Capacity planning, a major function of computer installation management, has the objective of determining cost-effective configurations to provide acceptable user service and system performance levels according to workload changes. The use of a performance predictive model is essential in the capacity planning process. This paper presents a research case study of the development of a performance model called PMOD, for the IBM OS/MVS operating system. The goal of the model is to predict user response times and system performance for different scheduling parameters, workloads, and configurations, with reasonably simple input requirements and fast run times. Both the validation and usage of the model for capacity planning and system tuning are discussed.

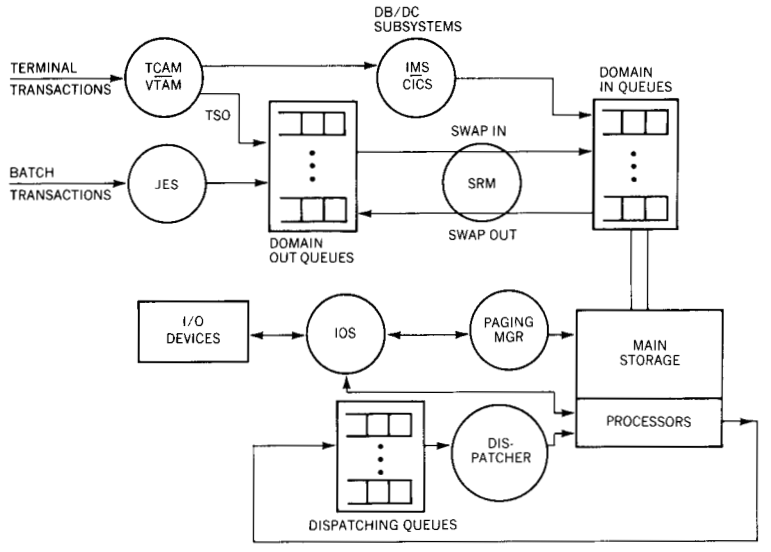
A performance model of MVS

by Willy W. Chiu and We-Min Chow

Capacity planning is an important function of computer installation management. The goal of capacity planning is to maintain acceptable user service and system performance levels with the most cost effective configuration. As new applications are added or increases in the number of users are anticipated, the installation management is most concerned with how the system should be reconfigured to meet the demand effectively.

The use of a performance model to predict user service is essential, since decisions on computer reconfigurations must be made ahead, due to the lead time required for equipment upgrading. An alternative is to perform experiments on similar configurations with benchmark workloads. This, however, can become extremely expensive. The growth in the workload may be estimated by employing statistical techniques on current or past measurements and a priori knowledge of increased load. Once workload characterization parameters are obtained, the best new configuration can be determined from exercising the computer performance model. The model should be capable of predicting accurately user response times and system performance for different scheduling parameters, workloads, and configurations. It is important that the input requirements are reasonably simple and model run times are fast.

Figure 1 MVS work flow



As computer systems become increasingly complex, developing a model to satisfy those requirements is more and more difficult. The main thrust of the paper is to present a case study of research into the development of a performance model that we call PMOD, for the IBM OS/VS2 Release 3.7 MVS (Multiple Virtual Storage) operating system and to discuss the model usage in capacity planning and system tuning. The model has a hybrid hierarchical structure that exploits the advantages of simulation and queuing theoretic and statistical techniques. The model is not designed for general use at the present time. Rather, it is intended to be a research vehicle and to demonstrate applications of modeling techniques to real-world problems.

A brief overview of how work flows through MVS is given in the next section, and then the model structure and validation results are described. OS/VS2 MVS is termed simply MVS throughout this paper. Finally applications of the model in the capacity planning and system tuning areas are discussed.

System description

Figure 1 is a simplified view of work flows through major components of MVS. For more detailed description of the MVS virtual storage paged operating system see References 1, 2, and 3. In the model development process, we have been particularly interested in factors that delay the progress of work units through the system. There are in general three types of work units, namely, batch jobs, TSO commands, and data base applications. The term

transaction is used in this paper to refer to any of these work units. The term *users* refers to those who generate transactions. We now describe how transactions are treated as they arrive at the system.

batch transactions

A batch transaction enters the JES component (Job Entry Subsystem) either from local or remote stations, and is queued by its job class parameters. The JES function is to accept batch jobs and to prepare and schedule their execution. When sufficient resources are available (such as tape drives, data sets, disk packs, etc.) to this transaction and its priority becomes the highest, a free initiator is assigned to direct the execution of this batch transaction. The total number of initiators at any given time is usually fixed. They are either assigned to batch jobs or they are idle. Each transaction, when in execution, occupies its own virtual storage Address Space (AS) and is under the control of the System Resources Manager (SRM). The terms users, transactions and ASs are sometimes used interchangeably when referring to work units.

TSO transactions

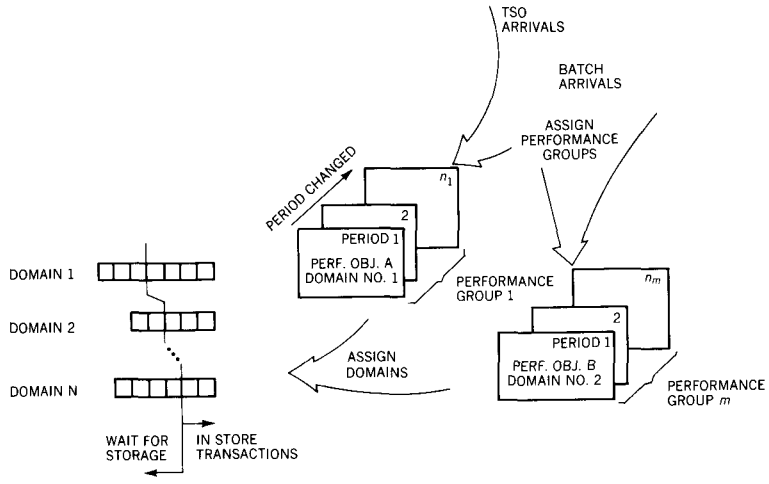
A TSO command (transaction), when entered by a user from a terminal into the system, is first received by the telecommunications component of MVS. Delays are normally caused by buffer space unavailability and message transmission time. The message is in turn passed to the TSO user's AS that is created when the user is logged on. The processing of this message in the AS is under the control of the SRM. Any outputs are also passed to the telecommunications component for transmission back to the user's terminal.

data base transactions

On-line data base application transactions (e.g., IMS) are handled similarly except that there is an extra layer, namely, the data base control. All transactions must pass through the data base control, which may have several priority queues. The data base control schedules an application AS (analogous to initiators) to process the transaction, based on its own priority scheduling. The application AS processes one transaction at a time, although there may be several application ASs. These types of transactions have not been modeled in the current version of PMOD.

The SRM is one of the more visible internal components of MVS. The reason is that many of the parameters that affect its operation are externally specifiable in the Installation Performance Specification (IPS). Its main task is to schedule the workload for execution according to the IPS and, in addition, to maintain resource utilizations within desired levels. The mechanism that the SRM uses to achieve this goal is that of swapping address spaces in and out of main storage, i.e., control of the multiprogramming set, since transactions can make progress only when they are executing on the CPU.

Figure 2 Relationship between workload level and service rate for three performance objectives



performance groups

The purpose of the IPS is to divide the workload into classes, called Performance Groups (PG), as shown in Figure 2. Usually TSO users and batch users are assigned to different PGs in order to give them separate treatment. A PG contains a number of periods. There is a set of parameters for each PG period that regulates service received by each transaction. A transaction starts off with the first period of its PG. If it does not complete in a specified period duration, it is demoted to the next period. The period duration is expressed in terms of either real time or attained service, such that short transactions may be favored over long transactions by giving different treatments to different periods.

domains

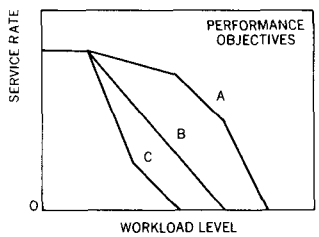
The in-store multiprogramming mix is further classified into so-called *domains*. Each domain is specified by a PG period parameter. Separating the multiprogramming mix into domains is quite a useful concept, since it allows the installation to relate logical workload classes to physical storage allocation groups. For example, TSO transactions may be controlled as a class if they fall into the same domains. It is, therefore, possible to allocate a fixed portion of processing capacity to any logical workload class. The ASs in the same domain compete with each other for a limited number of slots in the multiprogramming set. These limits are called domain target MPLs (Multiprogramming Levels), which can only be changed by the SRM. The total MPL is equal to the sum of the MPLs in each domain. Other domain parameters are the minimum and the maximum target values and the weights used by SRM to determine the importance of each domain. The reason for these parameters will become apparent after we discuss the paging storage allocation algorithm. The combination of aging trans-

actions through periods and assigning domains allows the installation to control effectively the level of service given to transactions of different resource demands.

priorities

The ASs in the same domain compete for a fixed number of MPL slots. When the number of ASs is greater than the number of slots in a given domain, some ASs are queued outside main storage (*out-queue*). When the priority of an AS on the out-queue is high enough, it can cause an *in-queue* AS to be swapped out and the high-priority AS to be swapped in to take its place. Priorities are calculated dynamically for each of the ASs, based on their performance objectives. The performance objectives are specified as a period parameter in the IPS, an example of which is shown in Figure 2. It is a mapping of service rate received to workload levels (a priority number). Service rate is calculated as the number of service units per second; service unit is a weighted sum of CPU time, number of I/O operations, and number of page-seconds (space-time product). In essence, priorities of in-queue ASs decrease with time, whereas those on the out-queue increase with time. The mechanism allows the installation to effect a variety of algorithms for sharing, such as round-robin and first-in-first-out.

Figure 3 Assignment of performance groups and domains



In addition to using service rates for priority calculation, the installation may also specify other factors to be taken into account. Since keeping resource utilizations within acceptable levels is an objective of the SRM, the SRM monitors those users who have caused an imbalance and those who may help balance resource usage. A composite priority is calculated to include the degree to which a particular AS influences resource utilization.

The CPU dispatching priorities of transactions are also specified as a period parameter. The period parameter is divided into the following groups: (1) fixed, (2) round-robin, and (3) mean-time-to-wait (shortest CPU burst time first). Such groups are arranged in order on the CPU dispatching queue, and the algorithm is preemptive resume.

real storage manager

One of the important system components is the Real Storage Manager, which allocates real storage page frames on demand in the MVS virtual storage environment. The Real Storage Manager, under the direction of the SRM, attempts to utilize real storage efficiently by keeping only those recently used program pages in storage. This capitalizes on the fact that programs are known to exhibit locality of storage references, and therefore more programs may be allowed to be resident. The mechanism has supported both a global Least Recently Used (LRU) algorithm and the Processing Time Window (PTW) algorithm, which is an approximation of the working-set algorithm in the various releases of MVS.

The storage management mechanism works in the following way, and is further discussed in Reference 2. Associated with each real

page frame are a page AS identification number (has no meaning for shared pages), a Reference Bit (RB), and an Unreferenced Interval Counter (UIC). The RB is turned on by hardware when the page contents are referenced. At periodic intervals, the status of each page's RB is examined by software. If RB = 1, both RB and UIC are reset to zero. If RB = 0, UIC is incremented by one. An inventory of free pages is kept. When it falls below some threshold because of a page request, the global LRU strategy requires a fixed number of pages with the largest UIC values to be deallocated from their ASs. In the case of the PTW, the free-page pool is replenished with pages having UICs greater than some criterion number (the window size). The periodic update interval for the case of global LRU is in real time; no distinctions are made among ASs. The PTW algorithm updates according to the elapsed virtual processing time (in units of CPU time) of each individual AS. Shared pages are processed differently, where there is a minimum number of allocated pages. The most recent release of MVS employs the global LRU strategy.

Since program behavior is dynamic in nature, the same MPL sometimes may cause unacceptably high page fault rates that reduce actual throughput. To prevent overcommitment of main storage, the SRM monitors the paging rates among main storage and secondary paging devices (such as drums or disks), and, in addition, the number of unprocessed page transfer requests and the average value of the highest overall UIC. If any one of these factors is not satisfactory, the target MPL of some domain is decreased by one. The actual domain MPL is then lowered to reflect this change. The domain parameters, the average number of ready users in each domain and the current domain MPLs are used to select the lowest priority domain for MPL reduction. On the other hand, if resources are under-utilized the reverse happens; the domain with the highest computed priority has an increase in its target MPL.

The MVS performance model

The first step of model development requires not only studying the system logic, but also performing experimentation and exploratory data analysis, such that important system components and parameters can be identified. The next important step is workload characterization, i.e., determining which workload parameters are needed as input to the model. As a result of these steps, we have arrived at a research model that captures the salient features of OS/VS2 MVS and its workload. (Details of these steps are the subject of a paper to be published elsewhere.)

Queuing theoretic models have been quite successfully applied to study computer system performance. There are, however, at

least the following three difficulties: (1) the need to analyze complex workload scheduling algorithms; (2) the problems of modeling storage contention (swapping); and (3) the inability to account for the effects of different page allocation algorithms in a total system model. Although some progress has been made recently in this area, adequate general results are still unavailable.

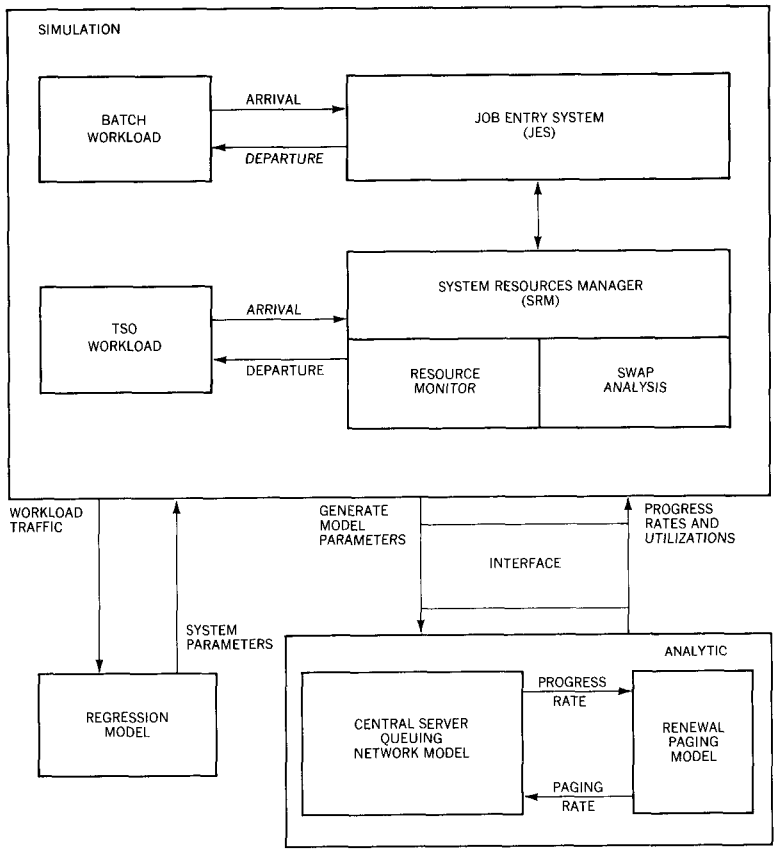
To overcome these difficulties, practitioners have frequently turned to detailed simulation models. Due to the level of detail modeled, simulation development and running costs are often relatively high. Measurements from MVS have indicated that frequencies of dispatches, I/O interruptions, and page exception events are of the order of several hundreds per second. Events that cause multiprogramming level changes, however, are usually much less frequent. A simulator usually runs very slowly if the high-frequency events are explicitly modeled. The approach we take is to subsume those high-frequency events in a computationally efficient analytic part and to simulate the relatively infrequent but complicated workload scheduling events. Thus the total model run time can be shortened and benefits of both techniques can be exploited.

Figure 4 illustrates the major PMOD components. The model has a hybrid hierarchical structure that combines simulation, queuing theoretic, and statistical techniques. The simulation part includes workload generation, Job Entry Subsystem (JES), System Resources Manager (SRM), and the interface to the analytic part. The analytic part consists of a multiclass Central Server queuing network Model (CSM) and a Renewal Paging Model (RPM). Parameter values are supplied by the simulation part to the analytic part, which returns processing rates of each storage resident Address Space (AS) and resource utilizations. To further simplify the model, statistical techniques (such as regression analysis) are used to estimate system parameters for swapping, paging, I/O, terminal communications, and job spooling overheads.

workloads

The current version of the MVS performance model allows two types of users, batch and TSO, and a number of system jobs, e.g., JES and TCAM/VTAM. Each user type may have several classes, and these classes are used to specify resource requirements, e.g., long and short, I/O or CPU bound, etc. The system jobs are never swappable and each occupies a permanent AS. Their resource requirements, estimated from regression techniques, reflect the current estimation of activity. The JES requirements depend on the batch arrival rate and the amount of spooling required. The TCAM/VTAM requirements are functions of TSO transaction arrival rates. Owing to a lack of efficient solutions, the contention of software locks is not considered in this model.

Figure 4 Components of PMOD



For each type of transaction the following attributes are needed:

- Class selection probability.
- Performance group number.
- Interarrival time distribution (think time for TSO).
- Unreferenced Interval Counter (UIC) distribution or inter-reference time distribution (for program page fault rate estimation).
- CPU time distribution.
- Number of I/O operations to each data set and number of bytes per transfer.
- Number of terminal I/Os per TSO transaction.
- Number of job steps and spool data sets for batch.

installation performance specifications

For each performance group the following parameters are specified:

- Number of periods.
- Domain number of each period.
- CPU dispatching priority for each period.

- Performance objective number of each period.
- Duration in service units for each period.
- Interval Service Value (ISV) for each period.

The *interval service value* is the number of service units that must be accumulated by a transaction before its swap-out is allowed. The duration parameter is in service units only. The CPU dispatching priority is given as a number up to 255.

The number of domains, maximum and minimum Multiprogramming Levels (MPLs), and the weights for domain priority calculation are also required as input. A *swap threshold parameter* is used to prevent unnecessary swaps when the differential of priority between in-queue AS and out-queue AS is not large enough. The time interval between invocations of an MPL adjusting algorithm is to be specified in time units of the model.

The JES queuing discipline is First-In-First-Out (FIFO), but it may easily be changed to any complex structure necessary.

system parameters

The system functions such as dispatching, I/O interruption handling, page fault processing, and swapping are modeled as additional CPU times to the transactions and are specified externally. The following are the parameters of the particular system configuration:

- CPU model (or speed if all CPU values are in times).
- Main storage size available for user allocation (in pages), excluding nucleus and long-term fixed pages.
- Number of channels.
- Number and speed of direct access devices (disks and drums).
- Designation of devices for paging and for swapping.
- User data set locations.
- Number of TSO terminals and batch initiators to be started.

simulation component

The simulation language used with the simulation component of Figure 4 is SIMPL/1.[4] The workload generator introduces TSO and batch transactions into the system, according to the interarrival time distributions. Each transaction is assigned attributes as discussed. The batch transactions join the JES queues and wait for the assignment of available initiators. Next the batch transaction joins one of the domain out-queues as depicted by its performance group. The JES processing time requirements are not reflected on a per-job basis.

TSO transactions join the designated domain out-queues directly. Transmission delays are modeled as part of the think time, and should be taken into account when the think time parameter is specified. The SRM is invoked at the transaction arrival time. The SRM consists of two main modules, swap analysis and resource

monitor. The swap analysis module is invoked by arrivals, completions, or by the resource monitor, and performs the following functions:

- Updates attained service of each user, based on current processing rates (as supplied by the interface).
- Computes each user's priority, based on attained service and time in the system (using the performance objectives).
- Checks for period change and enters the change into a new domain if necessary.
- Swaps users into and out of main storage because of one of the following conditions: (1) a transaction completes with another out-queue user of the same domain taking its place; or (2) a user on the out-queue is to replace an in-queue user due to priority difference; or (3) target MPL is different from the current domain MPL.
- Adds CPU time and I/O required for swapping to every swap-in candidate's total resource requirements.

The resource monitor is invoked at regular time intervals (of the order of tens of seconds). It performs the following operations:

- Computes a *Contention Index (CI)* of each domain using the formula

$$CI = \frac{(\text{in-queue length} + \text{out-queue length}) \times \text{weight}}{\max(1, \text{current MPL})}$$

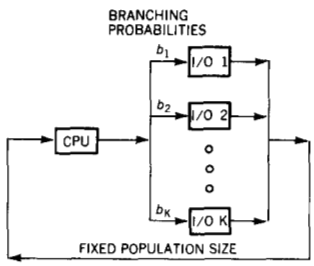
- Increases domain target MPL with the highest contention index by 1 if CPU utilization $< v_1$ and paging device utilization $< v_2$, where v_1 and v_2 are threshold values.
- Decreases domain target MPL with the lowest contention index by 1 if CPU utilization $> v_1$ or paging device utilization $> v_2$.
- If resource utilizations do not cause target MPL changes, then the resources monitor equalizes the highest and lowest contention index domains by changing their target MPLs each by 1, but in the opposite direction.
- Invokes Swap Analysis Module to swap users.
- Collects statistics on workload arrival characteristic for the regression submodels.

Note that the model uses paging device utilization instead of paging rates and unprocessed page-in request queue lengths, since this is simpler and has been found to have the same effect. When a target MPL is changed, the minimum and maximum MPL parameters that are associated with the domain are checked and violations are avoided by choosing the next domain.

The interface module is the link between the simulation component and the analytic part, and has the purpose of building parameters to drive the analytic part. In return, the interface mod-

ule obtains the effective execution rates, R_i, for the ith resident user, and hardware utilizations. Each user has the following attributes:

Figure 5 Central server queuing network model



- Total CPU time including time for swapping, $CP_i + SWCP$, where CP_i is the transaction CPU time requirement and $SWCP$ is CPU time for swapping, which is added to CP_i each time there is a swap.
- Total I/Os, including data set accesses (FIO_i) and swapping I/Os ($SWIO$), as well as the distribution of these I/Os to devices, expressed as I/O device branching probabilities. These probabilities are estimated from data set placements.
- CPU dispatching priority.
- Unreferenced Interval Count (UIC) distributions expressed as hyperexponential parameters.

System users are characterized by linear regression techniques. The JES CPU and I/O usage rates are functions of job arrivals per second, spool data sets per job, and average size of spool data sets. TCAM and VTAM resource usages are obtained in a similar fashion, as a function of TSO traffic rates. These values are periodically supplied by the resource monitor module.

Based upon the progress rates of each resident user, completion times are estimated. Let $Q_i(t)$ be the remaining CPU time requested by transaction i at time t, and R_i be its present progress rate. Then the expected completion time will be the following:

$$C_i = t + \frac{Q_i(t)}{R_i}.$$

When the completion time is reached a swap analysis module is invoked. Each time there is a change in the mix because of swapping or completion, the Interface part is called, which then recomputes the next nearest completion time.

analytic component

The analytic part of the performance model of MVS—which we call PMOD—is a multiclass Central Server queuing network Model (CSM) of CPU and I/O contention coupled with a Renewal Paging Model (RPM) of program behavior and page allocation. The relation among paging, workload characteristics, and system configuration can, therefore, be accounted for in the total system model. The CSM results are obtained using QNET4[5] for processor sharing CPU dispatching discipline, and using an iterative technique[6] for priority preemptive resume discipline.

A typical queuing network model is shown in Figure 5. The CPU time between I/Os (the execution interval E_i) is defined as the inverse of the sum of file I/O and page fault I/O rates. The former is given as an attribute of transactions and the latter is computed by employing the Renewal Paging Model (RPM). The I/O server discipline is FIFO. Service times include time spent through the chan-

nels, control units, and devices. The technique described in Reference 7 can be used to estimate the effects of channel contention.

The RPM is a program behavior model in a periodic aging environment. The global LRU case only is discussed in this paper. [The processing Time Window (PTW) algorithm case is presented in Reference 8.] The RPM accepts as input for each user its Unreferenced Interval Counter (UIC) distribution of hyperexponential form, its program size, and its execution rate. The output is each user's page fault rate, for a given total allocatable real storage size m. The RPM models the program pages as distributed among different discretized age groups (i.e., the UIC values). A program page then makes transitions from one age group to another. If, in a discretized interval h, a page is referenced, its UIC is reset to zero (as mentioned in the system description section), i.e., the page returns to the zero age group. Otherwise, its UIC is increased by one and demoted to the next older age group. For the sum of all user programs, size requirements may exceed the m allocatable real pages. However, those m pages that have youngest ages are allowed to remain in real storage. These m pages are a mixture of all those programs resident in real storage. Consequently, the intensity of references to the $m + 1$, $m + 2$, $m + 3$, etc. youngest pages outside m is the page fault rate. The references made to the same page are assumed to form a renewal process. Average page fault counts per h interval can be estimated. For the global LRU case, programs with the same intrinsic page referencing behavior could execute at uneven rates, because of differences either in priority or I/O delays. Their program pages, therefore, distribute differently among the UIC age groups and may result in differing page fault rates. The differing page fault rates, however, do, in turn, affect the execution rates. Iteration schemes have been tried, but to shorten computation we find the following procedure to be quite acceptable:

- Compute the initial user execution rates, R_i, by running the Central Server Model (CSM) under the assumption that the users' page fault rates, G_i, remain the same as those obtained at the last time the RPM was invoked. If a user's program is just swapped in, however, its page fault rate is set equal to zero. The execution interval is given by

$$E_i = IOSYS + \frac{CP_i + SWCP}{PIO_i + FIO_i + SWIO} + PFSYS\left(\frac{PIO_i}{PIO_i + FIO_i + SWIO}\right),$$

  where $IOSYS$ is the system time required per I/O, PIO_i is the number of page I/Os based on G_i, and $PFSYS$ is the system time required per page I/O.

- Normalize intrinsic UIC distributions to the real-time domain. An age of T real-time units is approximately equal to $R_i T$ CPU time units. For a given intrinsic age distribution $F(t)$ in CPU time units, the age distribution in the real time domain is given by $H(t) = F(R_i t)$.
- Compute the page fault rate G_i by RPM. The procedure is similar to that of Reference 8, where the RPM addresses the single program case. In the present instance, however, there are several programs. It is required to obtain the age distribution of those pages outside main storage and their distribution among the programs. Consequently, G_i, given by the reference probabilities of those pages outside main storage, can be obtained. An overview of the RPM is provided in the Appendix.
- Run CSM with G_i to obtain the final R_i.
- The R_i are then converted to effective execution rates of user CPU times by removing the system times

$$R_i = R_i\left[1 - \frac{IOSYS + (PF_i \times PFSYS) + SW_i}{E_i}\right],$$

where

$$PF_i = \frac{PIO_i}{PIO_i + FIO_i + SWIO},$$

and

$$SW_i = \frac{SWCP}{PIO_i + FIO_i + SWIO}.$$

run modes

PMOD can be executed in two modes, a distribution-driven mode or a trace-driven mode. The distribution-driven mode has workload attributes that have been generated randomly from specified distributions. The trace-driven mode allows these workload attributes and their arrival times to be supplied from an external workload file. The workload file may be created from measurements such that actual peculiar arrival patterns can be used. Important run-time statistics are also provided at the end of each run, including response-time distributions and throughput by workload class or domains, various resource utilizations, queue lengths, time spent on queues, and the number of swaps by type.

Typical examples show that PMOD execution time to simulated time ratios are about one, which is thirty to one hundred times better than for purely simulation models. The execution time goes up with increasing load, but is still within our requirements for a practical model.

Model validation

To thoroughly validate such a model requires tests in a large number of environments. In this section we report the results of com-

paring three sets of measurements against model prediction, although more tests are underway. Benchmark experiments have been conducted on a System/370 Model 158 with three megabytes of storage under three different loads. The experiments do not show the highest possible stresses on the system, especially that of the paging subsystem, but they should be adequate for an initial test of the model accuracy. The workload of the first two experiments consists of TSO users only, with fifteen and thirty users logged on. The third experiment has thirty TSO users and five batch initiators active. The TSO transactions are mainly COBOL compiles, editing, and data set manipulation loads. The batch jobs are a mix of commercial and scientific loads. Measurement intervals are approximately half an hour each.

The performance model, PMOD, is validated in a completely deterministic trace-driven mode, so that we do not need to replicate model executions with several random number seeds. Thus uncertainties of empirical distributions estimations can also be avoided. A software monitor that is both event- and timer-driven has been developed for the modeling project. It is capable of capturing detailed transaction characteristics and system performance variables. A transaction trace file is produced from reduction of the measurement data. Each entry of the file contains the transaction's arrival time, CPU time, I/O counts per data set, and number of terminal I/Os. The Unreferenced Interval Counter (UIC) distributions are obtained for all transactions instead of for each transaction, since these transactions are usually short and do not have enough samples to estimate the distribution.

Measurements have shown that some of the CPU time consumed is not charged to any address space and that it can be more than five percent in many cases. Correlation studies have shown this uncharged CPU time to be related to system activities such as page fault rates, I/O rates, dispatching, and swapping rates. Uncharged CPU time used per system function is estimated by linear regression techniques. Stepwise regression techniques have also been used to characterize JES and TCAM/VTAM resource usage rates from a number of input variables. A detailed description will be the subject of a forthcoming paper.

The system parameter values such as the ones discussed in the last paragraph are estimated from a fourth experiment, so that the model predictions will not be biased by the input. The Installation Performance Specifications (IPS) specify three domains. Domain 1 is for first period TSO (short transactions). Domain 2 is for the second and last TSO period. Domain 3 is the only domain for batch. The TSO response time values do not include time spent in transmission or in TCAM processing, but rather the interval from the time the transaction enters the System Resource Manager (SRM) to the time when it completes and is swapped out of main

Table 1 Comparison of measurement and model output

BATCH	TSO	Total CPU utilization (percentage)		User CPU utilization (percentage)		Response time in seconds (mean/standard deviation)						CPU time per TSO transaction in seconds (mean/ standard deviation)	TSO transactions per second
						Domain 1		Domain 2		Domain 3			
		Measured	Modeled	Measured	Modeled	Measured	Modeled	Measured	Modeled	Measured	Modeled		
0	15	43.0	42.9	17.6	20.0	1.31 1.36	1.32 1.42	18.00 20.36	19.46 21.42	—	—	0.30 0.93	0.59
0	30	64.1	61.0	23.6	22.5	1.28 1.43	1.12 1.22	12.30 14.24	14.38 16.52	—	—	0.15 0.45	1.18
5	30	99.5	99.9	56.0	58.5	1.00 1.08	1.14 0.95	46.29 89.40	51.56 80.78	28.08 79.05	28.16 81.55	0.08 0.44	1.29

Figure 6 TSO response time density function for 15 TSO users case

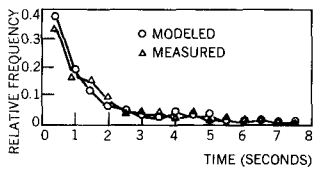


Figure 7 TSO response time density function for 30 TSO users case

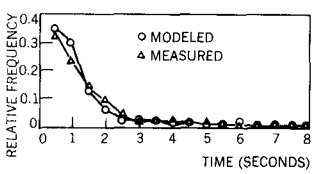
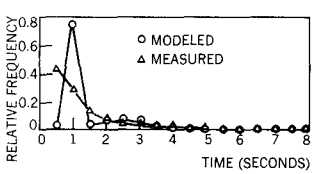


Figure 8 TSO response density function for 30 TSO users case and 5 batch initiators case

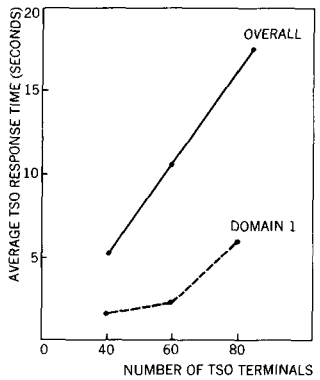


storage. Batch response times do not include time spent in JES; they include the time from job selection of an initiator to job termination.

Comparisons of the measurements with the model predictions are shown in Table 1. The user CPU utilization column indicates CPU times consumed by TSO and batch only, and excludes times taken up by system functions. The reason for the rather low user CPU utilizations compared to the total is that the total includes the measured CPU overhead, which is explicitly taken into account in the model (over ten percent). Both the mean and the standard deviation of response times are given in the table. In the fifteen-TSO case, for example, the mean and the standard deviation of the measurement from Domain 1 are respectively 1.31 and 1.36 seconds. The model predictions have about five to fifteen percent error in all categories. Figures 6 to 8 show comparisons of TSO response time distributions. The model's prediction curves in general follow the measurement curves closely, except for the first few points on Figure 8 (30 TSO and 5 batch case). The interference of batch jobs on TSO response is overestimated, since this particular model run employs QNET4 for the Central Server Model (CSM) where equal CPU dispatching priority is assumed (i.e., processor sharing CPU dispatching). To address this problem, an iterative queuing analysis technique discussed in Reference 6 is being implemented for the case of priority CPU dispatching. Further testing is underway. The average CPU times per transaction were found to vary from experiment to experiment, since different segments of the TSO script were exercised in different experiments. This has caused Domain-2 response time for the thirty-TSO case to be lower than that of the fifteen-TSO case. Longer measurement runs probably result in a more even overall workload characteristic. However, we do not need the same precise loads in each case to test the effectiveness of the model.

Validation for the distribution-driven mode requires more effort. Empirical workload data must be analyzed and fitted with theoretical distributions, such as TSO think time, batch interarrival time, CPU time, number of I/Os, etc. The next problem is to determine appropriate model run lengths and the number of replications with different random number seeds, so that statistical stability of the model results can be achieved. Usually statistical stability is quantified by confidence interval at a given level. Confidence interval estimation methods can be found in simulation textbooks, such as Reference 9.

Figure 9 Average response time under different TSO workloads

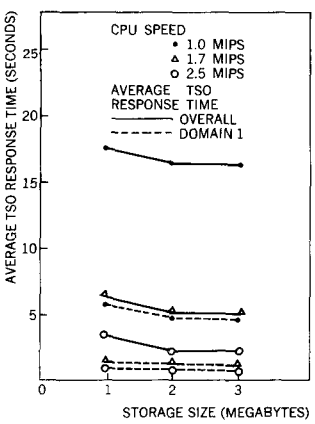


Capacity planning

Capacity planning is a key step of computer resources management. The goal of capacity planning is to define and maintain acceptable user service and system performance levels with the most cost-effective configuration and equitable scheduling policy. Since the future workload is usually not deterministic at the capacity planning stage, the distribution-driven model is essential for this purpose. The process involves the following steps:

- Measure and analyze workload resource usage patterns and current service levels (response time and throughput).
- Measure and analyze system performance, i.e., the overheads incurred in supporting the service levels.
- Project future workload levels and requirements.
- Use the projected workload parameters as input to the model to obtain predicted performance. If the performance is not adequate, try several configurations with the model and select the best one.
- Use the model experiment with scheduling policies, data set placements, and hardware rearrangements for performance optimization.

Figure 10 Average TSO response time for 80 terminals with different storage sizes and CPU speeds



The model plays an important role in increasing the understanding of the interrelationships and interactions among the various workload types. From that understanding one can make intelligent tradeoffs. It is necessary to make projections on workload increases and plan computer reconfigurations, since long lead times may be required for equipment upgrading. After a change in the configuration is made, performance optimization is required. For example, when a workload increase causes an upgrading in main storage size, the IPS must be changed accordingly to allow a higher multiprogramming level.

Figures 9 and 10 delineate a family of performance curves for different configurations over a range of workload levels. Figure 9 shows average TSO response times for all transactions and for Domain 1 completions (short transactions) over 40-, 60-, and 80-

terminal loads. This example is for the case of a one-million-instructions-per-second CPU, one million bytes of user pages (i.e., total storage from which has been subtracted those pages that are required by the system and common areas), and 13 disk drives spread over 2 channels. The workload is similar to that used in the validation experiments, except that distributions are assumed (average 270 000 instructions per transaction, exponentially distributed). No batch loads are included in the example.

Suppose that the current load is at 40 terminals and is expected to increase to 80 terminals of the same workload characteristics. For the heavier load, even short transactions have over 5 seconds response time, and 17.5 seconds overall response. If this is not acceptable, one may increase storage size or the CPU speed. Figure 10 shows the results of running the model for different storage sizes (1, 2, and 3 million bytes) and CPU speeds (1, 1.7, and 2.5 million instructions per second). These CPU speeds are approximately equivalent to some System/370 CPUs.

Suppose we are required to have the same response characteristics as the 40-terminal case. Clearly the one-million-byte, 1.7-million-instruction-per-second configuration satisfies the requirement. We also see that increasing storage size for a one-million-instruction-per-second CPU yields little improvement. This contrived example demonstrates the use of the model for capacity planning. In a real application, equipment costs must be taken into consideration.

Concluding remarks

In this paper we have presented a case study of the development of a performance model for the IBM OS/VS2 MVS operating system. The purpose of the case study is to demonstrate applications of modeling techniques to real-world problems such as capacity planning and system tuning. The model is also to be used as a research vehicle for studying advanced architectural enhancements. Although further tests are being conducted with actual workloads, the initial validation results have been encouraging.

The running time of the model is intermediate between purely simulation and totally analytic models. The input requirements are closer to those of analytic models, but gross and distorting assumptions often used need not be made here with simulation models. Key internal system features are represented without the high cost of running detailed simulation models. The high cost of the data processing installation and the gravity of capacity planning and design decisions often justify the use of a more accurate model.

Measurements obtained via System Management Facilities (SMF)[10] and the Resource Measurement Facility (RME)[11] can be used as input to the present model. These tools provide workload resource use parameters as well as I/O service times and probabilities, with the exception of storage reference behavior parameters.

Despite the level of detail of PMOD, some of the performance bottlenecks in real situations are not predicted, e.g., the SMF data set enqueue problem. To handle this type of performance prediction, the model will have to include the contention of software locks, which may cause the model to be overly cumbersome. A more efficient method of treating such lock contentions is under investigation. Other extensions to the model are expected to include data base applications (IMS and CICS) and channel/control-unit contention modeling.

ACKNOWLEDGMENT

The authors express their gratitude for the assistance of Messrs. M. C. Zwilling, S. J. Loftesness, T. L. Moeller, and W. H. Tetzlaff during the course of this work.

Appendix: Page fault rate calculation in the global LRU case

In the global LRU case, the age of page j, X_{ij}, of program i is a random variable subject to the distribution $H_i(t)$. Arrange the X_{ij} in an ascending order and call the nth smallest random variable Y_n. Then these pages with their X_{ij} less than or equal to Y_m are considered to be residing in main storage of size m. In other words, a page fault occurs whenever a reference is made to a page with its X_{ij} greater than Y_m. The computation procedure is given as follows:

- From $H_i(t)$ obtain discretized UIC density $A_i(y)$, for program i (of program size N_i), where $y = 0, 1, 2, \cdots$.
- Compute $B_n(y)$, the probability density of Y_n, for all n.
- Compute the conditional probability $C_i(n, y)$ that a page with the nth smallest UIC, Y_n, is in program i, given that $Y_n = y$. This probability is proportional to N_i and $A_i(y)$.
- Compute the conditional probability that a page of UIC X_{ij} is referenced, given that $X_{ij} = Y_n$ and $Y_n = y$. Denote this probability by $D_i(y)$.
- The page fault rate of program i is obtained by summing the product of $D_i(y)$, $C_i(n, y)$ and $B_n(y)$ over all possible y and n such that $n > m$.

CITED REFERENCES

1. *OS/VS2 System Logic Library*, Volumes 1–7, SY28-0713 to SY28-0720, IBM Corporation, Data Processing Division, White Plains, New York 10604.

2. A. L. Scherr, "Functional structure of IBM virtual storage operating systems, Part II: OS/VS2-2 concepts and philosophies," *IBM Systems Journal* **12,** No. 4, 382–400 (1973).
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. *SIMPL/1 Program Reference Manual*, SH19-05060, IBM Corporation, Data Processing Division, White Plains, New York 10604.
5. M. Reiser, "Interactive modeling of computer systems," *IBM Systems Journal* **15,** No. 4, 309–327 (1976).
6. K. M. Chandy, U. Herzog, and L. Woo, "Parametric Analysis of Queuing Networks," *IBM Journal of Research and Development* **19,** No. 1, 36–42 (1975).
7. P. H. Seaman, R. A. Lind, and T. L. Wilson, "On teleprocessing system design; Part IV: An analysis of auxiliary storage activity," *IBM Systems Journal* **5,** No. 3, 158–170 (1966).
8. W. Chow and W. Chiu, "A program behavior model for paging systems," *Proceedings of the Symposium on Computer Performance Modeling, Measurement, and Evaluation*, edited by K. M. Chandy and M. Reiser, North-Holland Publishing Company, Amsterdam (1977).
9. G. S. Fishman, *Concepts and Methods in Discrete Event Simulation*, John Wiley & Sons, Inc., New York (1972).
10. *OS/VS System Management Facilities (SMF)*, GC35-0004, IBM Corporation, Data Processing Division, White Plains, New York 10604.
11. *OS/VS MVS Resource Measurement Facility (RMF) Reference and User's Guide*, SC28-0740, IBM Corporation, Data Processing Division, White Plains, New York 10604.

Reprint Order No. G321-5083.