Discussed is a new subcomponent of the control program of the IBM OS/VS2 Release 2 operating system that has been designed to use the resources of the system to satisfy two distinct but potentially conflicting performance objectives, i.e., response and throughput.

Termed the System Resources Manager, the subcomponent controls performance by address space swapping through the use of a swapping analysis algorithm and a workload management algorithm.

The OS/VS2 Release 2 System Resources Manager

by H. W. Lynch and J. B. Page

A new subcomponent—the System Resources Manager (SRM)—has been embodied in the IBM virtual storage operating system OS/VS2 Release 2. The SRM is a new subcomponent of the control program that combines many performance-related functions that were previously handled by other parts of the operating system. The System Resources Manager has been designed to address three major problems of earlier operating systems.

system resource management objectives

In OS-based systems, the problem of distributing resources among various elements of the workload in a way that satisfies the response requirements of individual system users is handled for batch jobs by the use of scheduling and dispatching priorities. The Time Sharing Option (TSO) uses a set of parameters for the TSO driver that makes swapping decisions. TSS/360 uses a table-driven algorithm[1] that schedules and dispatches transactions according to resource-use characteristics. The SRM in OS/VS2 Release 2 increases batch and TSO job control by permitting performance objectives to be specified in measurable terms, and as functions of system-wide and transaction-related variables. A related problem with a somewhat similar solution is discussed in Reference 2.

The second problem addressed by the SRM is that of optimizing the use of the four major system resources—Central Processing Unit (CPU), Input/Output (I/O) facilities, main storage, and the paging subsystem. Previously, most of the control associated with these resources was done by the selection of prespecified parameters. These parameters, however, were static, and, therefore, did not take into account the changes in system load that occurred during system operations. One method of dynamically carrying out performance-related adjustments is the I/O load balancing algorithm[3] that is used in OS/VS2 Release 1 and OS/VS1. Other means of system tuning are page replacement algorithms that have been implemented in all virtual systems,[4] and dynamic dispatching algorithms,[5] which were first available in OS/MVT. In OS/VS2 Release 2, dynamic control has been refined in these areas, and has been extended to include control of CPU utilization, main storage occupancy, and several aspects of the paging subsystem.

The potential for interference between individual performance algorithms constitutes the third problem addressed by the System Resources Manager (SRM). In earlier systems, little if any means existed for coordinating the decisions of independent resource utilization algorithms. The incorporation within a single component—the SRM—of many performance-related algorithms has made it possible to evaluate interactions and to avoid conflicts between them. One example where we have not achieved coordination among algorithms is the absence of interfaces between the Job Entry Subsystems (JES 2 or 3), and the scheduling done on behalf of the jobs selected by the SRM.

One of the OS/VS2 Release 2 design objectives is to provide each installation manager with greater control over the turnaround times of the installation's batch jobs and the response times of Time Sharing Option (TSO) commands without requiring installation personnel to write their own scheduling algorithm. The attainment of this objective requires installation personnel to use the System Resources Manager (SRM). The installation manager specifies the system response and turnaround time objectives as the rate at which processing resources are to be provided to the batch jobs and to the TSO commands. The SRM uses the resources of the system to achieve two distinct but potentially conflicting performance objectives. The first objective is concerned with service, i.e., to distribute processing resources among individual jobs and commands so as to satisfy the installation-specified objectives for response and turnaround times. Given the installation-specified performance objectives, the second objective is throughput optimization, i.e., keeping the use of CPU, main storage, and I/O resources within acceptable limits.

response and throughput conflicts

A scheduling algorithm that satisfies one of those two objectives may fail to satisfy the other. For example, a scheduling decision that has the effect of improving system throughput may seriously perturb the specified response time of certain TSO commands. In the design of OS/VS2 Release 2, we have recognized the requirement of minimizing and resolving such throughput and response-time conflicts. The coordination of the functioning of the distinct performance algorithms has been eased by including most performance algorithms in one subcomponent, the System Resource Manager (SRM). In practice, each performance algorithm that is concerned with making a decision evaluates the decision and makes a recommendation. The individual recommendations are combined by the SRM into one composite action that represents the best use of system resources at that time.

In OS/VS2 Release 2 it is feasible under certain circumstances to initiate more background (batch) and TSO (foreground) programs than can fit in main storage at any one time. Particularly appropriate for overinitiation are long-running jobs that do not overuse important serial resources such as tape drives. The result of this over-initiation strategy is that at any given time some of the ready-to-execute address spaces are in a swapped-out status. Periodically, the SRM selects from the entire set of ready-to-execute address spaces those that should then be in main storage. The selection process results in swapping if: (1) the rate at which processing resources are being given to individual address spaces is not consistent with installation-specified service goals; or (2) the utilization of system resources is not within acceptable limits. Thus the SRM uses swapping to control both response and throughput as discussed later in this article.

The use of swapping to dynamically adjust the execution program mix makes OS/VS2 Release 2 significantly different from OS/MVT and OS/VS2 Release 1, wherein decisions that affect the makeup of the executing program mix are irrevocable. Even if the mix becomes inefficient, it cannot be changed until one of the programs has been completed. In those cases, resource utilization algorithms attempt to minimize the effects of contention. However, the next job to be initiated could aggravate an existing problem.

In OS/VS2 Release 2, the SRM has the capability of undoing job selection decisions that result in an execution mix that does not make efficient use of system resources. For example, if two jobs contend for the same logical channel, one of these jobs is swapped out and is replaced by a previously swapped-out job that will not cause contention. The replacement decision makes use of information gathered by the SRM that describes the execution characteristics of partially executed programs.

This article next describes the specifications of response and turnaround goals by the installation manager. The article then describes swapping decisions made by the SRM that attempt to satisfy the two (potentially conflicting) performance objectives of system responsiveness and high throughput.

Specification of system service goals

workload manager

The Workload Manager is a subset of the SRM that enables installation personnel to control the distribution of processing resources among individual users of a system. By using this function, an installation manager can prescribe the relative response and turnaround times, and, to a lesser degree, the actual response and turnaround times afforded to individual users of a system. Clearly, all data processing systems have capacity and speed limitations. The function of the Workload Manager is to distribute the limited resources according to information supplied by the installation manager. Thus, it may be impossible to give an entire population of TSO users very good system response under very heavy workload conditions. However, it may be possible and desirable to give a small subset of priority users excellent response; give another subset of users moderate response; and give the remainder of the users acceptable response under heavy workload conditions.

transactions

An installation manager associates his response and turnaround objectives with *transactions*. A batch transaction is either a job or a jobstep. A TSO transaction is either a TSO command or subcommand. The SRM works at the address-space level and is unaware of work-element subsets within an address space. There is therefore a one-to-one correspondence between address spaces and transactions.

service rates

The manager specifies the installation response or turnaround objectives for transactions as the rate at which processing resources are to be supplied to the transactions. Processing rate rather than actual response time was chosen to express performance objectives because processing rate depends less on the amount of processing required by the transaction. The particular measure of processing rate that an installation uses is called the *service rate*, which is expressed in terms of *service units* per second. The amount of processing done by a transaction is measured in service units. Service units are a linear combination of the quantities of three basic resources used by a transaction as follows:

Service units = A(CPU units) + B(I/O units) + C(main storage units)

The service units used by a transaction over a short time interval are computed as follows:

- *CPU units* equal the amount of System Management Facilities (SMF) task time (in the transaction address space) accumulated since the start of the time interval divided by the time required by the CPU model to execute 10,000 instructions.

- *I/O units* equal the number of EXCPs counted by the SMF during the time interval for all data sets allocated to the transaction.

- *Main storage units* equal the number of real page frames allocated to a transaction, at the end of the time interval, times the accumulated CPU units; i.e., main storage units are a space-time product.

The service units accumulated by a transaction throughout its existence is the sum of all service units accumulated in the successive measurement intervals during the existence of the transaction. A, B, and C are installation-defined coefficients that allow the installation to change the way service is computed. The main storage term may tend to detract from the reproducibility of the measure, but it is included with the understanding that an installation manager may specify a zero coefficient for this term.

The composite nature of the service definition is prompted by the requirement that an installation be able to control the performance of transactions that use widely differing proportions of the three basic resources. Thus, even though an I/O-bound job makes relatively little use of the CPU, the fact that is does a great deal of I/O results in a significant accumulation of service. This in turn means that both a CPU-bound and an I/O-bound job can be controlled by the same service rate objective. The choice of the service definition coefficients can further aid in arriving at a definition of service that adequately handles many diverse kinds of transactions.

As mentioned earlier, it is sometimes feasible for an OS/VS2 Release 2 installation to run over-initiated so that not all transactions fit in main storage at any given time. Most transactions spend only part of their existence in main storage, accumulating service units. The rest of their existences is spent in a swapped-out, ready-to-execute state.

service versus absorption

The rate at which a transaction uses service units in main storage is called its *absorption rate*. The rate that an installation manager specifies for a transaction is the *service rate* for that transaction, i.e., the rate at which the transaction is to receive service units over its entire existence. The Workload Manager is free to swap a transaction in and out of main storage as long as the transaction receives its specified service rate. In fact, the Workload Manager uses swapping to control the service rate. When the service rate specified for a transaction exceeds its absorption rate, the transaction is not swapped out for service rate considerations. This implicitly identifies the transaction as being highly important.

installation performance specifications

The Workload Manager provides an installation with control over the service given to its batch and TSO jobs. It is the job of the installation manager to translate the various performance requirements of users of the installation into one or more Installation Performance Specifications (IPS). These specifications, or control data (which are included in the SYS1 · PARMLIB data set) permit the Workload Manager to schedule the availability of processing resources among individual transactions in accordance with the installation manager's specifications. An IPS is a mapping that associates a particular transaction with an intended service rate. An IPS mapping takes into consideration the system workload, the amount of service units already accumulated by the transaction, and the age of the transaction.

system workload level

Each ready-to-execute transaction, both in and out of main storage, generates a demand for service units. The magnitude of each individual demand is determined by the service rate specified for that transaction by the installation manager. The magnitude of the total demand for service units may exceed the rate at which the system can supply them. The system workload level, discussed more fully later in this article, is a measure of the degree to which the demands for service units exceeds the supply. When the system workload level is light, the demand can be entirely accommodated, and no transaction need be swapped out or have their swap-ins delayed. As the system workload level increases, delays becomes inevitable. In OS/VS2 Release 2, the installation manager specifies, by means of an IPS, how the delays are to be distributed among the transactions.

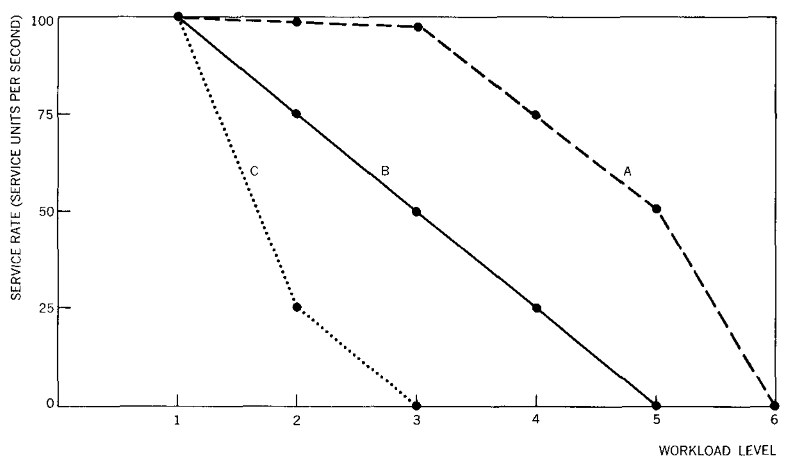
performance objective

An IPS contains a number of performance objectives, which are mappings between a fixed number of designated workload levels and the service rates that the installation manager wants transactions with those objectives to receive. At any given moment, a transaction is associated with a single performance objective. Transactions of equal importance are normally associated with the same performance objective.

The solid curve (B) in Figure 1 represents one system performance objective. The points on the curve are those specified by the installation manager. Although the specification includes only a fixed number of discrete levels, the workload manager algorithm uses linear interpolation to obtain intermediate values. Thus, as the workload level increases from the value 1 to the value 3, each transaction associated with this performance objective demands 50 service units per second. At workload level 4, each individual transaction demands 25 service units per second.

Figure 1 includes two additional performance objectives. Installation management might also include these objectives so

Figure 1 Service rate and workload for three performance groups



as to accommodate transactions with higher and lower priority than those associated with the moderate performance objective illustrated by the solid curve. Higher priorities are represented by the dashed curve, and lower priorities are represented by the dotted curve.

The lower priority transactions incur most of the delays when the workload level grows to the value 2. Moderate priority transactions are shown to have some increase in response time, and high priority transactions continue to have the same high performance they have at workload level 1.

When the workload level increases to the value 3, all low priority transactions (dotted curve) are delayed indefinitely (swapped out) until the workload level again drops below 3. Moderate priority transactions have their service rates cut in half, which is comparable to doubling their response times, so that the high priority transactions can continue to be handled without delay.

Finally, as the workload level grows to the value 5, the moderate priority transactions are completely sacrificed so that only high priority transactions continue to receive acceptable response times.

An installation manager decides how to subdivide his workload into performance groups. (Table 1 shows a workload divided into five performance groups). His performance goals for each individual group are specified by including a performance definition for it in his Installation Performance Specifications (IPS). The major function of the *performance group* definition is to allow the installation manager to associate a transaction with different performance objectives at different times or periods in its existence. Each job submitter and TSO user specifies part of this association by supplying a PERFORM parameter, which is a performance group number that uniquely identifies a single performance group definition. For the job submitter, PERFORM is a new JCL parameter. For the TSO user, PERFORM is a new LOGON command parameter. Note that it is the installation manager's responsibility to control the accesses and use of the various performance groups assigned by the users of the system. One possible control is for the manager to use of an SMF exit routine to scan all JCL parameters for compliance with the installation policies.

A performance group definition contains up to eight period definitions. One reason the workload manager monitors the progress of a transaction is to keep track of its progress through periods. A transaction proceeds successively from one period to the next, always starting with the first and ending up in whatever period it happens to be in when it completes. A transaction remains in a particular period for either a specified amount of elapsed time or until it accumulates a specified number of service units. When the condition that is specified in the associated period description is met, the transaction enters a new period.

Table 2 is a sample IPS containing the five performance group definitions described in Table 1. As an example, consider a transaction in performance group 3. In period 1, the transaction is associated with Performance Objective C in Figure 1, and which remains in until it ends or until it receives 200 service units.

Table 1 Example workload divided into five performance groups

Performance group	*User description*	*Performance description*
1	Miscellaneous batch	Turnaround expected only under light workload conditions
2	Low priority TSO	Poorer response than that given to critical transactions at moderate and heavy workload conditions
3	Standard TSO	Good to excellent response under light workload conditions
4	Medium duration batch	Transactions requiring less than 50K service units; two-hour turnaround under moderate workloads
5	Critical TSO and batch	Good to excellent response or turnaround under all workload conditions

(Period 1 has the effect of screening out nontrivial transactions.) If the transaction is not completed, it becomes associated with Performance Objective B for the second period or for the remainder of its existence.

A transaction associated with Performance Group 4 spends the first hour of its existence associated with the lowest performance objective, using whatever resources are available. At the end of that time, it proceeds to the second period, and it is associated for up to 50K service units with Performance Objective B. If not completed, it reverts to the lowest priority performance objective for the rest of its existence. The attempt is to provide two-hour turnaround as described in Table 1.

interval service value

Each period description contains an *Interval Service Value* (ISV) that specifies the minimum number of service units that an associated transaction is to receive each time it is swapped into main storage. (ISV values are not shown in Table 2.) The System Resources Manager (SRM) increases the interval service value as a function of the number of times a particular transaction has been in main storage. Specifying a relatively large ISV normally results in a reduction of the number of times a transaction is swapped and also a reduction of system overhead. At the same time, a large ISV reduces the precision with which the Workload Manager can control the service rate of a transaction, particularly in the case of short transactions. By using the ISV option, an installation manager can fold into the scheduling process his own evaluation of the tradeoffs between system throughput and system response.

To summerize, the mapping that permits a transaction to be associated with an intended service rate is a transaction associated with a Performance Group description by a JCL or LOGON PERFORM parameter, which is a specified performance group number. Depending on the age and/or the accumulated service units of a transaction, it belongs to period one, two, or three, etc. The current period description links the transaction to a perfor-

Table 2 Example performance group definitions

Performance	*Period 1*			*Period 2*			*Period 3*		
group	*Duration*	*Unit*	*Objective*	*Duration*	*Unit*	*Objective*	*Duration*	*Unit*	*Objective*
1	R		C						
2	200	S	B	R		C			
3	200	S	A	R		B			
4	3600	T	C	50K	S	B	R		C
5	B								

R – Remainder of transaction
S – Accumulated service
T – Elapsed time in seconds

mance objective, and, depending on the current system workload level, the performance objective associates the transaction with its intended service rate.

Since service rates and Performance Groups determine the progress of a transaction, computer operations departments that customarily distribute their costs to user departments may wish to use these items as a basis for determining charges. Using service rate as a basis for charge can lead to two problems. As previously defined, service rate depends on three constants (A, B, C) and three variables (CPU time, I/O operations, and processor storage occupancy). In most cases, the constants are set to maximize turnaround time and throughput, a situation that may bear no relationship to the economic value of the equipment involved. Moreover, the figure for storage occupancy may not be reproducible from run to run. Thus CPU and I/O figures taken directly from the System Management Facilities might provide the best basis for assessing charges.

In previous OS systems, many users distributed costs to jobs at a fixed rate, regardless of dispatching priority, or perhaps they added a shift surcharge. In OS/VS2 Release 2, it should be possible to use the performance group as a basis for surcharges, since service rates, and, hence, turnaround time, are established according to performance group. For example, for the groups described in Figure 1, it might be desirable to apply a higher rate to transactions in group five than to transactions in group one. Such a rate schedule would help prevent the situation wherein all users relate their jobs to the performance group with the shortest turnaround or response times.

Event driven swapping

Some swapping decisions are made as a result of the occurrences of specific events, and some are made as a result of time-dependent circumstances. We first discuss event-driven swapping, and then swapping decisions in greater detail.

A mix of programs currently swapped in may be changed as the result of an occurrence of an event that may or may not be related to the address space(s) that is subsequently swapped in or out of the mix. An example in which the event is related to the swapped address space is the occurrence of a TSO address space's entering a terminal wait state. This event causes the address space to be swapped out. An example in which an unrelated event causes an address space to be swapped out occurs when an extreme shortage of available page frames exists. Then the address space most ahead of schedule is swapped out. An algorithm for preventing a shortage of auxiliary storage causes the swapping out of an initiator that completes a job while a shortage of page slots exists. The address space that acquires page slots at the fastest rate is swapped out when a shortage of page slots is detected. A *page slot* is the place where a page resides in DASD. A *page frame* is the place where a page resides in main storage.

An ENQ/DEQ algorithm is an event-driven algorithm that is used when an address space is delayed because another address space is holding a resource that it needs. If the address space that holds the resource is swapped out, the ENQ/DEQ algorithm identifies it as a candidate for swapping in. If the address space that holds the resource is in main storage, it is made nonswappable until it has used a specified amount of CPU time. If the address space that holds the resource uses the specified amount of CPU time, and then does not release the needed resource, it is made swappable and is identified as a candidate for swapping out. An address space that is identified by the ENQ/DEQ algorithm is evaluated for swapping the next time the swapping analysis algorithm is called.

Timer-driven swapping and swapping analysis

The remainder of this article is concerned with timer-driven swapping, the main component of which is the swapping analysis algorithm. The swapping analysis algorithm evaluates the desirability of swapping a subset of the ready-to-execute address spaces. In the process of making each swapping decision, the swapping analysis algorithm calls on the following three algorithms to assess the impact of the swap: (1) the CPU load adjustment algorithm; (2) the I/O load balancing algorithm; and (3) the workload management algorithm. The swapping analysis algorithm uses the individual assessments to make each decision.

We first describe the method of determining the address spaces that the swapping analysis algorithm evaluates, and then briefly describe the swapping analysis algorithm. Address spaces that are evaluated for swapping are identified by one of the following four algorithms: (1) the CPU load adjustment algorithm; (2) the I/O load balancing algorithm; (3) the ENQ/DEQ algorithm; or (4) the workload management algorithm. These algorithms identify candidates for swapping.

candidates for swapping

The CPU load adjustment algorithm monitors the system CPU utilization and keeps track of address spaces that are heavy users of the CPU. Use of this algorithm precedes that of the swapping analysis algorithm. If CPU utilization is not within acceptable limits, the algorithm identifies two address spaces whose swaps can bring the CPU back within the acceptable range. If the CPU is underutilized, the algorithm identifies two swap-in candidates. If the CPU is overutilized, the algorithm identifies two swap-out candidates. The identification of heavy users of CPU as swapping candidates is one mechanism by which the CPU load adjustment algorithm tries to keep the system CPU utilization within predefined limits.

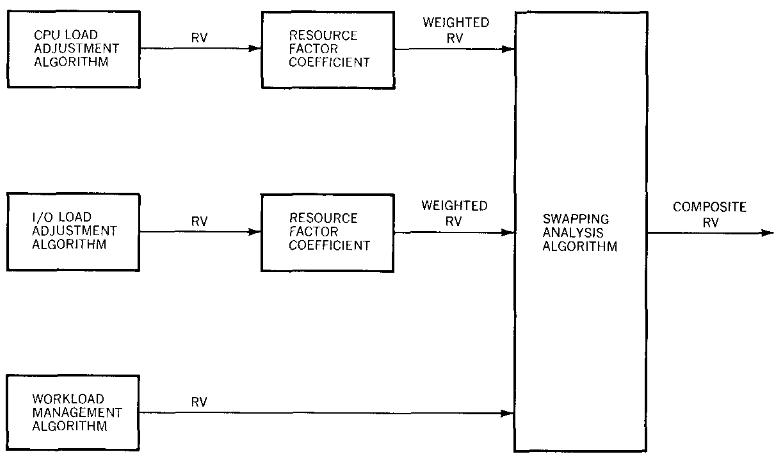
The I/O load balancing algorithm monitors the utilization of each logical channel. The algorithm keeps track of address spaces that have high EXCP rates. This algorithm also keeps track of those address space EXCP rates on the individual logical channels. (The I/O load balancing algorithm is called prior to using the swapping analysis algorithm.) If the usage of any logical channel is not within the predefined limits, the I/O load balancing algorithm identifies up to two address spaces that, when swapped, can bring the logical channel utilizations back within the predefined limits. The algorithm identifies address spaces with high EXCP rates on imbalanced logical channels. A swap-in candidate is identified if the utilization is too low. A swap-out candidate is identified if the utilization is too high. The identification of heavy users of the logical channels as swap candidates is one mechanism by which the I/O load balancing algorithm keeps the utilization of each logical channel within the predefined limits.

The workload management algorithm periodically reorders ready-to-execute address spaces on the basis of predictions of when the address spaces are to be swapped. The predictions are based on system workload level, measured absorption rates, specified service rates, and interval service values. The workload management algorithm implicity identifies address spaces as swap-in candidates. That is, when the swapping analysis algorithm is called, it treats those address spaces whose predicted swap-in times are next as candidates for swapping. The workload management algorithm, however, does not implicitly identify address spaces as candidates for swapping-out.

swapping out

The function of the swapping analysis algorithm is to decide whether to swap address spaces that have been identified as candidates for swapping. Swap-out candidates are evaluated first, since they may free up page frames that swap-in candidates may be able to use. The first step in the swapping decision is to compute a composite Recommendation Value (RV). A flow chart for the Recommendation Value computation is shown in Figure 2. The swapping analysis algorithm requests the CPU load adjustment algorithm, the I/O load balancing algorithm, and the workload management algorithm to evaluate each swapping candidate. Each algorithm computes a signed recommendation value (RV). The magnitude of the RV is proportional to the importance of the decision to the algorithm, and the sign indicates whether the algorithm favors the swap.

Figure 2 Logic flow of the swapping analysis algorithm


CPU LOAD ADJUSTMENT ALGORITHM
RV
RESOURCE FACTOR COEFFICIENT
WEIGHTED RV
I/O LOAD ADJUSTMENT ALGORITHM
RV
RESOURCE FACTOR COEFFICIENT
WEIGHTED RV
SWAPPING ANALYSIS ALGORITHM
COMPOSITE RV
WORKLOAD MANAGEMENT ALGORITHM
RV

The RVs produced by the CPU load adjustment algorithm and the I/O balancing algorithm are then weighted by installation-defined resource factor coefficients to produce weighted RVs. The swapping analysis algorithm algebraically adds the weighted RVs and the workload management algorithm RV to form the composite RV for the swap. A no-swap decision occurs when the composite RV does not exceed a swap-out threshold. A swap-out is inititated if the composite RV exceeds a swap-out threshold. In either case, the next candidate for a swap is evaluated.

swapping in

The first step in evaluating the swapping of a swap-in candidate is to compute a composite Recommendation Value (RV) for the address space. The computation is the same as for a swap-out candidate as outlined in Figure 2. A do-not-swap decision is made if the composite RV does not exceed a swap-in threshold. If the composite RV exceeds the swap-in threshold, then the swapping decision depends on the availability of page frames. Swapping-in is scheduled if the number of available page frames is sufficient to contain the swap-in candidate's working set. If the swap cannot be scheduled, an effort is made to find room for the swap-in candidate by combining its swapping in with one or more swap-outs. This action is called forming a swapping package.

forming a swapping package

The formation of a swapping package involves the following four steps in the swapping analysis algorithm:

1. Evaluate the desirability of swapping-out the address space most ahead of schedule. This means computing a composite RV for that address space by the procedure shown in Figure 2.
2. Compute a swapping package RV by algebraically adding the composite RV for the swap-in candidate and the composite RV for the swap-out candidate.

3. Compute a swapping package threshold by adding the swap-in and swap-out thresholds.
4. A do-not-swap decision is made if the swapping package RV does not exceed the swapping package threshold.
5. Initiate the swap if the swapping package RV exceeds the swapping package threshold, and if the swap-out yields a sufficient number of page frames to contain the working set of the swap-in candidate.
6. Otherwise try to enlarge the swapping package.

Note that a swap-in candidate with a large working set may be bypassed for swapping even though its composite RV exceeds the swap-in threshold. The RV of the workload management algorithm, however, tends to grow while the address space is swapped out. Eventually, the composite RV becomes large enough to force the swap-in candidate's working set to be accommodated in main storage.

Each of the following three sections describes an individual RV computation.

CPU load adjustment algorithm RV

The CPU load adjustment algorithm is used by the swapping analysis algorithm to evaluate each potential swap in terms of its impact on the system CPU utilization. A potential swap may have been identified as a swapping candidate by the CPU load adjustment algorithm. In any case, the algorithm computes a swapping RV in terms of a signed number.

The CPU RV is nonzero if the following two conditions exist: (1) the system CPU utilization is outside acceptable limits; and (2) a swapping candidate is a heavy user of the CPU, i.e., its CPU rate exceeds an SRM threshold. A nonzero RV is proportional to the degree of the resource imbalance and to the ability of the address space to correct it. The magnitude of the CPU RV is given by the following formula:

$$RV(CPU) = (D^2R)$$

where D is the distance the CPU utilization is from the acceptable range, and R is the CPU rate of the address space. For a swap-in candidate, the sign of the CPU RV is positive if the CPU is underutilized, and negative if the CPU is overutilized. For a swap-out candidate, the sign of the CPU RV is positive if the CPU is overutilized, and negative if the CPU is underutilized.

I/O load adjustment algorithm RV

The swapping analysis algorithm uses the I/O load adjustment algorithm to evaluate each potential swap in terms of its impact on the utilization of the logical channels. The potential swap may have been identified as a swapping candidate by the I/O load balancing algorithm. In any case, the I/O load balancing algorithm computes a swapping RV in the form of a signed number.

The following two conditions must exist for the algorithm to return a nonzero RV: (1) the address space EXCP rate exceeds an SRM threshold; and (2) the address space has data sets on imbalanced logical channels. A nonzero RV is proportional to the degree of the resource imbalance and to the ability of the address space to correct it.

The magnitude of the I/O RV is given by the following formula:

$$RV(I/O) = (D^2R)$$

where D is the distance that the logical channel utilization is from the acceptable range, and R is the address space EXCP rate on the imbalanced logical channel. If the address space has data sets on more than one imbalanced logical channel, then D and R are computed on the logical channel with the greatest imbalance.

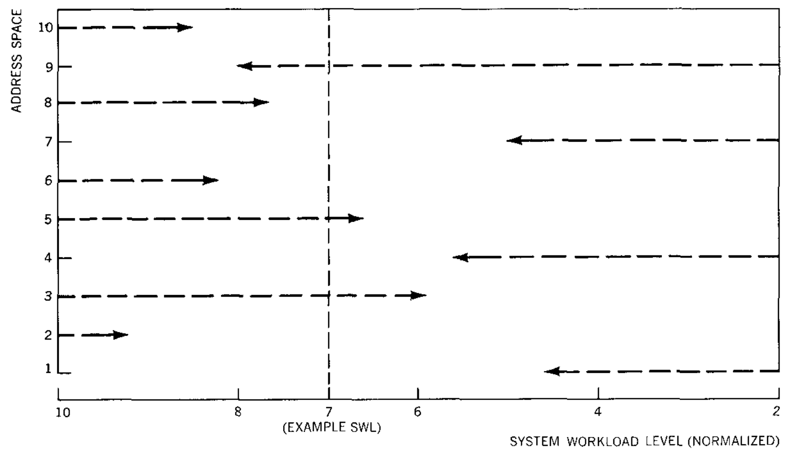
For a swap-in candidate the sign of the I/O RV is positive if the logical channel on which the RV is computed is underutilized; and the sign for the I/O RV is negative if the logical channel on which the RV is computed is overutilized. For a swap-out candidate, the sign of the RV is positive if the logical channel on which the RV is computed is overutilized, and the sign of the RV is negative if the logical channel on which the RV is computed is underutilized.

workload management algorithm RV

The workload management algorithm is used by the swapping analysis algorithm to evaluate each potential swap in terms of its effect on the rate at which the address space is supposed to receive service units. A potential swap may have been implicitly identified as a swap-in candidate by the workload management algorithm. In any case, the algorithm computes a swapping RV in the form of a signed number. The computation is based on the address space Normalized Workload Level and on the System Workload Level. The role of each of these factors in the RV computation is described first, and then the computation formula is presented.

Normalized Workload Level (NWL) is the measure used by the workload management algorithm to compare the progress of transactions that may be associated with different performance objectives. Comparing the progress of transactions involves: (1) computing the service rates that each of the transactions is currently receiving; and then (2) mapping (translating) the service rates into workload levels. For example, assume that the progress of three transactions has to be compared, and that each transaction is associated with a different Performance Objective, as shown in Figure 1. Further, assume that the service rates (in service units—SU—per second) of the transaction associated with Performance Objectives A, B, and C are 75, 50, and 25

Figure 3 System workload



service units per second respectively. Each Performance Objective curve can be considered as a mapping from service rates into workload levels. As can be seen from Figure 1, the service rate of the transaction associated with Performance Objective A can be mapped into a workload level of 4. This is called the *normalized workload level* of the transaction. Figure 1 also shows that the normalized workload levels of the transactions associated with Performance Objectives B and C are respectively equal to 3 and 2. The transaction associated with Performance Objective is least ahead of schedule since its normalized workload level exceeds those of the other two transactions.

With respect to the workload management algorithm, the execution mix is correct if it contains the address spaces with the highest normalized workload levels. Therefore, if the three types of transactions just described are each being evaluated for swapping out, the one associated with Performance Objective A would receive the lowest RV. Similarily if these transactions were being evaluated for swapping in, the one associated with Performance Objective A would receive the highest RV.

Before the workload management algorithm can compute an RV, it computes a value known as the System Workload Level (SWL). Assume a system with ten active address spaces. At a given time, the workload management algorithm computes the ten service rates and maps them into Normalized Workload Levels (NWL). The mapping concept is depicted in Figure 3. The direction of the arrows shows that the normalized workload level of a transaction in main storage tends to decrease over time, whereas the normalized workload level of a transaction not in main storage tends to increase over time. In any case, transaction 10, 8, 6, 5, 3, and 2 are in main storage.

Figure 3 shows that if only six address spaces can fit in main storage they should be 2, 10, 6, 8, 9, and 5. The mix is correct if address space 3 is replaced with address space 9. Note that address space 3 has the minimum NWL of the address spaces in main storage, whereas address space 9 is the swapped-out address space with the maximum NWL. If the NWL of address space 9 were less than that of address space 3, no swapping would be required, i.e., the execution mix would be correct with respect to the workload management algorithm.

The *System Workload Level* (SWL) is computed so that the workload management algorithm can quantify the desirability of swapping an address space. The SWL is defined as being at half the distance between the maximum NWL of the swapped-out address spaces and the minimum NWL of the swapped-in address spaces. Figure 3 shows that in the example described above the SWL is 7.

The workload management algorithm RV is a signed number, the magnitude of which indicates the importance of swapping to the workload management algorithm and the sign indicates whether the algorithm favors swapping. For a swapping-in candidate, the workload management algorithm RV is computed as follows:

$$RV(WMA) = (NWL - SWL)|NWL - SWL|$$

where $|\cdots|$ represents the "absolute value of . . ." The RVs for address spaces 9 and 7 in Figure 3 are respectively plus 1 and minus 4. For a swapping-out candidate, the workload management algorithm RV is given by the following formula:

$$RV(WMA) = (SWL - NWL)|SWL - NWL|$$

The RVs for address spaces 3 and 1 in Figure 3 are respectively plus 1 and minus 4.

Concluding remarks

One of the OS/VS2 Release 2 design objectives is to provide each installation with control over the responsiveness and turnaround times of its batch jobs and TSO commands, without requiring installation personnel to write their own scheduling algorithm. Installation personnel specify their response and turnaround objectives in terms of the rates at which processing resources are to be provided to individual jobs and commands. The system resources manager uses swapping to dynamically adjust the execution mix in order to achieve two distinct, but potentially conflicting, performance objectives. The first objective is service—to distribute processing resources among the batch jobs and TSO commands, according to installation specifications. The second performance objective is throughput—to optimize the use of CPU main storage and I/O resources, given the service objectives.

At the time this article is being written, OS/VS Release 2 has not been widely enough used to evaluate fully the efficacy of the System Resources Manager. However, a number of possible areas for further study can be postulated. The effect of adding another mechanism for controlling the rate at which processing resources are given to individual transactions could be usefully studied for improving performance in an undercommitted main storage environment. Also, consideration could be given to extending the System Resources Manager to handle work element subsets within an address space. Finally, ways of improving coordination between the System Resources Manager and Job Entry Subsystem (JES 2 and 3) could be studied.

CITED REFERENCES

1. W. J. Doherty, "Scheduling TSS/360 for responsiveness," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **37**, 97–111 (1970).
2. A. J. Bernstein and J. C. Sharp, "A policy-driven scheduler for a time-sharing system," *Communications of the ACM* **14**, 2, 74–79 (February 1971).
3. J. B. Page, *Controlling Input/Output Subsystem Utilization*, Technical Report TROO·2087, may be obtained from IBM Systems Development Division, P. O. Box 950, South Road, Poughkeepsie, New York 12601.
4. Y. Bard, "Experimental evaluation of system performance," *IBM Systems Journal* **12**, 3, 302–314 (1973).
5. K. D. Ryder, "A heuristic approach to task dispatching," *IBM Systems Journal* **9**, 3, 189–198 (1970).

GENERAL REFERENCES

S. Stimler, "Some criteria for time-sharing system performance," *Communications of the ACM* **12**, 1, 47–53 (January 1969).

W. J. Doherty, *The Effects of Adaptive Reflective Scheduling*, IBM Research Report RC-3672, may be obtained from the IBM T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, New York 10598.

P. J. Denning, "The working set model for program behavior," *Communications of the ACM* **11**, 5 (1968).

H. Hellerman, "Some principles of time-sharing scheduler strategies," *IBM Systems Journal* **8**, 2, 94 - 117 (1969).