greatest possible extent. Often tuning is a cost-effective alternative to the acquisition of additional hardware. MVS lends itself to system tuning much more efficiently than previous operating systems, since

a conscientious design effort has been made to provide the computer user with external capability to allocate system resources among the various users of the system. The primary vehicle of this adjustment is the System Resources Manager (SRM). The SRM contains many of the performance-oriented algorithms of the operating system.

System tuning consists of much more, however, than manipulation via the SRM. Because of the complexity of MVS and the system applications, the resources of the installation may not be optimally used. One installation, for example, may use teleprocessing intensively, with highest priority being given to such work, whereas another installation may emphasize completion of batch work on time. In either case, storage, CPU capacity, or Input/Output (I/O) devices may represent scarce resources, the use of which must be optimized to attain the installation's performance objectives. Techniques of system tuning are designed to maximize system throughput and/or to improve response time under such a variety of conditions. Typical manifestation of poor tuning might be excessive paging, jobs queued up waiting for I/O completion while the CPU is idle, or contention for storage revealed by excessive swapping.

The tuning of an MVS system may be accomplished in stages. The subject of this paper is first-stage or coarse tuning. In the first stage, certain parameters are examined, and the system is adjusted so that contention is minimized as indicated by these parameters. After the coarse-tuning parameters have been adjusted, the overall response of MVS may still not be the one desired. If that is the case, further tuning may be required. The objective of first-stage tuning is to bring the system into the range of perhaps sixty to seventy percent of optimum. Both the optimum and the current point of operating efficiency are continuously fluctuating, but first-stage tuning is a good place to begin to understand the practical effects of MVS tuning.

The examples given and the methodology used are based on the OS/VS2 Release 3.6 version of MVS, but the methodology is generally usable. Release 3.7 of MVS, with the installation of the Supervisor 2 enhancement, contains significant changes in some performance algorithms. This paper, therefore, includes references to this version of MVS (i.e., OS/VS2-3.7).

After a presentation of some historical considerations of MVS, the following topics are discussed:

- System performance measures.
- Performance tools in an MVS system.
- Performance parameters.
- MVS performance control philosophy.

- Performance tuning an MVS system (swapping, I/O load, CPU load, main storage, SRM).
- Evaluation of a tuning effort.
- Installation control consideration.
- Case histories.

**history**

The IBM operating system that is now known as OS has three major versions, MVT, SVS (OS/VS2-1), and MVS (OS/VS2-2 and OS/VS2-3). (The two operating systems with more limited functional capabilities, MFT and OS/VS1, are not discussed in this paper.) MVT represents a sophisticated enhancement of computer usability by allowing the concurrent executions of multiple jobs, and is limited primarily by the capability of fitting jobs into main storage. SVS removed a significant part of the main storage restriction by allowing that during job execution at any given time only portions of a job region size need reside in main storage. Other parts of a job region not needed at a given period need not reside in main storage, but rather in virtual storage. Thus main storage page frames are freed by *page stealing*.

Because of main storage limitations on many MVT systems, CPU utilization has typically been in the thirty to fifty percent range. In contrast, SVS (and MVS) systems often run with eighty to one hundred percent CPU utilization. The main storage limitation in MVT exists in part because of the cost of main storage and in part because often uniform job sizes of 100 K to 250 K bytes are used in many installations, even though the effective main storage utilization of a job step at any given time may be only 30 K–50 K bytes. SVS and MVS have eliminated the main storage constraint by the use of virtual storage; i.e., only the main storage actually required in a particular job step is occupied, often with the result that the multiprogramming factor has increased. Another new consideration has been the increased functional capability provided in SVS and MVS, often leading to heavier CPU utilization by the system as opposed to the user's program. In MVT, therefore, the key constraint on a system's ability to perform work has been main storage. In SVS and MVS, CPU utilization also becomes a significant factor. In SVS, main storage contention between competing users is resolved by page stealing, that is, the freeing up of unused page frames.

An independent development was the introduction of a Time Sharing Option (TSO) into MVT. Initially, MVT had been essentially a batch-oriented system in which most jobs competed for CPU availability except when waiting for completion of an I/O operation. The TSO user at a terminal has a different utilization pattern. His function usually requires minimum CPU use followed by long delays consisting of transmission time and terminal use time lumped together as "think time." Given this pattern of computer utilization, it is natural that a swapping philosophy was in-

troduced for TSO, both in MVT and SVS. Thus a TSO region has been defined so that each TSO user operating in a region is swapped into main storage and allowed to execute for a period of time. This is often the execution time required by the TSO command. Then the main storage region is often swapped out to free up the region for another TSO user. As an example, an average main storage occupancy of three seconds and a think time of 27 seconds allows ten TSO users to share the same main storage region provided there are no delays. With the advent of MVS (OS/VS2-2), TSO ceased to be an option and became an integral part of the operating system. Furthermore, TSO swapping was extended to most users. Thus most address spaces (i.e. users) can be (and are) swapped. This swapping philosophy was very useful in TSO, and it was expected that it would be heavily used in the teleprocessing (TP) and data communications MVS environments. Thus many short TP transactions would compete and be swapped in and out as required. Furthermore, swapping was to be used in MVS to relieve main storage contention in lieu of page stealing as in SVS.

MVS has also removed many serializing bottlenecks that have existed in MVT and SVS, but performance problems still remain. This paper discusses MVS tuning procedures that eliminate or diminish the effects of some common performance inhibitors encountered in MVS systems. We begin with a discussion of system performance measures.

## System performance measures

Each installation defines its performance objectives differently. The objectives reflect habit as well as experience, and primary importance is placed on different types of work by the computer installation. Most installations agree, however, on the following key performance measures:

- *Batch job elapsed time:* the time elapsed from job entry into the computer system until the job leaves the computer system.
- *Batch job turnaround time:* the elapsed time from job delivery to the computer room until the job output is delivered.
- *Throughput:* the number of jobs completed during a certain period of time.
- *Response time:* the time from the entry of a transaction at a terminal until a response is obtained from the system.
- *Transaction rate:* the number of transactions completed during a certain period of time.

Often performance evaluation is done on a benchmark basis. A batch job stream is constructed and held in a queue. An internal

reader program reads the preloaded jobstream from a direct access device. Then the queue is released, and elapsed time is measured from the time the queue is released until the last job is finished. These measurements are usually fifteen to thirty minutes in length for practical reasons, and are not very useful unless the initiators become associated with the jobs quickly (i.e., no starting delay) or if a trail-off period occurs (i.e., the initiators do not become inactive in quick succession, say less than a minute). The system log or the System Measurement Facility (SMF) can be used for this measurement.

Another way of evaluating performance is to select the most heavily used half-hour period in the test environment, when performance problems are most clearly manifested, and measure and evaluate performance measures during that period.

Response time in a teleprocessing or a TSO environment can be measured with a stopwatch. A shortened version of the real TSO response time is, in fact, measured by Measurement Facility/1 (MF/1) or its enhanced successor, Resource Measurement Facility (RMF) in an MVS environment. MF/1 distinguishes among very short, moderately short, and long TSO transactions. MF/1 measurements represent internal system response times, whereas real TSO response times also include such additional external delays as line delays. The TSO transaction rate is also measured by MF/1.

**performance tools and facilities**

MVS is a more advanced operating system than SVS and MVT and provides the following facilities and tools for performance tuning that have previously been unavailable:

- The system measurement facilities MF/1[1] and RMF,[2] which give the user of the system the capability of monitoring and assessing system performance.
- The System Resources Manager (SRM), a component explicitly designed to provide the capability of resource allocation and control.[3]
- Various special programs[4] that provide a detailed performance analysis capability.
- Accumulated documented operational performance experience such as is shown in Reference 5.

Other conventional performance tools, such as a System Measurement Facility (SMF), Generalized Trace Facility (GTF), and a system log are also available.

**performance parameters**

If the system performance measures seem to indicate inadequate performance, these performance parameters are used to identify performance bottlenecks. The computer system has three major resources, the central processing unit, input/output facilities, and main storage. MF/1 provides the means of measuring the use of these three resources via key performance parameters.

Table 1 Typical system resource utilization parameter values

| | *Low (under-utilized)* | *High (over-utilized)* |
|---|---|---|
| CPU utilization | 70% | 98% |
| Main storage utilization (available frame count) | 90 | 10 |
| I/O utilization (channel and device utilization) | | |
| Block multiplexer channel utilization | 10% | 30% |
| Selector channel utilization | 10% | 60% |
| Seek direct access device utilization | 10% | 40% |
| Nonseek (IBM 2305) direct access device utilization (for one exposure) | 10% | 30% |
| Maximum paging rate for IBM 3330-1 (pages/s) | 5 | 15 |
| Maximum paging rate for IBM 2305-2 (pages/s) | 10 | 30 (50 in OS/VS2-3) |
| Swap ratio | — | 1.2 |

It has been observed that if any one of the three system resources is used beyond a certain limit, the system is faced with a performance bottleneck that must be removed before the system performance measures can be improved. On the other hand, low utilization of a system resource may indicate a bad balance of hardware resources. For example, low CPU utilization may be due to insufficient main storage or I/O resources. The author's experience indicates typical system resource utilization in the MVS environment as given in Table 1. In a well-balanced system, utilization data are between the low and high figures. Note that these data represent only general guidelines based on OS/VS2-3.6, and may differ from those that characterize optimum conditions under specific types of workloads. Performance parameter values were collected by MF/1 over a test interval of about fifteen minutes.

**MVS performance control philosophy**

As mentioned previously in this paper, MVS has been designed with address space swapping as its primary control mechanism. The System Resources Manager (SRM) uses swapping primarily to overcome main storage shortages. The SRM is also designed to swap address spaces that are using an excessive amount of I/O and CPU resources, or to replace an address space in main storage that is ahead of its performance objective with an address space that is behind its performance objective.

The original MVS philosophy, therefore, called for over-initiation, by which more batch initiators are started than can be accommodated in main storage at one time. Thus SRM can—in theory—balance the system resources and the use of these resources among the many users.

In practice it has been found that the over-initiation philosophy is misconstrued primarily because swapping is a heavy consumer of system resources, in terms of CPU cycles and I/O device utilization. Accordingly, it has been found more productive to modify the swapping philosophy by minimizing swapping, especially batch swapping.

A modified corollary objective is to limit swapping to TSO address spaces, preferably at transaction termination time. This corollary objective can be accomplished only for short TSO transactions. Sometimes most of the MVS tuning effort has to be concentrated on minimizing swapping.

In OS/VS2-3.7, swapping is not used directly to alleviate main storage shortages. The same effect, however, is accomplished indirectly. If paging is excessive, the SRM resorts to swapping to reduce the paging bottleneck.

A secondary control mechanism acts on dispatching priorities. Within the Automatic Priority Group (APG), heavy CPU users (i.e., those with infrequent I/O waits) receive low priority, whereas heavy I/O users receive high priority. If there are enough users in the system main storage, low-priority address spaces may not be dispatched at all. Thus dispatching priority has to be judiciously manipulated to give good response time to teleprocessing applications and to TSO transactions to reflect the priorities of the installation.

A third major control mechanism acts on page stealing. In this mode, page frames that have been unused for a period of time are stolen; i.e., they are placed in the list of available frames. In OS/VS2-3.6, pages in address spaces with low priority are stolen if the CPU use of the address space exceeds a given limit. Pages in the System Paging Area (SPA) and high-priority (system) address spaces are stolen on a real-time basis. In OS/VS2-3.7, page stealing takes place across all address spaces whenever a page frame shortage occurs. In both VS2-3.6 and -3.7, pages are also stolen from address spaces prior to swapout, to minimize the paging load due to swapping by reducing swap sizes. In some cases, the page-stealing algorithms may have to be manipulated by local modifications so that excessive page use (or stealing) in the System Paging Area (SPA) and in key address spaces is avoided.

## Performance tuning

The tuning principles of MVS are essentially an extension of tuning principles in MVT and SVS. Tuning in all three operating systems is basically the repetitive application of the simple operation of identifying and removing major bottlenecks in the system. Upon the successful removal of a system bottleneck, another bot-

tleneck is often identifiable. The successive removal of bottlenecks, however, can be a never-ending operation. This is particularly so because the load presented to the computing system is not a constant, but rather time variant.[6] Thus, a bottleneck may persist for a period of time, then be removed, and then another bottleneck may appear in its place.

Tuning normally begins by selecting perhaps a fifteen-minute period during which the system is heavily used. During the selected period, MF/1 is used to obtain data about the system. The MF/1 report is then examined to identify system bottlenecks. After this initial tuning analysis, other tools used in conjunction with MF/1 for more detailed tuning include the System Information Routine (SIR),[4] System Measurement Facility (SMF), and various reduction programs. The most detailed tuning makes use of the Generalized Trace Facility (GTF) data and analysis.

Then, using the normal methodology of scientists, the performance analyst makes a hypothesis about the cause of the bottleneck. While doing this, he may find that more data are needed. With sufficient facts, the analyst decides on corrective action. The next step is to perform an experiment and make another measurement after the corrective action to prove or disprove the hypothesis.

The unfortunate part of this procedure is that often, for several reasons, the experiments may not lead to a clear conclusion as to whether the performance bottleneck has been eliminated. The measurements, which must be of short duration for practical reasons, take place on a production system, and thus may not be repeatable. The change in measured results may be within the bounds of the measurement error. Also, the elimination of one bottleneck may lead to the emergence of another bottleneck, necessitating the repetition of the whole process. Nonetheless, through this iterative process, dramatic performance improvements are often possible.

**swapping considerations**

The first important system bottleneck usually encountered is excessive swapping. Therefore, the performance analyst must first minimize swapping in order to reduce both CPU and I/O loads that result from swapping. The performance analyst should aim for correct main storage use because excessive swapping often indicates excessive use of main storage.

Teleprocessing (TP) applications often require consistent and reasonably fast response time (perhaps one to ten seconds). In most such cases, the TP application programs should be set to nonswappable status by specifying this constraint in the program property table or in the programs themselves by issuing DON'T SWAP SYSEVENT. If consistent response time is not an important crite-

rion, if—in other words—an occasional long response time (perhaps thirty seconds) is acceptable, then the nonswappability of TP applications can be avoided. The penalty paid for nonswappability of an application is an effective loss of main storage for other processes.

The most frequently encountered TP application is the IBM program product Information Management System (IMS).[7] It is strongly recommended that both IMS control and message regions be set nonswappable.

Nonswappability is necessary in part because IMS is a TP system. In addition, IMS 1.1.1 introduced means called "latches" to allow concurrent processing in two parts of a multiprocessing (MP) system. If an IMS region holds a latch and is swapped out, another IMS region that requires the latch cannot execute. Nonswappability avoids this deadlock.

Batch swapping can and should be avoided by controlling the number of initiators started. A good strategy is to classify the initiator structure by region sizes used, and to start a sufficient number of initiators to saturate the available main storage that is not needed by the system. As a first approximation, when the maximum region sizes are added, their sum should not exceed two times the available main storage. Such a strategy can be verified by ascertaining that batch jobs are not swapped extensively and no excessive paging takes place.

Page frame shortage conditions signaled by the AVQLO SYSEVENT cause swapping directly in OS/VS2-3.6 and indirectly in OS/VS2-3.7. If too many initiators are started, exchange swapping can occur; i.e., an incomplete batch job is swapped out and replaced by another batch job in main storage. Thus, if extensive batch swapping occurs, a reduction in the number of initiators started is often an appropriate remedy.

It may be necessary to set jobs that must be completed by a certain time and long-running batch jobs to an effectively nonswappable status. This can be accomplished by placing the batch application in a separate performance group and by setting its Interval Service Value (ISV) high (e.g., to a value of 10 000) or by making it nonswappable. Swapping these jobs may lead to unpredictable termination times and may be an unnecessary additional load on the system.

Swapping of TSO users is a design feature of MVS. The performance analyst should attempt to provide that trivial TSO transactions, such as simple data set editing operations that consume a minimal amount of system resources, are not swapped during their execution. Experience shows that in many well-balanced

systems eighty to ninety percent of all TSO transactions are trivial ones. If multiple swaps are necessary for the completion of the trivial TSO operation, the swapping load becomes very heavy, and both TSO response time and batch throughput deteriorate rapidly. One may assume that trivial TSO transactions must be completed in less than five seconds. For this to happen, sufficient system resources must be provided. In practice, sufficient main storage frames must be available to swap in ready TSO users. Access to the CPU is to be provided for the swapped-in transactions, and swapping itself should not be a bottleneck through a dearth of paging devices. The performance analyst may perform a quick calculation to assure the availability of sufficient main storage for TSO users. The number of main storage frames $N$ required for TSO use is given approximately as follows:

$$N = A \times \frac{RT}{TT + RT} \times MS$$

where

$A$ = number of active TSO users in the system.
$RT$ = system residence time of a TSO user.
$TT$ = user think time.
$MS$ = number of frames required for each TSO user.

Typical values of these parameters are the following:

$RT$ = desired TSO trivial response time plus 1 second.

Desired TSO trivial internal or system response time (as reported by MF/1) can be set, for example, to 1 second for System/370 Model 168 installations, and 3 seconds for System/370 Model 158 installations. Accordingly,

$RT_{158}$ = 4 seconds and $RT_{168}$ = 2 seconds.
$TT$ = 18 seconds.
$MS$ = 20 frames.

In this example, the numbers of main storage frames are the following:

$N_{168} = 2A_{168}$.
$N_{158} \cong 4A_{158}$

Thus, for System/370 Model 158 with $A = 20$ active users, $N = 80$ frames, and for System/370 Model 168 with $A = 100$ active users, $N = 200$ frames.

The number of main storage frames required for TSO use is additional to the operating system and batch requirements. Access to the CPU must be provided for the TSO user while his address space is swapped in. This may have to be accomplished through the manipulation of dispatching priorities, a function that is discussed later in this paper in the section on CPU load considerations.

A measure of swapping is the *swap ratio*, which is defined as the number of swaps divided by the number of terminated TSO transactions, the data for which are reported in MF/1. Ideally, this ratio should approach 1.0. Since terminated TSO transactions are automatically swapped out, a ratio of 1.0 means that only terminated TSO transactions are swapped.

The performance analyst may estimate the TSO transaction rate $T$ in the system, which, in turn, gives an estimate of the paging rate $SPR$ due to swapping.

$$T = \frac{A}{RT + TT}$$

$$SPR = F \times T \times MS$$

The factor $F$ reflects the fact that whereas both swap-in and swap-out take place, a swapped-out working set may be reduced through page stealing at swap-out time. Experience shows that the value of $F$ is in the range 1.7 to 2.0.

A typical value for the quantity $RT + TT$ is 20 seconds; thus,

$$T = \frac{A}{20}$$

and

$$SPR = 1.7 \times A$$

assuming that most transactions are trivial. Thus, for a System/370 Model 158 with $A = 20$ users, $T = 1$ transaction per second, and $SPR = 34$ pages per second; for a System/370 Model 168 with $A = 50$ users, $T = 2.5$ transactions per second and $SPR = 85$ pages per second. The calculations just given represent a first approximation. Congestion, particularly in the I/O area, introduces nonlinearities and increases response time.

Paging not due to swapping should be small in a well-balanced system. In such a system, the Nonswappable Paging Rate (NPR) includes the System Pageable Area (SPA) paging, Virtual Input/Output (VIO) paging, and the recovery of address space pages lost through stealing. As a rough guideline, NPR in paging should rarely exceed five pages per second on a System/370 Model 158 in a non-TSO environment, and ten pages per second in a TSO environment. NPR for System/370 Model 168 is about 15 pages per second in a non-TSO environment, and 25 pages per second in a TSO environment.

Whenever the swapping frequency is reduced in a system (usually by minimizing batch swapping), considerable CPU and I/O savings can be obtained, since the swapping function requires the execution of many instructions and the transmission of all the swapped pages.

**input/output load considerations**

After a swapping problem has been relieved, the performance analyst can address the next bottleneck, which is usually in the I/O area. The I/O resource use can be excessive and can consequently be reduced. The use of I/O resources may also be incorrectly distributed, and may therefore require load balancing, which begins with the very old and honored task of *data set placement*. Data set placement is distribution of data sets such that no channels, control units, and I/O devices are excessively used. Furthermore, on devices with multiple data sets, seek distances between data sets should be minimized. Bad data set placement in any operating system makes it impossible for the system to achieve its full potential. In MVS initial tuning of I/O configurations can be done to a large extent by the channel and device use reports of MF/1, which can be used to identify channel and device bottlenecks.

Channel bottlenecks may be deduced from unevenly distributed channel utilization data, or from channel-busy and CPU-wait data in excess of ten to fifteen percent. Excessive channel utilization data (i.e., higher than thirty percent utilization on a block multiplexer channel with seek devices) can be another indicator of channel bottlenecks. Direct Access Device (DASD) bottlenecks can be suspected if average queue length is higher than 0.05 (as reported by MF/1 on the basis of sampling) or if DASD utilization data are above forty percent. A DASD bottleneck may also be present if the device activity count is higher than fifteen accesses per second.

The first problem in data set placement is that of paging data sets. The performance analyst, therefore, should evaluate the paging rate required in the system to make sure that an adequate number of paging data sets are defined to handle the paging load and that the paging devices are spread across channels and control units so as to reduce contention. Paging data sets should not be placed on channels with heavy tape use. In a heavily loaded TSO system (where there are 1.5 or more TSO transactions per second), the use of an IBM 2305-2 storage system as a paging device is desirable for fast response time.

The Auxiliary Storage Manager (ASM) paging algorithm maintains the (assumed) position of the seek arm for each paging data set and, when next using that data set, it selects the nearest cylinder beyond the one last processed regardless of the actual physical position of the seek arm. The ASM does not recognize interference on the seek arm position caused by another data set. Accordingly, two paging data sets should not be placed on the same seek device, and only low-activity nonpaging data sets (if any) should be placed on the device. Otherwise, serious performance degradation may result.

As in the case of paging data sets, TSO user catalogues, SPOOL data sets, user data bases, and SCRATCH space may have to be spread across volumes, control units, and channels to minimize contention and enhance availability. In large systems, SPOOL data sets and catalogues are also candidates for storage on IBM 2305 drums.

One way of reducing I/O accesses is to make use of main storage capacity, where available, by using larger buffers and block sizes for the Job Entry Subsystem (JES2), catalogues, and user data sets. In particular, use of VIO (for temporary data sets) essentially results in data blocking into page-size units, since physical I/O operation takes place only for the whole page regardless of the actual block size used.

For small data sets, the use of VIO may mean the avoidance of delays due to I/O operations, through the use of a page-reclaim function. Page reclaim for VIO occurs when a written VIO data set is subsequently read, but the written page frames still contain the original data and are reclaimable for the reading operation. This obviates the need for physical I/O operation for reading.

The use of larger block sizes on I/O data sets leaves the amount of transmitted data unaltered, but it reduces the number of control bytes transmitted, and, more importantly, reduces channel start-up times, search and seek frequency, and delay for direct access devices. Similarly, large block sizes on tapes mean less frequent occurrences of inter-record gaps. In both DASD and tape devices, the busy condition of these devices is reduced considerably. Also in both DASD and tapes, larger block sizes mean better space utilization by the device.

This argument assumes the frequently occurring practical case of small block sizes. In general, block sizes should be optimized rather than unconditionally maximized, since unbridled growth may lead to unnecessary wastage of main storage and wasteful data transfer, in the case of small data sets. Thus I/O load tuning, based on MF/1 reports, can eliminate many gross performance errors.

Upon completion of I/O balancing, i.e., spreading and/or reducing the I/O load, the MF/1 report should show a reduction in channel and device utilization and device activity counts. However, I/O bottlenecks may still prevail, as identified for example by the I/O queue lengths reported in MF/1. The performance analyst then may have to apply more refined I/O tuning techniques.

Detailed data set placement considerations are necessary when multiple data sets appear on a single direct access device. The

objective of performance tuning under these conditions is to minimize seek distances (to a few cylinders) among the most frequently used data sets. Such performance tuning requires use of the Generalized Trace Facility (GTF) and appropriate reduction programs to identify the most frequently used physical cylinders, in order to give guidance to the most frequently used data sets, which are to be placed on adjacent cylinders.

Another very important facet of I/O load reduction is the consideration of module utilization. Modules are fetched from libraries, particularly the LINKLIB and the TSO command library, and various local libraries. Many reentrant modules that are frequently used in various installations do not normally appear in the pageable Link Pack Area (LPA). The performance analyst can reduce the number of fetch I/O operations required if he carefully examines reentrant module use with the aid of GTF and reduction programs, and places frequently used reentrant modules in LPA. Similarly, frequently used nonreentrant modules (e.g. compilers) should be identified and a fixed BLDL list set up for them. Often the packaging of modules does not take the paging factor into consideration. Modules should be packaged on page boundaries and should use a large percentage (over ninety per cent) of storage bytes available within a page. When a whole or part of a module is smaller than 4 K bytes, the same page should be shared by another module(s) with affinity to the first module. "Module affinity" implies that when a certain module A is called, another module B is also called, whether directly by module A or indirectly. Usually, modules A and B are associated with the same function.

Another strategy is to package frequently used modules together, regardless of their affinity. This strategy has been found very useful and practical since it can easily be tailored to the requirements of each installation.

The performance analyst should consider the use of page fixing (i.e., making certain pages permanently resident in main storage) for particular Link Pack Area (LPA) modules. It is true that frequent reference to a page results in the functional fixing of these pages. However, there are modules such as the STIMER that are used only about every half second, which is not frequent enough to result in functional fixing, but is frequent enough to represent a real reduction in paging when these pages are fixed. Naturally, fixed pages mean an effective loss of main storage frames and should not be used indiscriminately.

There are some modules that require fixed status during part of their execution (because of locking), and, consequently, they are constructed to make use of PGFIX/PGFREE services. Some of these modules have been changed to be sensitive to whether they have been fixed by the installation, and thereby avoid issuing the

PGFIX/PGFREE SVCs if not needed. Fixing such modules, even if they are used frequently enough to remain in storage without fixing, reduces the system overhead. Modules such as FETCH are not fix sensitive. A local performance fix for these modules may be useful.

One way to reduce the I/O load is to eliminate unnecessary I/O operations. Savings can be achieved by not writing unnecessary or unusual SMF records and in some cases suppressing JES2 journaling. The reduction in I/O operations may lead to other savings, such as the avoidance of main storage or CPU interference by the channels. The most important gain is through a reduction in contention for the hardware in the I/O path, such as channels and control units, and the avoidance of unnecessary instructions in the CPU for the I/O transaction.

**CPU load considerations**

The Central Processing Unit load in the MVS system has become a more important consideration than it is in MVT. In MVT, system performance is often constrained by the availability of real storage. Thus only a few initiators can be started. Initiator bottlenecks have been largely removed in MVS. However, the use of virtual storage has introduced paging and swapping overhead, and many of the supervisory functions, such as EXCP, have increased in length because of the overhead associated with virtual storage operations such as the relocate function. MVS has introduced system recovery and MP capabilities that also increase the number of instructions required for the execution of supervisory functions. It is clear, therefore, that one appropriate action for reducing the CPU load, which has the effect of reducing a CPU bottleneck, is to reduce the number of supervisory services invoked.

Many of the tuning actions discussed earlier in this paper result in reducing the number of supervisory services used. Thus the use of increased block sizes not only reduces the load, but also reduces the number of EXCPs required, which in turn reduces the number of CPU cycles used. The use of VIO often has a similar effect. Also, reduction of paging and swapping means a significant reduction in CPU cycles used for paging. User programs should be reviewed to reduce use of such SVC instructions as GETMAIN, XCTL and LINK. For example, the repeated use of GETMAIN can be avoided by the use of a single large GETMAIN.

Another consideration in a heavily CPU-loaded system is the distribution of dispatching priorities. All jobs without explicitly defined dispatching priority are classified into the Automatic Priority Group (APG). In OS/VS2-3.6, priorities within the APG are rearranged by the System Resources Manager (SRM). As a consequence, fast response requirements (such as TP applications

and to a certain extent TSO) may fall into the APG. TP applications should, therefore, often be assigned priorities above APG so that response time is adequate, in spite of a heavy batch load. The Telecommunications Access Method (TCAM) and IMS should have very high priority.

Response time for trivial TSO transactions merits the same consideration. It is possible to set TSO priority altogether above APG to improve TSO response time. This approach is somewhat dangerous, however, since it enables nontrivial TSO transactions (including those with extensive error loops) to monopolize the system. Thus in OS/VS2-3.6 systems with many TSO users it is advisable to leave TSO in the APG, or to use a nonstandard-performance ZAP. (ZAP is the capability to change an assembled program in object form.) This starts out TSO transactions above APG, but, in the last period of the TSO transaction, it drops into the Automatic Priority Group.

In OS/VS2-3.7, the APG functions have been extended. The SRM rearranges priorities only within the mean-time-to-wait portion of the APG. All considerations mentioned in connections with OS/VS2-3.6 just described also apply to OS/VS2-3.7 if the term "APG" is replaced by the term "mean-time-to-wait portion of APG." In OS/VS2-3.7, a TSO period can be explicitly set above the mean-time-to-wait portion without the necessity for a ZAP.

**main storage considerations**

In systems with small main storage capacity (two to three megabytes) the utilization of main storage has to be carefully considered. Difficulties may arise because the main storage requirements of MVS are around one megabyte and possibly more; the main storage available for user address space is thereby limited. Special attention must be paid to fixed storage requirements such as TCAM buffer spaces and Job Entry Subsystem 2 (JES2) buffer sizes. The analyst asks such questions as: Should buffer size be 4 K bytes or less? Are all buffers really needed? Similarly the packaging of TCAM, other TP applications, Link Pack Area (LPA) modules, and user programs should be carefully reviewed. Is it possible that part of a page frame is wasted either by being unused or by containing modules that are not related to (i.e., not called by) the other modules that reside in the same page frame? GTF reduction programs and the Linkage Editor can be used for this purpose. Every page frame saved (not used) represents clear CPU and I/O saving through reduced paging/swapping.

MVS establishes the working sets of user address spaces and System Pageable Area (SPA) through page stealing. The rate of page stealing is regulated by constants in the SRM. In some instances, the rate of page stealing (especially in the SPA) has to be adjusted for best performance. Nonstandard local modifications (performance ZAPs) may have to be used for this adjustment.

OS/VS2-3.6 provides page stealing on a real-time basis from the SPA and nonswappable address spaces with priority above a given level. Below that priority page stealing occurs from user address spaces only after the address space has used a certain amount of CPU time. This distinction between the two kinds of stealing can be advantageously used. To provide page stealing from an address space with low use of CPU time, the priority can be raised. In contrast, page stealing can be decreased for an address space with low CPU use by lowering the priority below the cutoff point.

In OS/VS2-3.7, page stealing takes place uniformly across all address spaces. In some situations, it may be desirable to reintroduce page-stealing discrimination similar to that available in OS/VS2-3.6. For example, in an IMS environment, SPA and IMS address spaces may have to be favored over other user address spaces. Nonstandard performance ZAPs may have to be used for this purpose.

In general, tuning of page stealing often involves the compromise of selective increase or reduction in page stealing, with the objective of gaining available main storage for the favored address spaces, at the expense of other parts of the system.

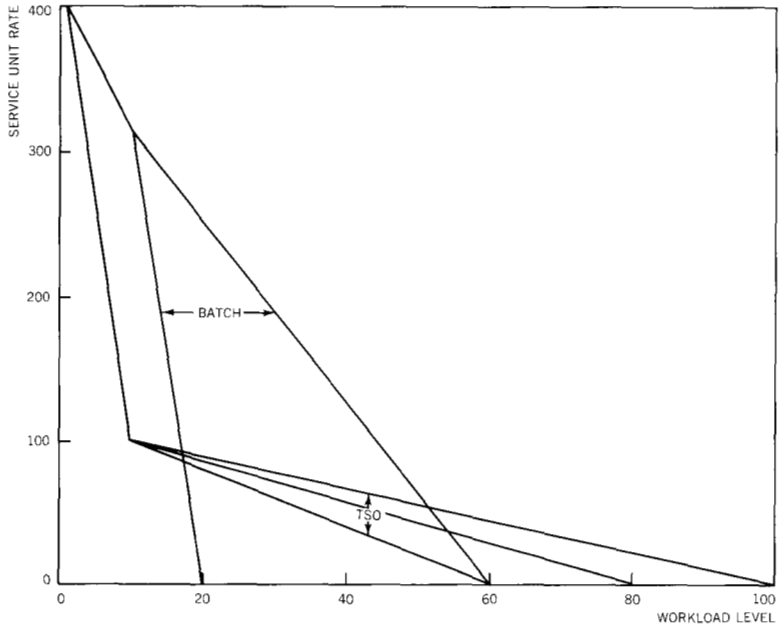
**SRM considerations**

The System Resources Manager (SRM) controls the distribution of system resources, primarily by swapping. The Installation Performance Specifications (IPS) represent the guidelines that the SRM attempts to follow. Tuning with the SRM often requires an IPS with a more suitable distribution of system resources among competing job categories. Performance improvement for one category of jobs often leads to performance degradation for another category.

The IPS supplied with the OS/VS2-3.6 system shown in Figure 1 provides a reasonably good balance. The balance provides for adequate batch throughput and adequate TSO response time. (It is, however, somewhat batch oriented.) Therefore, tempting though it may be to experiment with the SRM and adjustments to the IPS, such temptations should be resisted initially. Many other tuning adjustments can constructively be made prior to IPS adjustments. Ill-considered and incompletely understood IPS changes lead to increased swapping. It must be remembered that the modified MVS control philosophy calls for minimal swapping; in fact, the objective of the performance analyst is often to limit swapping to completed TSO transactions. Nonetheless a modified IPS can often improve TSO response time, perhaps at the expense of batch throughput, and can be used to reduce—or increase—swapping for a particular kind of job or for the whole system.

The IPS supplied by the system can be changed to reduce the swapping of batch jobs. The SRM tends to keep a job in its Interval

Figure 1 Example installation performance specifications for OS/VS for OS/VS2-3.6




Service Value (ISV) period until it uses up its service-unit quota in main storage. Hence a selective increase in the ISV values of batch jobs probably results in reduced swapping for the favored jobs. Perhaps one way of accomplishing this is to provide separate performance groups with high ISV values for batch jobs of great importance.

Yet another way is to increase the initial ISV value for all batch jobs (where PGN = 1); currently ISV = 1 K when the Performance Group Objective OBJ = 3, and ISV = 2 K for OBJ = 4. An adjustment of ISV = 2 K for OBJ = 3 and ISV = 10 K for OBJ = 4 might reduce batch exchange swapping.

Another consideration in the IPS is the Main Storage Occupancy (MSO) factor. In the IBM-supplied IPS, MSO is set to 0, which can be interpreted as meaning that main storage use is unimportant. In fact, in environments where TSO and batch users all use only 10–30 frames of main storage, and there is no shortage of main storage, MSO can be set to 0 because at least it allows a measure of repeatability that is reduced if MSO becomes nonzero. (Working-set size depends on system load.) However, in environments where main storage is critical (two- and three-megabyte configurations), a nonzero MSO factor should be used. Using an MSO factor results in rapid service unit accumulation by users with large main storage requirements. Rapid service unit accumulation ultimately results in faster swapping out of these jobs, which in turn means longer elapsed time. This is a desired goal, however, since users with excessive main storage should be penalized.

In OS/VS2-3.7, SRM has been completely redesigned in such a way that its control mechanisms are simpler and easier to understand and manipulate. SRM still controls the system resources and its main method of control is still swapping. Additional parameters for the IPS are provided to control the system. For each user domain (grouping of competing jobs and transactions), Minimum and Maximum Multiprogramming Levels (MPLs) and weights can be defined. SRM keeps the multiprogramming level (i.e., the number of jobs and transactions) of a domain within the limits defined. If system contention arises (for example, by excessive paging or page stealing), SRM decreases the MPL target of the least favored domain. The MPL target adjustment often leads to swapping out an address space from the affected domain.

In OS/VS2-3.7 systems with heavy batch and TSO loads, the IPS design becomes very important. The performance analyst must carefully consider the domains to be defined, the minimum and maximum MPLs to be defined for each, and the relative importance of the domains as identified by the weights. Bad IPS design usually leads to poor system performance. The extended parameters in the IPS can be readily used to prevent extensive main storage contention by limiting multiprogramming levels in all domains.

TSO response time can be readily improved by assuring fast swap-ins for ready TSO users. This can be accomplished by setting a high minimum MPL for the TSO domain. TSO can be favored over batch by giving higher weight to a TSO domain than to a batch domain, with the attendant result that a batch MPL target is reduced in system contention situations. CPU and main storage resources can be reserved for TSO use by setting low maximum MPL limits on batch domains. In contrast, TSO resource consumption can be limited in the system by setting a low maximum MPL for the TSO domain, with an almost inevitable response-time degradation due to transactions that are waiting for main storage during periods of heavy use. Swapping frequency can be still further reduced by the use of high ISV values.

## Evaluation of a tuning effort

The performance analyst is often called upon to perform the system tuning because of the deterioration of system performance measures. The ultimate check on such work, therefore, is the improvement in system performance measures. The analyst's work, however, cannot take into account long-term batch throughput or real (user) response time, since normally he is not operating with those factors because of time constraints in his tuning effort. Instead, system evaluation (for the short-term measurement interval) can usually be accomplished only with the aid of the system log and MF/1.

The analyst assumes that his tuning work has been successful if one or more (preferably all) of the following conditions have been met, compared to previously obtained results:

- The number of service units delivered has increased.
- The transaction rate has increased.
- The reported system response time has been reduced.
- The throughput has increased.

A subsequent check on the work of the analyst must be made, however, by comparing measurement trends over longer periods.

Sometimes, of course, one performance measure shows improvement, whereas another measure shows deterioration. The objectives of the installation then determine whether the tuning has resulted in performance improvement. Naturally, the performance analyst must also verify whether the bottleneck that he wants to eliminate has been reduced. To do this, he checks the performance parameters (CPU, channel and device utilization, swap ratio, etc.). The performance analyst can call a halt to his effort if most of these parameters are neither underutilized nor overutilized, as is discussed previously in this paper in the section on performance parameters, without any obvious bottlenecks present.

**installation control considerations**

It is necessary to use direct performance tuning, but this is not enough by itself. The installation has to orient its total system operations toward good performance. Performance analysis has been discussed up to this point, but in many installations there is no one specifically assigned to perform such analyses. In such a case, it is advisable in fact to assign a well-trained system programmer to become a performance analyst. His job is to monitor system performance, tune the system, and act as advisor to users of the system. He should also become an advisor on operational procedures. It is suggested that in an operational MVS environment MF/1 data be collected throughout the day on the SMF data set and the MF/1 analyzer program be used to reduce the data on a daily basis, for use by the installation manager and his performance analyst.

It is desirable to have a well-established system of job classes and to give priority to jobs that use only a small portion of the system resources (CPU, I/O, and main storage) over long-running and expensive jobs. Priority of small resource users can be enforced by a judicious selection of initiator classes and by the IPS.

For example, class A might be defined as that of jobs that use less than a 100-K region size for less than five seconds of CPU time. Class B might be defined to include jobs that use less than a 150-K region size for less than ten seconds of CPU time. (A direct correla-

tion between region size and main storage occupancy is assumed.) More class A initiators are started than class B initiators, and class A jobs may have their own performance groups (or domains) with better performance objectives and/or constraints than class B jobs.

SMF and JES2 exits should be used to enforce the CPU time estimates by abnormally terminating jobs that exceed their limit. Also, exits can be used to catch those who use an improper class for a long job. Inactive TSO terminals should be logged off by SMF exits. Another important consideration is to limit the size of VIO data sets used by a JES2 exit or by other methods, such as use of the IBM 2305-1 storage system as a virtual storage device for VIO.

In spite of vigorous testing, performance problems sometimes occur in the operating system because of incorrect coding. Such code sometimes manifests itself only in poor performance as opposed to ABENDs. Therefore, it is important to apply Program Temporary Fix (PTF) tapes soon after they become available, and not to be reluctant to apply so-called performance ZAPs. This way, the user incorporates the latest performance improvements into his system.

## Case histories

**case history 1**

The system installed is a two-megabyte System/370 Model 158 that is configured for TP applications, IMS data communications applications, and 15 batch initiators running OS/VS2-3.6. Initial tuning resulted in the following performance:

- Service definition coefficients: CPU = 12, IOC = 3.7, MSO = 3.0.
- Interval Service Value of TP application set to 15 000 (high) to reduce swapping.

The problem experienced is erratic TP response time and low batch throughput. Performance analysis, using SIR, reveals that the TP application swaps once per minute. It further reveals that only three batch initiators are in main storage in addition to TCAM and two TP applications. Further, both TCAM and the TP applications use an excessive amount of main storage.

On the basis of the initial analysis the following actions are taken:

- Make TP applications nonswappable; set dispatching priority to 255 (to enable stealing, which does not take place because of low CPU use).
- Reduce the number of TCAM buffers. Out of 150 buffers allocated, only 50 are being used.
- Reduce the number of initiators from 15 to 4.

- Make the IMS application privileged, so that it is swapped only if in a long-wait status. Set the IMS priority above the real-time page-stealing limit to 253. (Later it was found that the privileged setting for IMS was not sufficient. IMS had to be made nonswappable.)

The actions taken have had the following results:

- TP application main storage use has been reduced from 73 frames to 35 frames.
- TCAM use has been reduced from 55 to 40 frames.
- TP response time has stabilized.

**case history 2**

The installation has two System/370 4-million-byte Model 168 systems configured in a loosely coupled multiprocessing environment, using JES3. The system under investigation is the CPU that runs the controlling (global) side of JES3, using MVS 3.7. The system exhibits the following symptoms of unsatisfactory performance: low batch throughput and hesitating printers.

Initial performance measurements reveal the following conditions:

- Heavy utilization of the Pageable Link Pack Area (PLPA). Paging data set as measured by RMF is 48 percent; queue length is 0.09; and the SPA page-in rate is 17 per second.
- Paging in the JES3 address space, as measured by SIR, is 25 plus per second, which is considered to be heavy.
- The JES3 working storage varies rapidly between 180 and 360 frames.
- Number of initiators started is 13.
- CPU utilization is between 60 and 85 percent, as measured by the RMF.

Analysis:

- Paging is excessive, especially for PLPA and JES3.
- Storage overcommitment is too high.

Tuning actions taken:

- Paging configuration is changed so as to obtain dedicated paging devices and thereby reduce the delay in paging.
- Batch ISV values are increased in order to reduce exchange swapping in batch domain.
- The batch minimum MPL value is reduced so as to allow batch swapouts.
- Some unnecessary modules from the fix list are removed, in order to reduce serious storage overcommitment.
- Batch default region size is reduced from 800 K bytes to

256 K bytes in order to reduce the main storage requirement of some jobs.

- A local performance fix (ZAP) is used to guarantee a certain number (250) of real storage frames for JES3. This operation is called "fencing."
- A local performance fix is used to favor PLPA over other areas of the system regarding page stealing (i.e., less page stealing in PLPA, and more elsewhere).
- Number of initiators is reduced to 11.

Results of the tuning action:

- JES3 paging is reduced from 25 pages per second to 8-15 pages per second with the JES3 working set range between 250 and 360 frames.
- PLPA working set size is increased to 175 from 110 frames, with PLPA paging reduced to 12 pages per second from 17 pages per second. There is no more queuing on the PLPA data set.
- Utilization of the PLPA data set is reduced to 30 percent.

The batch throughput measurement taken over a one-day period also reveals an improvement of 10 percent in the number of jobs processed, as compared with previous results. Following this tuning action, attempts are being made to further reduce paging. When this is done, throughput is correspondingly reduced. Thus in the present storage-constrained environment, the paging level attained is necessary for adequate throughput, although the prevailing situation would be too high for many installations. The balance between paging and throughput in this installation can be further improved only by the addition of more storage.

## Concluding remarks

This paper has presented a practical initial approach to performance tuning; detailed tuning must be tailored to each specific installation. Performance tuning in MVS has become easier through the wider availability of performance tools and the use of an analytical approach to performance evaluation and improvement. Performance tuning has also become much more important because of the cost-consciousness of customers and the heavy resource use of MVS. As a further result of improved analytical capabilities, user expectations are also higher. Thus the conscientious user of MVS has to devote trained analysts to performance monitoring, analysis, and tuning. Such efforts tend to pay for themselves in cost reductions and user satisfaction because of better performance. Although the illustrations provided are mostly for OS/VS2-3.6, the methodology is generally applicable.

Tuning involves the minimizing or elimination of unnecessary CPU, I/O, and main storage use, the elimination of system bottlenecks, and the possible balancing of the system by favoring one job category at the expense of another, if necessary. Often savings in one resource means additional expenditure in another. Just as often, however, savings in one resource means savings in all. The art and science of performance tuning lie in finding the suitable mix of system adjustments that provides the best available system performance to meet the installation requirements.

CITED REFERENCES


1. *OS/VS2 System Programming Library Initialization and Tuning Guide*, Order No. GC28-0681, IBM Corporation, Data Processing Division, White Plains, New York 10604.
2. *OS/VS2 MVS Resource Measurement Facility (RMF) Reference and User's Guide*, Order No. SC28-0922, IBM Corporation, Data Processing Division, White Plains, New York 10604.
3. H. W. Lynch and J. B. Page, "The OS/VS2 Release 2 System Resources Manager," *IBM Systems Journal* **13,** No. 4, 274–291 (1974).
4. *MVS System Information Routines (SIR) Description/Operations*, Order No. SH20-1813, IBM Corporation, Data Processing Division, White Plains, New York 10604.
5. K. Soper, Editor, *MVS Tuning Report by the MVS Tuning Committee*, SSD No. 277, SHARE, Inc., New York, New York (July 15, 1977).
6. H. Hill, "Data base system evaluation," *Data Systems Proceedings, 5th Information Symposium Proceedings*, September 1975, Springer-Verlag, Berlin (1975).
7. W. C. McGee, "The information management system IMS/VS," *IBM Systems Journal* **16,** No. 2, 84–168 (1977).


Reprint Order No. G321-5077.