In recent years there has been a growing need to develop techniques and tools for computer installation capacity planning. This paper presents both a tool, in the form of an analytic queuing model, and a methodology for performance analysis and capacity planning. Although general in its approach, the model was developed specifically for the CICS/VS environment.

Analytic queuing model for CICS capacity planning

by M. Deitch

Two of the more common trends in data processing shops today are the increase in on-line applications and the corresponding increase in data processing budgets. As businesses place more and more of their critical functions under control of on-line computer systems, capacity planning becomes an increasingly important part of business planning. In the past, many businesses were content to look at their data processing capacity needs once a year or less frequently. Capacity was often measured in terms of using the CPU and the DASD to their effective limits (i.e., 90 percent CPU utilization and 40 percent DASD utilization). With heavy reliance on terminal-based systems today, capacity limits must be redefined. Systems are not out of capacity when their resources reach some peak utilization. Rather, capacity is exceeded when the system is no longer capable of providing a minimum level of acceptable performance. If this definition of capacity is accepted we can then approach the job of capacity planning from a new perspective. One result of this view is the need for a tool that can predict on-line system performance as a function of a business's increasing workload. This paper is a discussion of such a tool and a methodology showing how it can be used to perform analysis and capacity planning.

A queuing model for CICS and its assumptions

The model presented here is a mathematical analogy of essential parts of a computing system. Known as an *analytic model*, it has been found to be a useful method, since it can be programmed and used over and over to study a variety of data processing environments in a short period of time.

The effectiveness of any such model depends on the assumptions that are made to bridge the gap between the model and the system itself. To the extent that the assumptions are realistic, the model tends to mimic the real system and serve as a useful tool for predicting system performance. With this background in mind, we examine the Customer Information Control System/VS (CICS/VS) subsystem and derive an analytic queuing model to predict system performance.

CICS is a collection of transactions contending for CPU service among themselves and with other tasks that execute at a higher dispatching priority. If we collect all the high-priority tasks (including operating system overhead) together into one class of transactions and CICS into another class of transactions, we have created a two-level priority queuing system.

If a higher-priority transaction enters the system while a CICS transaction is being serviced by the CPU, we assume that the higher-priority task preempts the control that CICS has. We further assume that the CICS transaction resumes executing at the point of interruption after the completion of the higher-priority task. Such a queuing system is said to be governed by a *preemptive/resume* queuing discipline.

It is necessary to make some assumptions about the arrival pattern of these transactions and about the distribution of CPU service requests. We can then apply queuing theory to our model and derive equations with which to compute average CICS response time. Assume first that both the transaction interarrival times and their CPU service times can be approximated by a Poisson (i.e., exponential) distribution. These assumptions may be stated in shorthand queuing terminology as an M/M/1 queuing system. The first M (after Markov) signifies an exponential (or Poisson) distribution of interarrival times; the second M (also after Markov) represents an exponential distribution of transaction service requests; the 1 means that there is only one server, the CPU, in our model. Using these assumptions and Equation 1 from Reference 1, we have the means for calculating the average CICS response time:

$$T_c = \frac{t_c}{1 - U_{gc}} + \frac{t_h U_{gt}}{(1 - U_{gc})(1 - U_{gt})} \qquad (1)$$

where

T_c = average CICS response time
t_c = average CPU service time required by a CICS transaction
t_h = average CPU service time required by a high-priority transaction
U_{gt} = CPU utilization by high-priority transaction
U_{ge} = CPU utilization by combined CICS and high-priority transactions

T_c represents the time CICS transactions wait to be served by the CPU (queuing time) plus the time during which the transactions are being served by the CPU (service time). Missing from Equation 1 are two important components of response time: the effect of page faults within CICS and the time to perform nonpaging disk I/O. The following analysis shows how these effects can be factored into Equation 1. Each time the CICS transaction experiences a page fault, the entire CICS task must wait until the page is retrieved from the page data set. This is typically accomplished by a single 2K or 4K disk I/O. The time required to resolve a page fault is termed Page Delay Time, *PDT*. If we treat the disk that contains the page data set as the server and a page fault as a request for service, we can apply queuing theory again to solve for *PDT*. This time we treat the distribution of service time as being a general (G) distribution rather than a Poisson distribution, and denote the system M/G/1. The use of a general service time distribution rather than a Poisson distribution is based on empirical observations of disk service time characteristics.

From Reference 2 we obtain the following equation:

$$PDT = \frac{A_p E(t_p^2)}{2(1 - U_p)} + t_p$$

where

A_p = page fault rate within CICS
$E(t_p^2)$ = second moment of service times for page I/Os (also called expected value of t_p^2
U_p = average utilization of disk pack containing page data set
t_p = average I/O service time for the page data set disk pack

If the distribution of service times were truly random (i.e., a Poisson distribution), the second moment of service times for page I/Os would be $E(t_p^2) = 2t_p^2$.

Experience has shown, however, that disk I/O service times have a smaller variance than a Poisson distribution. This means that $E(t_p^2)$ must fall between t_p^2 (zero variance) and $2t_p^2$, or explicitly, $t_p^2 \leq E(t_p^2) \leq 2t_p^2$. If we split the difference, we can set $E(t_p^2) = 1.5t_p^2$. This leads to the following expression for Page Delay Time:

$$PDT = \frac{0.75 A_p t_p^2}{1 - U_p} + t_p$$

But since $A_p t_p = U_p$ (arrival rate times service time equals utilization),

$$PDT = \frac{0.75 U_p t_p}{1 - U_p} + t_p \tag{2}$$

The assumption concerning the arrival pattern of a CICS page fault must be questioned here. In truth a page fault cannot occur within CICS until a previous CICS page fault has been resolved. This means that the average gap between CICS page faults must be greater than the average service time for a page fault (PDT).

This characteristic of CICS page faults will have the effect of reducing the time that page faults are queued for service. Queue time is denoted by the first term of Equation 2. We can approximate this effect by making the following change. If we assume that U_p is defined as the utilization of the paging pack, excluding the utilization caused by CICS paging, we will reduce the calculation for the queue time component of PDT. Note also how in systems where page packs are dedicated to CICS (i.e., $U_p = 0$), Equation 2 simplifies to $PDT = t_p$. In other words, all queuing disappears when the pack is used only for CICS paging.

We can now calculate the fraction of time that the CICS system must wait because of page faults U_f as follows:

$$U_f = A_p \times PDT \tag{3}$$

If we lump utilization of the CPU by high-priority tasks U_h with the fraction of time the CPU is unavailable to CICS due to page faults U_f, we have the total fraction of time in which the CPU is unavailable to CICS, which can be used as the value of U_{gt} in Equation 1 as follows:

$$U_{gt} = U_h + U_f - (U_h \times U_f) \tag{4}$$

Notice that we reduce this sum by the product of U_h and U_f to eliminate the fraction of time that page faults and high-priority-task-busy periods are overlapped. This modification of the U_{gt} term in Equation 1 must also be reflected in a new calculation of the CPU utilization by the combined CICS and high-priority transactions U_{ge}. This is shown later in this paper in Equation 7.

One remaining parameter in Equation 1 is high-priority service time t_h. Although we do not know this value, we can use the value of page fault delay time as an approximation, that is, $t_h = PDT$. This is used because page fault delays comprise a large part of U_{gt}.

Finally, we must account for the time each CICS transaction waits to perform a disk I/O. Using the same assumptions made in calculating PDT, we arrive at the following equation for average disk I/O delay time:

$$DDT = \frac{0.75U_d t_d}{1 - U_d} + t_d \tag{5}$$

where U_d is the average disk pack utilization, and t_d is the average disk I/O service time. The average time a CICS transaction waits for disk I/O must be

$$IO = IOR \times DDT \tag{6}$$

where IOR is the average number of disk I/Os per CICS transaction.

The calculation of the CPU utilization by CICS and higher-priority work U_{ge} is given as follows:

$$U_{ge} = (t_c \times A_c) + U_{gt} \tag{7}$$

where A_c is the average CICS transaction rate.

We now combine equations and express the model in terms of the required parameters.

From Equation 2:

$$PDT = \frac{0.75U_p t_p}{1 - U_p} + t_p$$

From Equation 5:

$$DDT = \frac{0.75U_d t_d}{1 - U_d} + t_d$$

From Equations 3 and 4:

$$U_{gt} = U_h + (A_p \times PDT) - (U_h \times A_p \times PDT) \tag{8}$$

From Equation 7:

$$U_{ge} = (t_c \times A_c) + U_{gt}$$

From Equation 1 and using the $t_h = PDT$:

$$T_c = \frac{t_c}{1 - U_{ge}} + \frac{PDT \times U_{gt}}{(1 - U_{ge})(1 - U_{gt})} + IOR \times DDT$$

These equations reveal that the average CICS response time T_c is a function of the following parameters only:

- t_c = average CPU service time per CICS transaction
- U_p = average page data set disk pack utilization (excluding utilization caused by CICS paging)
- t_p = average page data set disk pack service time
- U_d = average disk pack utilization
- t_d = average disk pack service time
- U_h = average CPU utilization by tasks running at a higher priority than CICS
- A_p = average page fault rate within CICS

A_c = average CICS transaction rate
IOR = average number of disk I/Os per CICS transaction

Data collection

These parameters can be evaluated by examining reports produced by several software data gathering tools. CICS performance and utilization data are reported by two products, the CICS Performance Analyzer-II (PA-II) and by the CICS Performance Analysis Reporting System (CICSPARS).[3] The information reported by PA-II and CICSPARS includes:

- CPU time consumed by CICS
- Total number of transactions completed
- Average response time
- Total number of page-in operations

From these reports and further data reduction we can obtain such model input parameters as the following:

t_c = CPU time consumed divided by total number of transactions
A_p = total number of page-in operations divided by report interval
A_c = total number of transactions divided by report interval

The remaining parameters can be obtained from the disk subsystem reports produced by VSE/PT[4] (for DOS/VSE[5]), VS1/PT (for OS/VS1), or from RMF[6] reports (for MVS). Utilization of the CPU by high-priority tasks U_h can be derived by analyzing the workload activity reports in RMF. That technique, however, is not discussed in this paper.

example

Consider the example of a particular computing center that runs CICS/VS on an IBM System/370 and an IBM 3031 under control of the MVS operating system. Both paging and CICS data bases reside on IBM 3350 disk drives spread across two channel and control unit paths. Two sets of parameters have been evaluated and are designated here as Cases 1 and 2. Case 1 represents the environment when the 3031 contained 4 megabytes of real storage. Case 2 shows the model parameters for the same installation after 2 megabytes of storage had been added, i.e., to give a total of 6 megabytes of real storage.

Case 1 (4 megabytes of real storage):

t_c = 0.23 second
U_p = 0.24 (i.e., 24 percent busy)
t_p = 0.039 second
U_d = 0.14 (i.e., 14 percent busy)
t_d = 0.029 second
U_h = 0.12 (i.e., 12 percent busy)
A_p = 6 page faults per second

A_c = 2.2 transactions per second
IOR = 5 I/Os per transaction

Using the model to calculate the average CICS response time, T_c was predicted to be 2.33 seconds.

Case 2 (6 megabytes of real storage):

t_c = 0.23 second
U_p = 0.15 (i.e., 15 percent busy)
t_p = 0.035 second
U_d = 0.14 (i.e., 14 percent busy)
t_d = 0.027 second
U_h = 0.14 (i.e., 14 percent busy)
A_p = 2 page faults per second
A_c = 2.2 transactions per second
IOR = 5 I/Os per transaction

From the model, the predicted value of the average CICS response time T_c was found to be 0.99 second.

Thus the model predicted a 2.4-fold reduction in response time, i.e., from 2.33 to 0.99 seconds, with the addition of the 2 megabytes of real storage. The reason clearly lay in the reduction in page fault rate from 6 to 2. Except for the change in U_p and A_p, all other parameters remained the same or changed only slightly. There was good correspondence between the model's prediction and the performance as measured by PA-II. For example, the response time measured by PA-II for a peak hour each day, averaged for the week preceding the memory upgrade, was 2.9 seconds. The average response time for the week after memory was added was reported as 0.84 second.

System tuning and capacity planning

Before discussing capacity planning methodology, a consideration of some of the differences between system tuning and capacity planning might be in order. The individual who is assigned to tune a system generally has a different perspective than one who is doing capacity planning. The tuner is typically seeking to improve performance by means that generally do not require hardware upgrades. Instead, the tuner is concerned with balancing such system resources as storage, CPU, and disk subsystem, with the expectation of removing bottlenecks and optimizing performance. A typical system tuning effort might involve the reallocation of disk files to remove contention on one disk pack that had been causing excessive disk service times. After an initial system tuning, further tuning typically results in diminishing returns. Whereas an initial tuning effort might reduce the CICS response time by a second or more, continued tuning might produce reductions of only milliseconds.

Tuners often view the system as a given, a permanent entity upon which they are free to make only minor adjustments. Capacity planners are required to project current computing systems and their operations into the future, and to relate future computing requirements to the expected growth of the business. Thus, planners consider future workloads in terms of such things as adding hardware or shifting work to off-hours. With such a view, the planner may ignore any ongoing tuning activity and assume that the current system is the base from which to make forecasts. Of course, the same person may be both tuner and planner in a particular computing center. But by separating these functions into jobs with different perspectives, we can form a methodology to aid both tuner and planner.

Tracking systems

Both tuner and planner require a good understanding of their current computing environment. For example, how busy are the CPU and the disk subsystem, and who are the users? Tuner and planner also need to know how the system is performing, its response time stability, and whether minimum service levels are being met. The planner, however, must also be concerned with such trends as the rate of growth of the on-line transaction load.

These requirements point up the need for a tracking system that can monitor the key system parameters. The performance model helps in determining which parameters should be tracked. Since the model identifies the parameters that determine performance, we can begin by tracking these key parameters. As noted earlier, these parameters are reported within CICSPARS or PA-II together with either VSE/PT or RMF. Ideally, the reported parameters should be tracked for the identified peak hour each day, a function that can be automated to great advantage.

One of the advantages of a tracking system is that it allows an installation to identify easily its average operating environment. The average values of the parameters are referred to as their *nominal values*. The next section describes how the model can be used together with nominal values to identify performance bottlenecks.

Influence coefficients

The tuner's job is to identify the bottlenecks that are degrading system performance and eliminate them if possible. In a CICS environment, the analytic queuing model discussed in this paper can be used to derive a starting point.

Here we introduce *influence coefficients,* which relate a relative change in a dependent variable to a small fixed change in an

Table 1 Influence coefficients

	Parameters								
	t_c	U_p	t_p	U_d	t_d	U_h	A_p	A_c	*IOR*
Case 1: Four megabytes real storage									
Nominal value	0.23	0.24	0.039	0.14	0.029	0.12	6.0	2.2	5.0
Influence coefficient (×100)	4.983	0.560	2.252	0.009	0.070	0.708	2.145	4.120	0.070
Case 2: Six megabytes real storage									
Nominal value	0.23	0.15	0.035	0.14	0.027	0.14	2.0	2.2	5.0
Influence coefficient (×100)	2.352	0.035	0.255	0.019	0.153	0.413	0.218	1.527	0.153

independent variable. Let us define $IC(T_c/A_p)$ as the influence coefficient for page fault rate A_p on CICS response time T_c, or $IC(A_c)$ for short. In mathematical terms, the influence coefficient for page fault rate is expressed as follows:

$$IC(A_p) = \frac{T_c - T_{cn}}{T_{cn}}$$

Here T_{cn} is the nominal value of response time, and T_c is the value of response time calculated by the model when A_p is increased by one percent. All values of the model input parameters are at their nominal value, except for A_p, which is set to 1.01 A_{pn}.

Thus $IC(A_p)$ represents the fraction of change in response time caused by a one-percent increase in the value of CICS page fault rate. Since the model gives the expression for T_c as a function of several independent parameters, we can easily calculate influence coefficients for each parameter around some nominal point. If we use the values shown in Case 1 and Case 2 (from the previous example) as nominal values, we can calculate the corresponding influence coefficients shown in Table 1.

Notice how influence coefficients change as the nominal values change. Case 1 shows that CPU time per CICS transaction t_c, which is also a measure of average path length, has the largest effect on response time. This indicates that a starting point for tuning is reducing the CICS path lengths. Other candidates, in order of importance, are transaction rate A_c, page service time t_p, and page fault rate A_p.

Another factor to be considered is the possibility of reducing these parameters and by how much. It may be very difficult to reduce path lengths by a significant amount, whereas reducing page fault delays can be accomplished simply and by relatively large amounts. Page fault delays might be reduced by adding page data set disk packs, by moving a disk pack to a less utilized channel, or by reducing the multiprogramming level of the system. In MVS, there are techniques for providing a real storage fence around CICS to reduce page faulting.

Case 2 shows that as storage is added, the influence coefficient for page faulting A_p is reduced from 2.145 to 0.218. This indicates the sensitivity of influence coefficients to nominal values. Another observation on Case 2 is the lack of large influence coefficient values, a mark of a relatively stable system. If performance is thought of as floating on an exponential curve, performance in Case 2 appears to be sitting on the flat part of the curve.

As the characteristics of an installation change, the nominal values change, and the influence coefficient values change. An advantage of the model is that we can constantly recalculate the values of the influence coefficients to identify changes in performance sensitivity. With this information, the tuner can focus on those activities that promise the greatest return on investment in the system.

We expand our analysis through the use of Table 2. Influence coefficients are a measure of how sensitive a dependent parameter is to each independent parameter. In mathematical terms, the influence coefficient for parameter X_1 on dependent parameter Y is defined as follows: $IC(X_1)$ is equal to the percentage change in U when X_1 is increased by one percent.

If $Y = f(X_1, X_2, \cdots, X_9)$, then for a given set of nominal values for our independent parameters (X_1 through X_9) we can calculate a nominal value of Y. The calculation of $IC(X_1)$ is accomplished by raising the value of X_1 to $1.01X_1$ and calculating a new value of Y, called Y', and so on for all the independent parameters $Y' = f(1.01X_1, X_2, \cdots, X_9)$. Then $IC(X_1) = (Y' - Y)/Y \times 100$, that is, the percentage change in Y caused by a one-percent change in X_1. This calculation can be repeated to fill in the IC values for the remaining parameters (X_2 through X_9).

Note that for each calculation all values of X are kept at their nominal value except the parameter whose IC is being calculated. Table 2 shows the influence coefficients calculated for four different sets of model nominal parameters. The top row for each set represents the nominal values and the bottom row the influence coefficients for each parameter. Average response times as predicted by the model are given in the last column.

Table 2 Influence coefficients calculated for four sets of nominal values

Environment	*Nominal value and influence coefficient*	*IORATE (Number of physical disk I/Os per transaction)*	*PGSRV (Service time—seconds—per paging I/O)*	*PGBUSY (Average fraction of time a paging disk pack is busy)*	*DSKSRV (Average service time—seconds—for CICS disks—nonpaging)*
Four megabytes real memory	Nominal value	5.0000	0.0390	0.2400	0.0290
	Influence coefficient value	0.0698	2.2523	0.5598	0.0698
Six megabytes real memory (environment after the addition of 2 megabytes of real memory	Nominal value	5.0000	0.0350	0.1500	0.0270
	Influence coefficient value	0.2021	0.2110	0.0290	0.2021
December 1982 (best case)	Nominal value	5.0000	0.0350	0.1500	0.0270
	Influence coefficien value	0.0867	0.5392	0.0739	0.0867
December 1982 (worst case)	Nominal value	5.000	0.0350	0.1500	0.0270
	Influence coefficient value	0.0203	3.3467	0.4481	0.0230

Capacity planning methodology

We now consider ways in which the model can be used for capacity planning. As defined earlier, capacity is exceeded when the system cannot deliver a specified minimum level of performance. In a CICS environment this minimum is usually specified as a maximum average response time. Thus we must first establish a maximum acceptable response time.

example installation

The technique employed by one installation is the use of the tracking system discussed earlier in this paper. On days when user feedback indicated unacceptable performance, the planner would place a red "x" next to the day's entry. After tracking performance this way for several weeks, the planner was then able to quantify acceptable and unacceptable performance in terms of reported average response

DSK BUSY (Average fraction of time a CICS disk is busy —nonpaging)	*SYS OV (Average fraction of time the CPU is busy running tasks at higher priority than CICS)*	*CPUR (Average CPU time— seconds—required by a CICS transaction)*	*PF (Page-fault rate—average number of page faults per second experienced by CICS)*	*TRATE (Transaction rate —average number of transactions per second experienced by CICS)*	*RESP (Average response time —seconds— predicted by the model)*
0.1400	0.1200	0.2300	6.0000	2.2000	2.3327
0.0085	0.7078	4.9830	2.1450	4.1201	
0.1400	0.1400	0.2000	2.0000	2.2000	0.7495
0.0256	0.3245	1.7785	0.1713	1.0105	
0.1400	0.1400	0.2000	2.0000	3.3000	1.7476
0.0110	0.9376	5.7274	0.4935	4.8135	
0.1400	0.1400	0.2200	3.0000	3.3000	7.4800
0.0026	3.9979	30.2474	3.2912	29.0460	

time. The tracking of average CICS response time over a peak hour each day led to a performance value that could be expressed in the same units as that calculated by the model.

The planner then projected the installation's workload (expressed as transaction rate) into the next twelve months. Projections were based on the introduction of new CICS applications as well as a factor for estimated growth applied to existing applications. The growth rate was attributed to additional terminals and a growing familiarity with existing applications. Once again, the planner used the tracking system to highlight trends in transaction rate growth and used this information to make the projections.

Finally, the model was used to estimate average response times for each of the next twelve months. All model inputs were kept constant

Table 3 CICS capacity planning model projections for example installation

Month 1982	A_c *CICS transactions per second*	T_c *CICS response time* (seconds)		*CPU percentage utilization by CICS*	
		Best case	*Worst case*	*Best case*	*Worst case*
January	2.22	0.76	1.02	44.4	44.8
February	2.36	0.81	1.13	47.2	51.9
March	2.39	0.82	1.16	47.8	52.6
April	2.53	0.89	1.31	50.6	55.7
May	2.64	0.95	1.47	52.8	58.1
June	2.67	0.97	1.52	53.4	58.7
July	3.01	1.26	2.59	60.2	66.2
August	3.04	1.30	2.77	60.8	66.9
September	3.09	1.36	3.14	61.8	68.0
October	3.21	1.56	4.67	64.2	70.6
November	3.26	1.66	5.89	65.2	71.7
December	3.35	1.88	11.37	67.0	73.7

Example installation assumptions:	*Best case*	*Worst case*
t_c (CPU time per CICS transaction—seconds)	0.20	0.22
A_p (page fault rate—faults per second)	2.0	3.0

at their nominal values, except for transaction rate. This appeared to be a very optimistic outlook, since other model parameters probably would have increased as well. In fact, this approach was treated as a best-case scenario. The planner later modified two key model parameters to reflect a worst-case scenario. These parameters were t_c and A_p (CPU time per CICS transaction and page fault rate). The selection of these parameters and the amount they were increased was based on trends and variations shown by the tracking system.

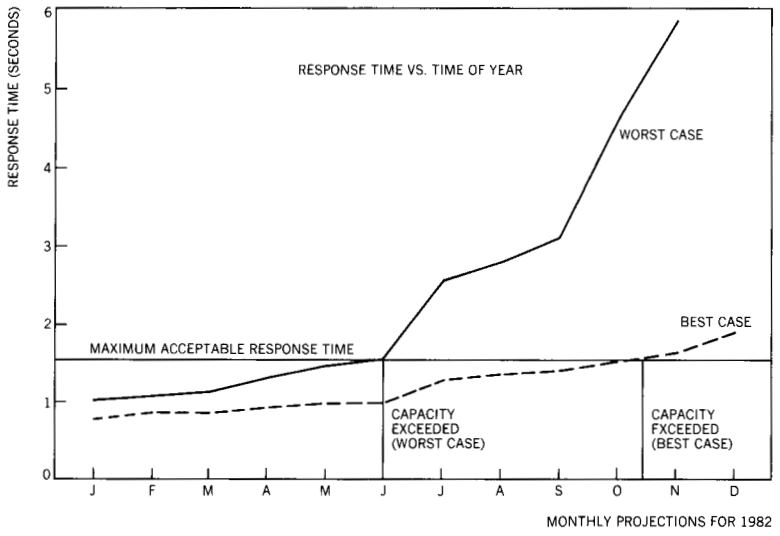
The tracking system had shown that the value of A_p varied between 1.5 and 2.5 page faults per second, with rare outliers reaching 3 page faults per second. Therefore, a worst case was defined with A_p running at an average of 3 page faults per second. The rationale for not changing disk subsystem parameters was the belief that ongoing tuning activity would keep these parameters relatively constant.

Transaction rates and best- and worst-case assumptions are shown in Table 3 for the example installation. The calculated response times and CPU utilization by CICS are also shown.

analysis of example

The projected response times were plotted as a function of time (month) for 1982, as shown in Figure 1. Perpendiculars through the intersections of the maximum acceptable response time and the best- and worst-case performance lines show the months at which capacity is exceeded for the two cases. The worst-case intersection projects

Figure 1 Example of monthly projections of CPU response time



that capacity is expected to be strained by June and exceeded by October (best case). The interval between the best and worst case may be looked upon as the time when action must be taken. That interval can also be considered as a buffer that smooths out the errors introduced by errors in the assumptions. If performance is critical to a business, corrective action should be taken before the worst-case intersection. If performance is less critical, action may be delayed until the best-case intersection.

corrective action

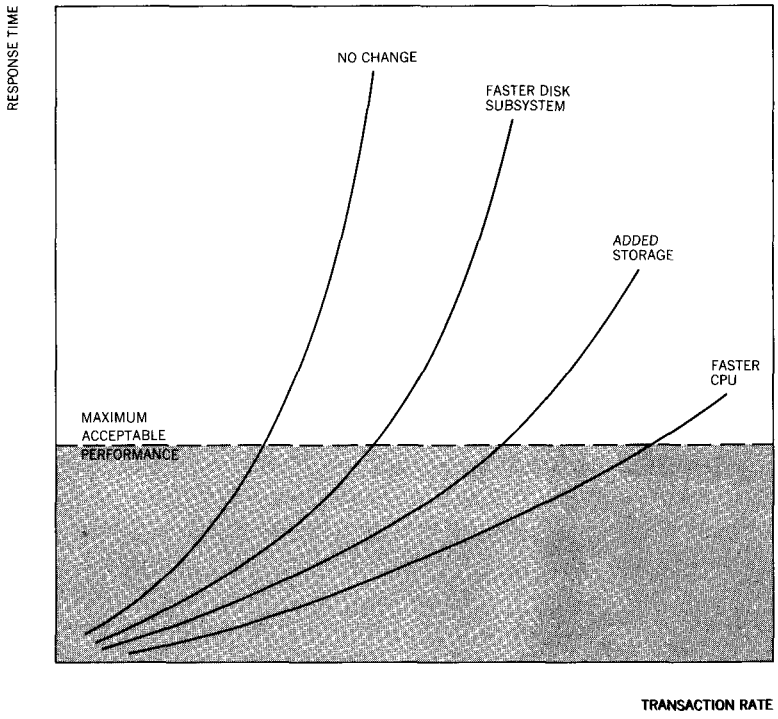
Having identified the limits of capacity, it remains for us to examine alternatives to correct the capacity problem. One possibility is to examine the influence coefficients at the points of intersection. Another approach is to list the options available to increase capacity, which might be the following:

- Install a faster CPU
- Add real memory
- Install faster disk drives
- Add more disk drives and channels

These capacity-planning-like options are more costly and thus receive more attention from upper management than the choices the tuner examined. Besides requiring capital outlay, these options may require a significant waiting period for delivery of the equipment. For these reasons, decisions must be made relatively early and must be based on sound analysis.

By translating these options into model input parameters we can use the model to examine each option. For example, a CPU of approxi-

Figure 2 Comparative alternatives from a system planning point of view



mately twice the speed of the installed machine would be expected to reduce the CPU time per CICS transaction t_c by one-half. We can also project a reduction in the high-priority task utilization U_h, although less accurately. A change from IBM 3350 disk drives to IBM 3380 drives with data streaming would certainly reduce both disk pack utilization values U_p, U_d and disk service times t_p, t_d. These can be approximated using the specifications of the new devices and by analyzing the measured values from the tracking system. Additional memory can be modeled by a sharp reduction in the CICS page fault rate A_p.

When these options are modeled, the results can be displayed as shown in Figure 2. These curves are produced by setting nominal values for each parameter, according to the option being studied, and then varying the transaction rates over a given interval. The value of this kind of display is that it shows clearly the relative performance and additional capacity obtained from each option. In this example, a faster CPU provides the best performance characteristics, but is also the more expensive option. Using this technique, we can combine the effects of multiple options and plot the combined effect to determine whether a combined solution can produce the required response time-transaction rate performance.

Concluding remarks

The following are the key points of the methodology discussed in this paper:

- Install tools such as PA-II or CICSPARS and VSE/PT, or RMF to collect performance and utilization data.
- Begin to track the key utilization and performance data, based on model parameters.
- Establish criteria for acceptable performance, in terms of peak-hour average response time.
- Project future workloads in terms of CICS transaction rates. Use the tracking system to analyze trends and variations.
- Use the model to identify times when capacity is exceeded, both best and worst cases.
- Use the model to examine alternatives to upgrade the system.

The value of the model is that it provides a tool that can be used within an installation as an ongoing process. Although the methodology presented here is fairly rigid, it is intended merely as a starting point. Each installation may have to tailor its capacity planning function to the availability and expertise of its personnel. The availability of a performance prediction model should encourage many installations to take a more active role in the important job of capacity planning.

Many of the ideas in this paper have been automated in a Field Developed Program (FDP) written by the author. This FDP is entitled *Capacity Planning/Performance Analysis for CICS* (5798-DKF).

ACKNOWLEDGMENTS

Much thanks to Joseph E. Flanagan, instructor at the IBM Systems Research Institute, for his knowledge and wit in making queuing theory so easy to understand. Thanks also to the staff of the performance analysis and capacity planning class at the IBM Information Systems Management Institute, Los Angeles. Many of the ideas presented in this paper have been derived from concepts taught in that excellent course.

CITED REFERENCES

1. L. Kleinrock, *Queueing Systems, Volume II: Computer Applications*, John Wiley & Sons, Inc., New York (1976), p. 125.
2. L. Kleinrock, *Queueing Systems, Volume II: Computer Applications*, John Wiley & Sons, Inc., New York (1976), p. 16.
3. *CICSPARS Program Description and Operations Manual*, SB21-2495; available through IBM branch offices.
   CICS Performance Analysis Reporting System (CICSPARS) program number: 5798-DAB.
4. *VSE/PT, Program Description/Operations Manual*, SH20-2171; available through IBM branch offices.
   VSE/Performance Tool (VSE/PT) program number: 5796-PLQ.

5. *VS1/PT Program Description and Operations Manual*, SH20-1837; available through IBM branch offices.
VS1/Performance Tool (VS1/PT) program number: 5796-PLG.
6. *RMF Version 2, General Information Manual*, GC28-0921; *RMF Version 3, General Information Manual*, GC28-1115; available through IBM branch offices.
RMF Version 2 (for MVS/370) program number: 5740-XY4.
RMF Version 3 (for MVS/XA) program number: 5665-274.

The author is located at the IBM Corporation, Two Jericho Plaza, Jericho, NY 11753.

Reprint Order No. G321-5176.