The performance of MVS (Multiple Virtual Storage) systems can be predicted for changes in workload and environment by an IBM marketing aid informally called SCAPE (for System Capacity and Performance Evaluation). Written in FORTRAN, the programs use simple queuing formulas with empirical modifications. Response times for complex workloads (IMS, CICS, TSO, and batch) through the CPU and auxiliary storage are expressed as functions of application loads and other parameters that define the system's environment. SCAPE can predict the effect on performance of different CPU models, larger memory, additional channels, additional direct-access storage, larger block sizes, and alternate workload projections.

System capacity and performance evaluation

by D. C. Schiller

For any capacity planning exercise, the analyst must first choose the type of methodology to be used. Available techniques range from simple rules of thumb to complex and expensive benchmarking. Not all are strictly performance prediction models, but all serve the same function in capacity planning, in that they help the user evaluate his present configuration as well as possible alternative configurations.

Among the simplest methodologies in general use is USAGE,[1] which is based on utilization of the CPU. The performance criterion in USAGE is not response time, as in the other techniques, but total CPU hours of utilization. The first, or calibration, phase of a USAGE analysis is to adjust CPU utilization by individual applications so they sum to the measured total CPU utilization. Future utilizations, then, are linear functions of future workloads. If the only capacity bottlenecks in a system are expected to be in its CPU, then USAGE is all that is required for capacity planning. However, if bottlenecks are expected in other parts of the system (such as memory, channels, tape drives, or direct-access storage), then another step may have to be considered.

The next more expensive methodology is the use of models based on analysis. Various IBM marketing aids[2] fall in this category.

They use *closed* models, which at each workload value determine first the value and then the system response time. The theory behind closed models is covered well in the literature.[3-6] A limitation of this theory is that it does not handle multiple priority levels.

Another analytic technique is exemplified by an IBM marketing aid informally called SCAPE (for System Capacity And Performance Evaluation), the subject of this paper. SCAPE is an *open* model, which uses the set of application workloads as input for each performance evaluation.

Other models, which use both analytic and simulation techniques for performance evaluation, are termed *hybrid* models.[7] Their use is primarily in research.

More expensive in terms of CPU time is simulation modeling. Among the simulation models in recent use are FIVE[8] and SNAP/SHOT,[9] which simulate not only the CPU, but the entire computer network.

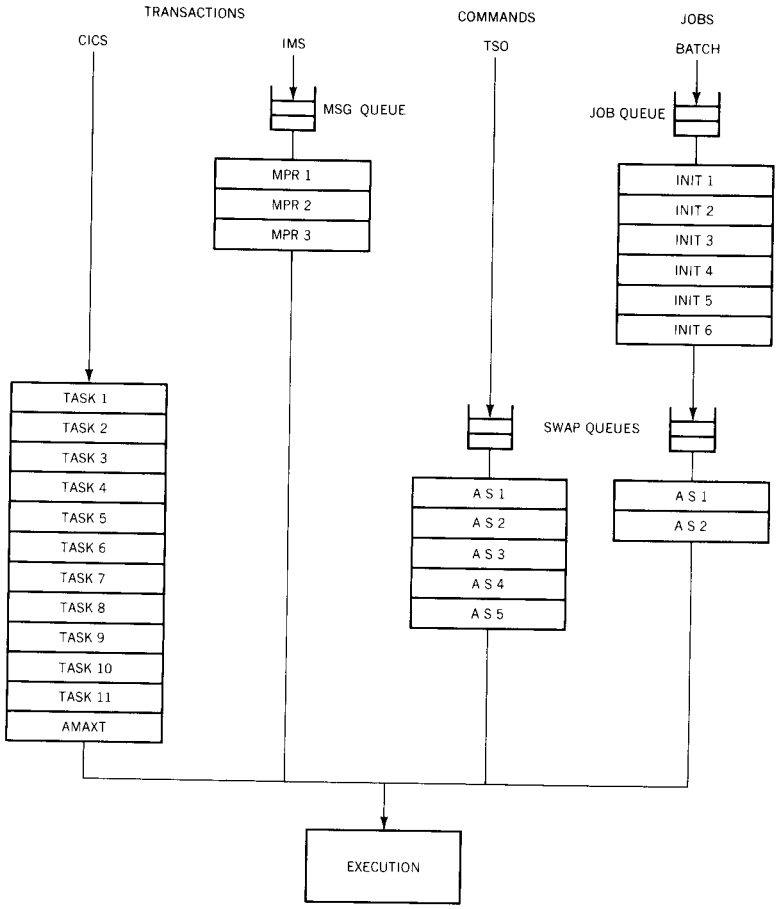
The most expensive technique for capacity planning is benchmarking, which is not really modeling, since it involves executing actual applications in the alternate environment under study. Workload projections for benchmarking can be provided either by manual input or by analytic simulation, as with the Teleprocessing Network Simulator (TPNS).[10] The primary use of benchmarking is not for capacity planning, but for feasibility studies made under contractual obligations.

The SCAPE model

SCAPE is the outgrowth of the author's work at IBM's Washington Systems Center in Gaithersburg, Maryland. Initially the work was done using pencil and paper, with correspondingly simple queuing formulas, mostly the M/M/1 formula in Kendall notation.[11] Later a desk calculator was used, then a programmable calculator was employed for a gross iterative technique more sophisticated than M/M/1. The separate phases of SCAPE were then written in FORTRAN for restricted use on computers. Also, the basic queuing formulas were revised and calibrated using GPSS (the General Purpose Simulation System),[12] and the input and output formats were revised and expanded to provide more flexibility. The remainder of this paper describes the model and the queuing equations used by SCAPE and the general procedure used in a SCAPE analysis.

Recent work with SCAPE has focused primarily on IBM's Multiple Virtual Storage (MVS).[13] Most of this work has been based on data

Figure 1 SCAPE system model



from the Resource Measurement Facility (RMF), available only with MVS. Several features of the SCAPE model, then, pertain only to MVS and its internal structure.

SCAPE handles four types of application workloads—IMS, CICS, TSO, and batch.[14-16] Each has different characteristics as seen by the software system. Most of the differences are in queuing that takes place before a specific application work unit starts, as illustrated in Figure 1.

batch processing

As batch jobs enter the system, they are held in a queue by JES (the Job Entry Subsystem)[17] until an initiator is free to accept them. The SCAPE user defines the number of batch initiators, and this number, together with the average time that a job spends in the initiator, determines the average time spent in the job queue.

Once a batch job is assigned to an initiator, it is eligible for execution only when it is swapped in. Because the number of swapped-in initiators, plus the average time that a job is swapped in real

memory, determines the total time spent by the job in the initiator, the number of swapped-in initiators is crucial to the computation of swap time and, consequently, job time. SCAPE calculates an initial estimate of swapped-in initiators from RMF data; this number may or may not change as the batch workload changes. As the number of swapped-in initiators changes, so does the total number of initiators. Whether or not the SCAPE user changes those numbers with batch workloads is optional.

TSO

In TSO, an address space is equivalent to a batch initiator. Since an MVS user, once logged on, never has to wait for an address space, a TSO command or subcommand is modeled before execution much like a batch application with an infinite number of initiators, SCAPE computes an initial number of swapped-in address spaces (called the target multiprogramming level, or TMPL), which then may or may not be increased with increasing load. Most TSO users do increase the TMPL as the load increases, thus effectively tying the load to the number of active TSO users, with each user's load essentially constant.

With TMPL and the average response time in a swapped-in address space, SCAPE computes the swap-out time before being swapped in. Thus it computes the total time that a command occupies an address space, whether swapped in or out.

Swapping that occurs after the initiation of a job (occasionally of a command) is treated by SCAPE as if it occurred before initiation. No differentiation is evident in RMF either—only the time that any work unit is swapped in or out. Swap-out time, then, is the total of the initial swap-out and all later swap-outs that interrupt execution.

IMS and CICS

The other types of applications, IMS and CICS, are usually considered nonswappable, although nothing in SCAPE prevents the modeling of a swappable DB/DC (data base/data communications) software system. Usually, then, no swap queues are modeled for DB/DC applications. However, for an IMS transaction, a job queue (in IMS called a message queue) is modeled the same way as in batch processing, using the number of message processing regions instead of the number of initiators.

As shown in Figure 1, CICS has neither message queues nor swap queues, the usual case if the maximum number of simultaneously active tasks is large (20 or more). If the number is small, say 4 or 5, a message queue exists for CICS just as for IMS, and the extra time spent in the message queue is included in SCAPE's overall turnaround time for the transaction.

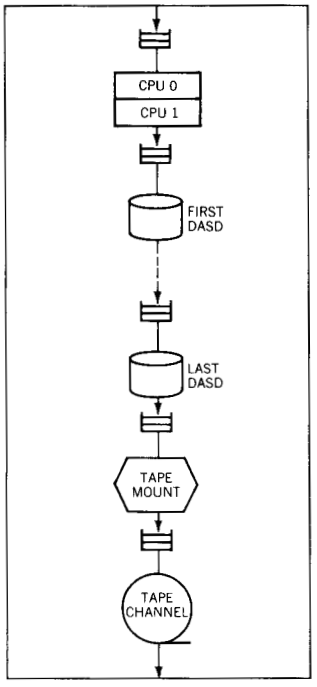
Most teleprocessing packages other than IMS and CICS resemble CICS in that they execute in a single address space, in either a

single-threaded or multithreaded mode. Occasionally they perform functions like those of both IMS and CICS, and the user has to exercise his ingenuity in defining the transactions within the SCAPE framework. Sometimes such a transaction is modeled as both an IMS and a CICS transaction, the total execution time being the sum of the two.

execution

Once an application enters execution (see Figure 2), it is handled more or less the same, regardless of application type. An exception is the CPU, where a CICS-type application can be executed in only one side of a multiprocessor.[18]

Figure 2 SCAPE execution model



The modeling of an executing application by SCAPE is essentially a serial process. Each work unit is passed through each resource in turn, its waiting time for that resource being added to its service time, then all individual response times (service plus waiting times) are totaled to produce the overall response time. Service time in the CPU is the sum of the small time increments between interruptions of any work unit. For I/O devices, service time computation depends on the device type. For each direct-access storage device (DASD), the service time is the number of accesses to that device for the work unit, multiplied by the average service time for accessing that device.

The service time for tape mounting is the average time required to mount a tape, multiplied by the average number of tape mounts per job. For tape reading and writing, the service time is the tape channel time apportioned among the different applications that use tape. Since tape units cannot be active unless the tape channel is working, SCAPE does not model the tape units themselves. It assumes either a single tape channel or that all tape channels are shared by all tape units.

For reporting, direct-access devices are divided into three groups: paging, swapping, and data base devices. Therefore SCAPE treats each work unit as if it passed through six subsystems: CPU, tape, tape mounting, data base DASD, paging, and swapping.

SCAPE considers JES and teleprocessing access methods only as they affect the other applications. JES is considered to exist in the CPU as the highest-priority task, but its service and response times are not considered. Neither are its accesses to DASD (SPOOL packs) considered. A user must consider JES a separate application if he wishes to determine JES time (mostly printing), which then can be accounted for by adding it to batch response or turnaround time. In that case, of course, unit-record times are not included.

Teleprocessing access methods are treated similarly, as top-priority overhead. The only difference is that this overhead is

considered constant with increasing load, whereas JES CPU utilization is considered to grow (or decline) as does the lowest-priority application.

Priorities are considered by SCAPE only as dispatching priorities within the CPU. They are treated as being in a definite order, so applications that reverse priorities dynamically are considered by SCAPE to have the same priority. The user may choose to model this phenomenon outside of SCAPE itself. With APL,[19] for example, priority often is changed dynamically so that it is sometimes above the priority of batch and sometimes below. A way to model this situation is to execute two SCAPE models, one with the APL priority above batch and one below. Then results from the two runs can be combined, weighted by the portion of time spent in each of the two priority states.

Mathematical foundations

The SCAPE procedure is based primarily on two types of queuing equations: internal and external. Internal queuing involves contention for a hardware resource, such as the CPU or a direct-access device. External queuing involves contention for a region or address space, which is a software resource.

To place internal and external queuing in context, the simplest of all queuing equations is, in Kendall notation,[11] the M/M/1:

$$tr = ts/(1 - U) \tag{1}$$

where tr is the response time through a single server, ts is the service time through the server, and U is the utilization of the server, or the portion of time that the server is busy. This equation, however, is suitable for only a single server in the system, a single application with random arrivals and exponential service.

A more general equation, which handles service distributions other than exponential, is (again in Kendall notation) M/G/1, where G stands for a service distribution that can range from constant to hyperexponential. The equation, known as the Khinchine-Pollicheck equation,[20] is:

$$tr = \frac{ts}{1 - U}\left[1 - \frac{U}{2}(1 - c^2)\right] \tag{2}$$

where c^2 is the coefficient of variation, or the variance of service time divided by the square of the average service time. If c^2 is unity, the distribution is exponential, and Equation (2) degenerates to Equation (1).

The restriction to exponential service distribution is the only limitation of Equation (1) corrected by Equation (2), and more data is

required for the implementation of Equation (2). Armstrong[21] assumes fixed values of c^2 for each component of the computer system (1 for CPU, 0.5 for DASD, 0 for tape). Two other phenomena not handled by Equation (2) are nonexponential arrival distributions and dependent flow from one system component to another. Allen[22] treats both by using:

$$tr = \frac{ts}{1 - U}\left[1 - \frac{U}{2}\left(1 - \frac{ca^2 + cs^2}{2}\right)\right] \tag{3}$$

where ca^2 is the coefficient of variation of the arrival time, and cs^2 is the coefficient of variation of the service time. In Equation (3), cs^2 has been replaced by the average of itself and ca^2. The justification for Equation (3) is mostly empirical.

Equations (1), (2), and (3) are open queuing formulas because they use the workload as an independent input variable. It is used indirectly through:

$$U = L \times ts \tag{4}$$

where L is the workload, in work units per time period.

internal queuing

SCAPE also employs open queuing formulas. The general form of the basic, or internal, SCAPE equation can be derived, but its specific form was obtained empirically by using comparisons with GPSS runs. The basic equation used by SCAPE for response times through the CPU is:

$$trc(j) = \frac{A(j) \times B(j)}{C(j)} \tag{5}$$

where

$$A(j) = \frac{tsc(j)}{1 - U(j)^s} \tag{6}$$

$$B(j) = 1 - U(j)^{(K(j)-s)\times r(j)+s} \tag{7}$$

$$C(j) = 1 \text{ for } j = 1$$

$$= 1 - U(j-1)^{s\times r(j)} \text{ for } j > 1. \tag{8}$$

In the equations above, the following values apply:

j = number of the application (1 − 14 for the SCAPE programs)
$trc(j)$ = response time of the jth application through the CPU
$tsc(j)$ = service time of the jth application through the CPU
$U(j)$ = utilization of the CPU, by the jth application plus all applications of higher priority
s = number of servers (1 for a uniprocessor, 2 for a multiprocessor)
$K(j)$ = maximum multiprogramming level for the application
$r(j) = trc(j)/tr(j)$ (9)
$tr(j)$ = response time of the jth application through all subsystems.

Equivalent formulas are used for all other components of the system, such as DASD, tape, and tape mounting. Then the system response time is the total response time through all subsystems.

As can be seen from Equation (9), the basic SCAPE equation is recursive. Initial values are assumed for response times through all subsystems, values of $r(j)$ are determined, and new values of $trc(j)$ and other response times are calculated. Then new values of $r(j)$ are determined, and the cycle is repeated until either successive response times are approximately the same or some maximum has been reached.

Equation (5) is complex enough that it is not practical to use repeatedly except by computer. However, it still represents an open model because the load is an input (implied in the utilization). Equations (1), (2), (3), and (5) all represent open queuing models, using different degrees of sophistication.

head-of-line modification

Equation (5) represents the basic queuing scheme used by SCAPE for the CPU. It assumes that all priority relationships are *pre-emptive resume*—that is, a lower-priority job can be interrupted at any time by a higher-priority transaction, after which the lower-priority job resumes its work. The high-priority work could itself be pre-empted by a yet higher-priority transaction.

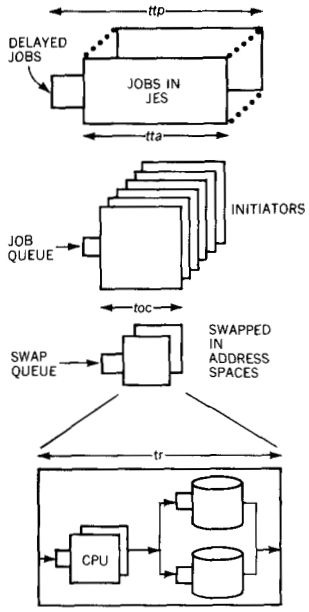
However, in the work of a CPU, a significant amount of code is disabled; that is, upon interruption, the CPU merely stores the interrupt and continues with the disabled code. When the disabled code is completed, the next transaction to be acted upon is the one of highest priority, even though it might have occurred last. This priority scheme is called *head-of-line*. Approximately 50 percent of MVS supervisor code, consisting largely of I/O subroutines, is disabled.

The effect of disabled code on response time is determined first by dividing the total CPU time into segments separated by I/O or other interruptions. Then the extra time that any higher-priority task has to wait is only a portion of the interval in which it happens to arrive. Assume a situation with n applications. If application j interrupts application i, let $N(i)$ be the number of interruptions of a work unit of i, and $ts(i)$ be the total CPU service time of a work unit of i. Then $dts(i)$, the average CPU time between interruptions of application i, is:

$$dts(i) = ts(i)/N(i) \tag{10}$$

One would normally think that the delay for application i would be half of this interval. However, since the interruptions occur randomly, the length distribution of the interval is exponential, so that $dts(i)$ is the expected delay caused by an interruption of application j.

Figure 3 Time relationships



Now, if $N(j)$ is the number of interruptions of a work unit of application j, and $duc(i)$ is the probability that i is in core, then the total delay of j because of i is:

$$F(i, j) = N(j) \times duc(i) \times dts(i)$$
$$= duc(i) \times ts(i) \times N(j)/N(i) \quad (11)$$

Note that $duc(i)$ is just the CPU utilization of i, the product of its CPU service time and its workload. To determine the total delay of j caused by applications of lower priority, it is necessary to sum all values of $F(i, j)$ for all i applications.

On the other hand, the lower-priority application is speeded up because it does not have to wait for these interruptions. Moreover, the decrement in time for i is the same as the increment in time for j, adjusted for their relative workloads. Therefore $G(j, i)$, the corresponding decrement in time, is:

$$G(j, i) = F(i, j) \times L(i)/L(j) \quad (12)$$

Again, the values of $G(j, i)$ are summed, this time for all applications of higher priority. Then for any application j, the modified response time $tr'(j)$, to allow for disabled code, is:

$$tr'(j) = tr(j) + f \times \left(\sum_{i=j+1}^{n} F(i, j) - \sum_{i=1}^{j-1} G(j, i) \right) \quad (13)$$

where f is a factor between 0 and 1, depending on the operating system used in the CPU. In SCAPE, these priority equations are employed only for the CPU.

external queuing

Once an application's system response time has been found, SCAPE uses other formulas to obtain waiting times in the swap-out queue and the job or message queue, and waiting times for delayed jobs. Figure 3 shows the relationships among the various compound times used by SCAPE: response time (tr), occupancy time (toc), turnaround time (tta), and throughput time (ttp).

To find the total system response time for any application, SCAPE merely totals the response times for all subsystems as calculated in Equations (5) through (8). However, an application is likely to spend additional time waiting to start execution. For example, all TSO commands must be executed by swapped-in address spaces, the number of which is set by MVS. If a TSO command, upon entering the system, finds that its address space is swapped out, it must wait in a swap queue until one of the swapped-in TSO address spaces is itself swapped out or completed. Similarly, IMS transactions may have to wait in a message queue for message processing regions, and batch jobs may have to wait in a job queue for initiators or in a swap queue for swapped-in address spaces.

The queuing involved in these cases is multiserver queuing, the servers being initiators, message processing regions, and address spaces instead of hardware subsystems as with internal queuing. The multiserver queue as used by SCAPE assumes a finite number of servers to handle jobs or transactions that come from an infinite population. In a computer system the population is not infinite, of course, but is limited for on-line transactions by the number of terminals and for batch transactions by the number of persons who might submit jobs. However, those numbers are presumed to be large enough that any discrepancy in assuming an infinite source population is insignificant.

Another assumption made by SCAPE is that the total response time of an application is the average of an exponential distribution. With these assumptions in mind, there is an exact expression for, say, the occupancy time of a TSO command (response time plus swap-out time), *toc*, as a function of its response time, *tr*, the average number of swapped-in address spaces, K, and the average utilization, U, of each of those address spaces.[23] An intermediate factor is Po, the probability that all address spaces are empty:

$$Po = \frac{1}{1 + KU + \frac{1}{2!}(KU)^2 + \cdots + \frac{1}{K!}(KU)^K\left(\frac{1}{1-U}\right)} \tag{14}$$

With that factor known, the occupancy time is:

$$toc = tr + \frac{tr \times U^2}{(K-1)!(1-U)^2} \times Po \tag{15}$$

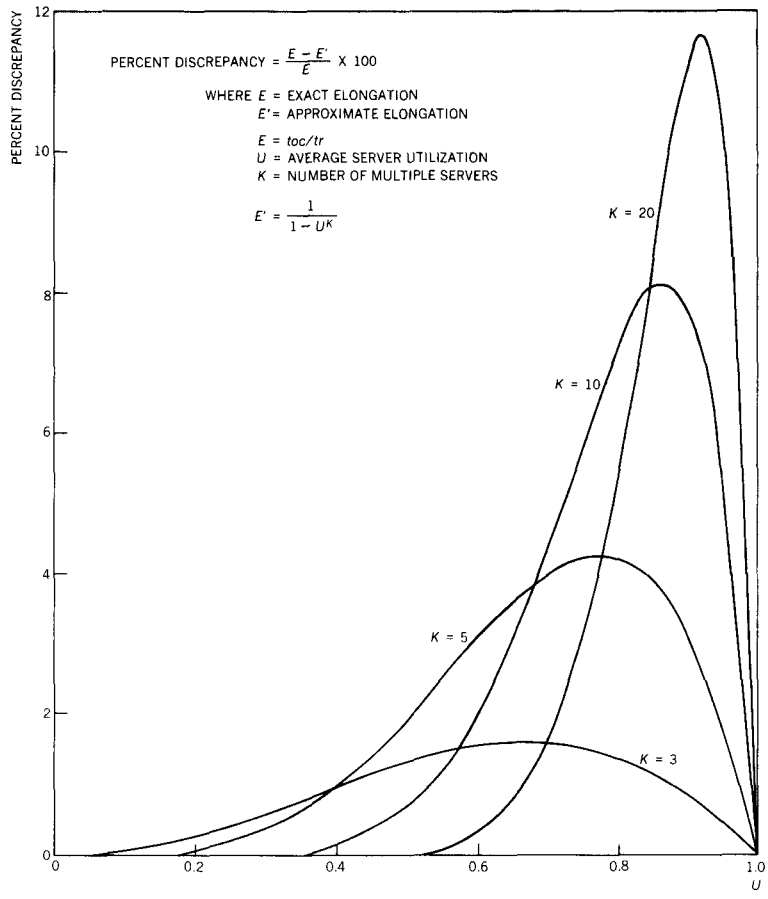
Equations (14) and (15) are valid only for integral values of K because the expression for Po contains exactly $K + 1$ terms in the denominator. However, it is probable that, as determined by MVS, the target multiprogramming level for the number of swapped-in address spaces over a base period of an hour is not an integral number. Therefore an expression must be found that approximates Equations (14) and (15) at integral values of K, yet is valid for all positive values.

SCAPE uses the following equation, which satisfies those requirements:

$$toc = tr \times \frac{1}{(1 - U^K)} \tag{16}$$

For K equal to 1 or 2, *toc* from Equation (16) is exactly the same as from (14) and (15). For higher values of K, *toc* as obtained from (16) is slightly less than the exact expression. However, the discrepancy is small, as shown in Figure 4, where values of K are as high as 20 (and the waiting times are small), and the discrepancy is still less than 12 percent of the true value. The fact that Equation (16) is valid for all positive values of K certainly justifies the slight discrepancy.

Figure 4 Multiserver approximation



For batch applications, the relationship between *toc* and *tr* must be changed slightly because of the finite number of batch initiators. SCAPE inserts an extra factor into Equation (16) to account for this requirement. For batch processing, then:

$$toc = tr \times \frac{(1 - U^Y)}{(1 - U^K)} \tag{17}$$

where U is domain utilization, K is the maximum number of jobs swapped in, and Y is the maximum number of jobs swapped in or out. Here, domain utilization, U, can be expressed:

$$U = L \times tr/K \tag{18}$$

For DB/DC applications, which usually are not swappable, occupancy time differs from response time only because of data reserve requirements, not because of time spent in a swap queue.

Turnaround time for batch processing bears the same relationship to occupancy time as does occupancy time to response time for TSO:

$$tta = toc \times \frac{1}{(1 - U^Y)} \tag{19}$$

where U is region utilization, Y is the number of initiators for batch processing, the number of message processing regions for IMS, the maximum number of active tasks for CICS, or infinity for TSO, so that $tta = toc$. Here, region utilization, U, can be expressed:

$$U = L \times toc/Y \tag{20}$$

Examination of Equations (17) and (19) shows the justification for the extra term in Equation (17). If the two equations are multiplied together, the term appears in both numerator and denominator. It can be cancelled, and turnaround time, as a function of response time, is dependent not on the total number of initiators, but on the number of swapped-in initiators. The only effect of the total number of initiators is to separate the job and swap queues.

All occupancy times and turnaround times can be obtained from the external equations (16) through (20). The only other time obtained by SCAPE is the throughput time, *ttp*, of batch jobs that are delayed to allow other workloads to grow. The equation used is a strictly linear relationship between the average response time and the interval of the peak load (assumed to be one hour):

$$ttp = tta + \left(\frac{external\ load}{internal\ load} - 1\right) \times 1800 \tag{21}$$

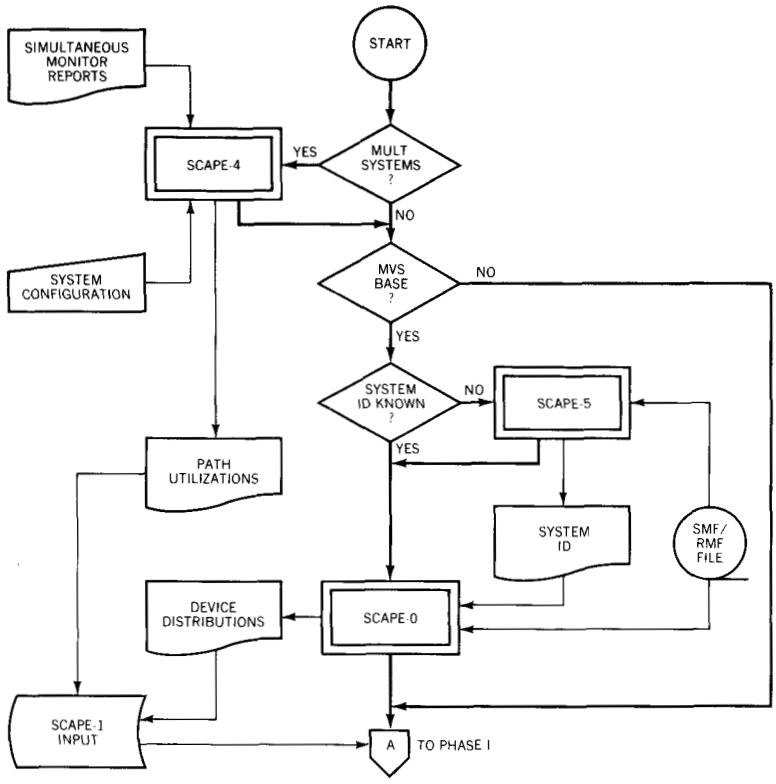
where *external load* is the batch load, in jobs per time period, placed on the system by its users, and *internal load* is the batch load actually processed by the system without saturating it.

In summary, SCAPE employs queuing equations to obtain both the basic response times through all hardware and the longer input queuing times spent waiting for software servers. The internal queues are based on a combination of *pre-emptive resume* and *head-of-line* priorities in the CPU and an absence of priorities elsewhere. The external queues are handled differently, depending on whether each application is DB/DC, TSO, or batch.

System overview

A SCAPE analysis proceeds in four phases: data gathering (Phase 0), calibration (Phase I), prediction (Phase II), and modification (Phase III). The flow of the analysis through these phases is described in the following sections, in which reference is made to six programs, SCAPE-0 through SCAPE-5, that handle different portions of the computer calculations.

Figure 5 SCAPE Phase 0 flow



Phase 0 As illustrated in Figure 5, the analyst first examines a set of measurement data for the purpose of choosing as a base that period which best exemplifies the system at its peak workload. During that period, all applications should be working and all should be measured. The peak period is typically an hour during the day and somewhat longer if at night. SCAPE data, most of which is available from RMF either directly or indirectly, consists of:

- CPU utilization during the base period.
- Activity of all tape and DASD channels (number of accesses to each channel).
- Utilization of all tape and DASD channels (portion of time that each channel is in use).
- Activity of all direct-access devices.
- Utilization of all direct-access devices.
- Paging and swapping rates, both in and out of the CPU.
- CPU service time for each application. For MVS, each application is a performance group, several performance groups, or a period within a performance group. Statistics from single

periods are used if the user decides to evaluate trivial and non-trivial TSO performance separately, instead of treating TSO as a single application. In this instance, a copy of the Installation Performance Specifications is required.

- Workload of each application (jobs per second for batch, commands per second for TSO, and transactions per second for IMS and CICS). To obtain the application workload for IMS and CICS, monitors other than RMF are required—for example, the DC Monitor for IMS[24] and the Performance Analyzer for CICS.[25] These monitors should span approximately the same time interval as does RMF.
- Average elapsed time of each application. Again, for IMS and CICS applications, the DC Monitor or Performance Analyzer is required. If such a tool is not available for the system under study, then no Phase I calibration is possible, and the user must rely on a reasonableness test of total response time.

Also during Phase 0, the user gathers the following information for the analysis:

- The hardware configuration during the base period.
- A list of the applications.
- For each non-TSO application, the maximum multiprogramming level. For batch, the equivalent information is the number of initiators; for IMS it is the number of message processing regions; and for CICS it is the maximum number of active tasks. For TSO the number is assumed to be infinite, which is equivalent to the application's never having to wait for a free address space; in practice MVS assigns an address space when each user logs on, before any commands are issued, and the user keeps that address space throughout his session.
- The rate and type of growth of each application. Growth may be linear, exponential, or a general type for which the workload is entered month by month.
- The critical turnaround time of each application. Critical times may be altered as the SCAPE analysis proceeds through later phases.
- Initial estimates of CPU capture ratio for each application. The capture ratio is the CPU service time for the application captured, or measured, by RMF, divided by the total CPU time for the application, including system time. These ratios can be obtained from previous SCAPE analyses or from USAGE analyses of the system. Default values are used for the different application types.
- The portion of each DASD devoted to each application, unless the devices are dedicated. For devices that are shared among several applications, the only sure way to apportion individual DASD activity is to use the Generalized Trace Facility,[26] which can be expensive and has to be planned ahead. Another tech-

nique is to use the program called SCAPE-0, which employs System Management Facilities (SMF) data[27] to apportion EXCPs (Execute Channel Programs) by device among performance groups. This program does not provide information about system accesses, but it is a partial aid in determining DASD portions. Later on, in Phase I, these values may very likely be altered.

- The portion of tape activity for each application. SCAPE-0 may be used for this purpose also.
- Tape mounting statistics: the number and average utilization of people mounting tapes or disks, plus the average time spent mounting tapes and disks for each application.
- A preliminary list of alternate hardware and software environments. This list could include all changes to the base system which the user wants to analyze with SCAPE. The list may change as successive runs are made in Phase III, if the results indicate other possible improvements in system capacity.

One other activity may be required during Phase 0 if more than one CPU is involved. If DASD controls are shared, it is necessary to select monitor reports simultaneously (as much as possible) from all systems that share the DASD configuration. These reports are used by SCAPE-4 to compute path utilization between all strings of DASDs and all CPUs. These path utilizations replace actual channel utilizations as input data for SCAPE-1.

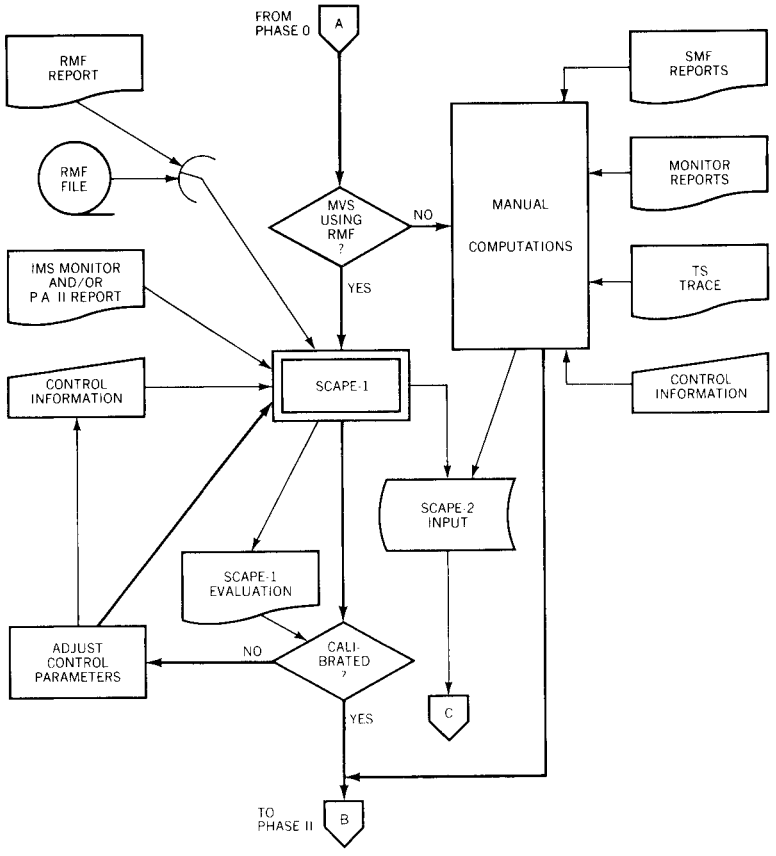
Phase I

Once information is collected for an MVS base system, it can be entered into SCAPE-1, which uses it to set up parameters for SCAPE-2, a prediction program (see Figure 6). SCAPE-1 uses input data in queuing equations that predict the total response time through the entire system for each application. The analyst compares that total with an input response time, which for each application is derived directly (without queuing) from other data. If any calculated response times differ appreciably from the equivalent input response times (say, over 10 percent), then the user must change some of the data entered into SCAPE-1 and run the program again. Some inputs which have been changed in this fashion have been:

- Capture ratios.
- Relative priority levels.
- Amount of disabled code.
- Fixed portions of different applications.
- Seconds of mount time per application.
- Portions of individual DASDs used by different applications.
- Maximum multiprogramming levels.

Occasionally no amount of adjustment will calibrate the base system. In that case there must be some systemic reason why the queuing questions used by SCAPE do not predict the measured

Figure 6 SCAPE Phase I flow

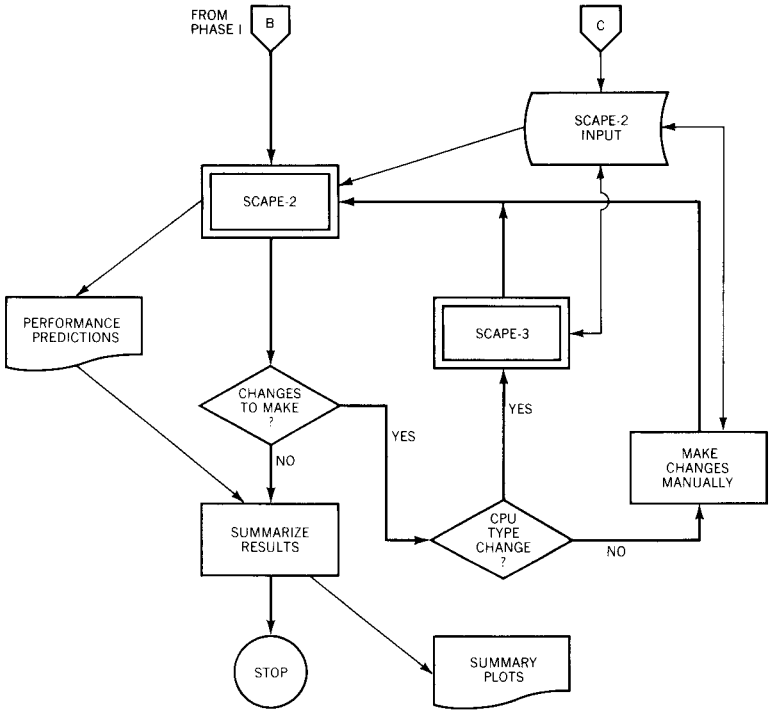


response times adequately. For example, if the DASD accesses are served in an extremely irregular fashion (with a large hyper-exponential distribution), response times through the DASDs will be long enough to prevent calibration of the base system. In this case, it is probably best to adjust other measured data. In this example, increasing the DASD utilizations is suggested.

Usually, however, the base system can be calibrated by repeated applications of one or more of the above input adjustments. On the average, between five and ten such runs are required, taking about a day for each system.

If the base system is other than MVS with RMF reports, the user must perform the SCAPE-1 function manually. Occasionally it is possible to convert such data as does exist into "pseudo-RMF" data that can be read by SCAPE-1. The only alternative is to perform the SCAPE-1 function with an equivalent program or with a desk calculator.

Figure 7 SCAPE Phases II and III flow



Phase II

Once Phase I is complete, Phase II can follow immediately with the execution of SCAPE-2 (see Figure 7). Initial results of SCAPE-2 should be compared with the final results of SCAPE-1. For all applications, the total response times from SCAPE-1 and the first output page of the SCAPE-2 listing should agree within one percent.

The rest of the SCAPE-2 report is a prediction of how the base system will perform in the future. Often this prediction is enough for Phase II, and the analyst proceeds with Phase III. Sometimes additional runs are made with the base system, however, and other parameters are modified. For example, the growth rates of the different applications may be altered one at a time. The resulting set of SCAPE-2 runs constitutes a sensitivity analysis for workload changes to the base system. Also, the analyst may reset some of the input switches so that, for example, the base system can continue to operate beyond saturation. These input switches could have been set this way as input to SCAPE-1.

SCAPE-2 produces a report for each month of the prediction. Figure 8 is an example for one month, showing statistics for a complex IMS-TSO-batch workload during the base period. The line labeled WORKLOAD describes the transactions per second, com-

Figure 8 Performance prediction for checkout example

APPLICATION	IMS	TSO	BAT
WORKLOAD	1.10749	0.66704	0.01893
CPU SERVICE TIME	0.21300	0.58200	9.12300
CUM. CPU UTIL.	0.32629	0.70517	0.87372
CPU RESPONSE TIME	0.37927	2.63168	161.17043
TAPE SERVICE TIME	0.01000	0.0	2.75100
TAPE CHANNEL UTIL.	0.06299		
TAPE RESPONSE TIME	0.01004	0.0	2.77541
MOUNT SERVICE TIME	0.0	0.0	60.00000
MOUNTER UTILIZATION	0.20000		
MOUNT RESPONSE TIME	0.0	0.0	73.27353
DASD PATH UTIL.	0.23900		
DATA BASE SERV. TIME	0.31393	1.29499	32.72629
DATA BASE RESPONSE	0.32117	1.36755	33.37027
PAGING SERV. TIME	0.12496	0.28648	4.48807
PAGING RESPONSE	0.12909	0.29257	4.52957
SWAPPING SERV. TIME	0.0	0.41405	0.12467
SWAPPING RESPONSE	0.0	0.44154	0.12474
TOTAL RESPONSE TIME	0.83958	4.73334	275.24365
CURRENT MPL	0.92982	3.15733	5.21036
DOMAIN UTILIZATION	0.30994	0.76486	0.86839
OCCUPANCY TIME	0.85217	7.07196	275.24365
CONCURRENCY	0.94377	4.71728	5.21036
REGION UTILIZATION	0.31459	0.0	0.86839
TURNAROUND TIME	0.87955	7.07196	481.90625
POWER	1.13694	0.14140	0.00208

mands per second, and jobs per second for the IMS, TSO, and batch applications respectively. All times in the report are given in seconds to avoid confusion; therefore batch workloads seem small relative to interactive workloads.

The next three lines in Figure 8, for the CPU, display service times, cumulative utilizations, and response times for each application. Each CPU utilization reported includes utilization for the

specific application, plus utilization for all applications of the same or higher priority, plus the utilization for paging, JES, and other high-priority background activity.

Next are three similar lines for the tape subsystem (service times, utilization, and response times), with the exception that one tape channel utilization represents all applications. The tape subsystem reports are followed by three similar lines for the mount subsystems (usually tape, but disk mounts could also be included).

Next in Figure 8 is a line for DASD path utilizations. (For the example shown, all DASDs use the same path, so only one path utilization is shown.) These values are not applicable to specific applications; they correspond to different DASD paths. Each path utilization is employed in determining the service time of each direct-access device that uses the path.

The following two lines show the total data base service and response times for all applications. In SCAPE, all DASD functions are assumed to be data base, paging, or swapping, and input is assumed to be given in those categories. SCAPE reports the times spent in each of the three subsystems separately. After the two lines for data base, there are two similar lines for paging and two for swapping. In SCAPE-2, switches are available to display paging and swapping times for individual devices, as well as device utilizations. To keep the quantity of output down, the entire set is not usually printed out.

The next line in Figure 8, labeled TOTAL RESPONSE TIME, shows the total response times (tr) summed over the six subsystems: CPU, tape, mount, data base, paging, and swapping. The next two lines are obtained from the total response times: the current multiprogramming level, which is the product of the response time and the workload, and domain utilization, the portion of the multiprogramming level divided by the number of address spaces allowed to be swapped in by the System Resources Manager of MVS. Domain utilization is obtained from Equation (18). The equation that relates response time (tr), workload (L), and the number of concurrent users in the system (n) is known as *Little's Law*:[28]

$$n = L \times tr \tag{22}$$

Following the three lines related to response times are three similar lines related to occupancy times. Occupancy time is related to response time by Equation (16), concurrency is the number of work units swapped in or out (and thus related to occupancy time by Little's Law), and region utilization is concurrency divided by the number of initiators, or the equivalent for IMS and CICS. Region utilization is obtained from Equation (18).

At the bottom of Figure 8 are turnaround times, related to occupancy times by Equation (17), and the powers, or values of quality of service. For all applications, either the turnaround time or the power is the final figure of merit for judging whether the application has been adequately serviced by the system.

Phase III

Phase III is the "what-if" portion of a SCAPE analysis (see Figure 7). Whenever a change is to be evaluated in the system under consideration, one or more of the input parameters to SCAPE-2 is changed, and SCAPE-2 is executed to provide a new prediction of how all the applications will behave in the future. Many changes have been evaluated successfully. The possibilities of modeling hardware and software are limited only by the imagination of the analyst.

SCAPE-3 is designed to handle some of these system changes, specifically those relating to the CPU. If the analyst wants to evaluate a change in the power (MIP rate) of the CPU, in the number of processors in the CPU, or in memory size, then SCAPE-3 can be run with the SCAPE-2 input to be modified by the SCAPE-3 program plus control input that defines the change to be made. The result is a revised set of data for SCAPE-2 and a report that describes new values of the changed parameters. Successive modifications can be entered in the same execution of SCAPE-3, in which case several sets of SCAPE-2 data are produced, one after another, each containing cumulative changes defined by the successive controls. It remains then to execute SCAPE-2 in order to predict the effects of the new configurations.

Any other changes must be evaluated by manual calculation of new input parameter data for SCAPE-2, followed by the execution of SCAPE-2. Some of the changes are:

- Additional DASD channels.
- DASD model change (3330 to 3350).
- Additional DASDs.
- Additional tape channels.
- Additional tape mounters.
- Tape model changes.
- Conversion to a Mass Storage System.
- Change in operating system.
- Change in software release.
- Change in multiprogramming level.
- Change in tape block sizes.
- Change in DASD block sizes.
- Moving workloads from one CPU to another.
- New application package.

Some of these changes are simpler than others. For example, additional tape channels can be modeled merely by decreasing the

initial utilization for tape, as input to SCAPE-2. However, additional DASD channels cannot be modeled so easily. Not only must the path utilization be reduced, but the utilizations of all DASDs using the path must be reduced in such a way as to reflect the reduced path utilization correctly.

The usual SCAPE analysis involves 10 to 40 cases in which 5 to 10 changes are analyzed in different combinations. About two days normally are required for Phases II and III. The total time, for Phases I, II, and III for a single CPU, is about three days.

Concluding remarks

SCAPE has been used as an aid for capacity planning in about 50 installations in the United States. In most instances, it has been of significant value in determining the optimal configuration for future use, justifying the procurement of hardware, or pinpointing possible bottlenecks. In addition, during the calibration phase of SCAPE, many users have gained a better understanding of their systems, how they might be improved, and how to track them in the future.

CITED REFERENCES AND NOTE

1. J. C. Cooper, "A capacity planning methodology," *IBM Systems Journal* **19,** No. 1, 28–45 (1980, this issue).
2. P. H. Seaman, "Modeling considerations for predicting performance of CICS/VS systems," *IBM Systems Journal* **19,** No. 1, 68–80 (1980, this issue).
3. J. P. Buzen, "Computational Algorithms for Closed Queuing Networks with Exponential Servers," *Communications of the ACM* **16,** No. 9, 527–531 (September 1973).
4. J. W. Boyse and D. R. Warn, "A Straightforward Model for Computer Performance Predictions," *Computing Surveys* **7,** No. 2, 73–93 (June 1975).
5. R. R. Muntz, "Analytic Modeling of Interactive Systems," *Proceedings of the IEEE* **63,** No. 6, 946–953 (June 1975).
6. L. Kleinrock, *Queuing Systems*, Vol. I: *Theory*, John Wiley & Sons, Inc., New York (1974), pp. 147–160.
7. W. W. Chiu and W. M. Chow, "A performance model of MVS," *IBM Systems Journal* **17,** No. 4, 444–462 (1978).
8. H. C. Nguyen, A. Ockene, R. Revell, and W. J. Skwish, "The role of detailed simulation in capacity planning," *IBM Systems Journal* **19,** No. 1, 81–101 (1980, this issue).
9. H. M. Stewart, "Performance analysis of complex communications systems," *IBM Systems Journal* **18,** No. 3, 356–373 (1979).
10. *Teleprocessing Network Simulator (TPNS) Release 3.0 General Information Manual*, IBM Systems Library, order number GH20-1907, available through IBM branch offices.
11. A. O. Allen, "Elements of queuing theory for system design," *IBM Systems Journal* **14,** No. 2, 161–187 (1975).
12. *General Purpose Simulation System V—Introductory User's Manual*, IBM Systems Library, order number SH20-0866, available through IBM branch offices.
13. *OS/VS2 MVS Overview*, IBM Systems Library, order number GC28-0984, available through IBM branch offices.

14. *Information Management System/Virtual Storage (IMS/VS) General Information Manual*, IBM Systems Library, order number GH20-1260, available through IBM branch offices.
15. *Customer Information Control System/Virtual Storage (CICS/VS) General Information Manual*, IBM Systems Library, order number GH20-1280, available through IBM branch offices.
16. *OS/VS2 TSO Terminal User's Guide*, IBM Systems Library, order number GC28-0645, available through IBM branch offices.
17. *OS/VS2 MVS System Programming Library: JES2*, IBM Systems Library, order number GC23-0002, available through IBM branch offices.
18. CICS Release 1.5, scheduled to be available late in 1980, operates in multiple address spaces and on both sides of a multiprocessor.
19. K. E. Iverson, *A Programming Language*, John Wiley & Sons, Inc., New York (1962).
20. T. L. Saaty, *Elements of Queuing Theory*, McGraw-Hill Book Co., Inc., New York (1961).
21. R. M. Armstrong, "Capacity planning and performance analysis using RMF," *Proceedings, GUIDE 45*, 1011–1060 (1977).
22. A. O. Allen, *Probability, Statistics, and Queuing Theory*, Academic Press, Inc., New York (1978), p. 221.
23. M. Sasieni, A. Yaspan, and L. Friedman, *Operations Research—Methods and Problems*, John Wiley & Sons, Inc., New York (1959), p. 138.
24. *IMS/VS Utilities Reference Manual*, IBM Systems Library, order number SH20-9029, available through IBM branch offices.
25. *CICS/VS Performance Analyzer II*, IBM Systems Library, order number SB21-1697, available through IBM branch offices.
26. *OS/VS2 MVS System Programming Library: Service Aids*, IBM Systems Library, order number GC28-0674, available through IBM branch offices.
27. *OS/VS2 MVS System Programming Library: System Management Facilities (SMF)*, IBM Systems Library, order number GC28-0754, available through IBM branch offices.
28. J. D. C. Little, "A proof for the queuing formula $L = \lambda W$," *Operations Research* **9**, 383–387 (1961).

The author is located at the IBM Washington Systems Center, 18100 Frederick Pike, Bldg. 2, Gaithersburg, MD 20760.

Reprint Order No. G321-5115.