The transactions in a retail system are a comprehensive set of store applications. These applications include point-of-sale operations and back-room activities. Store performance is described by the throughput rate of transactions and the delay in the processing of any particular transaction. Two techniques of performance analysis are discussed: analytic evaluation and store simulation. An important consideration is the annual operating cycle of the store that indicates the amount and types of demands on the system.

Design and performance considerations for the Retail Store System

by M. A. Berk, C. W. Dunbar, and G. C. Hobson

The functional requirements and load demands placed on the Retail Store System vary widely not only across different retail establishments, but also within a given retail establishment. Variability across retail establishments is easily understood in terms of store size, type of merchandise sold, terminal placement (i.e., distributed throughout the store, centralized, or front-end checkout), use of distributed functional capability (e.g., merchandise ticket preparation functions spread among the store, warehouse, and host processor locations), and the general business and operational philosophy implemented within each establishment.

Seasonal variation within each retail establishment will cause variability of function and load on the system. Change in the importance of various functions is directly related to the regular merchandising cycle of ordering, receiving and marking, and sale of merchandise. Ordering in many cases occurs months ahead of receipt of merchandise. To minimize inventory, peak receiving precedes the spring and winter holiday sales periods. Thus, different elements of the cycle peak at different times of the year.

Therefore, to predict the performance of the system, careful attention must be given to the general environment of the retail establishment and the anticipated load variability on the system due to seasonal variation. To provide adequate system performance with the variability anticipated, the basic system was designed flexibly to fit the fluctuating requirements of each installation, in particular by providing system generation options that affect the allocation of storage and the allocation of microcode and data on the disk.

Both the variability in function and load and the design flexibility of the system complicate the performance analysis that is essential to provide an installed system that will adequately satisfy the store's requirements. A series of analytic and simulative tools have been developed to support the performance analysis portions of the system design process.

The intent of this paper is to provide insight into the process of system design and performance analysis for the Retail Store System and to discuss the techniques developed for performance analysis.

System environment

The Retail Store System provides the capability to control operations in a retail store from the time merchandise is ordered through the time that the merchandise is sold and customers' accounts are sent out for payment. The following store operations are typical functions in the Retail Store System:

- Sale of merchandise.
- Credit authorization of customer charges.
- Capture of customer charges for billing.
- Capture of merchandise sales for subsequent analysis and inventory control.
- Stock inquiry and reservation of warehouse inventory.
- Purchase order creation.
- Check of merchandise received against the original purchase order.
- Preparation of merchandise tickets.
- Invoice validation against merchandise received.

To meet the requirements for the selling floor, credit office, receiving room, and store management, a family of terminals, a store controller, and a central processor are provided.[1] Prior to discussing system design concepts and performance analysis, a brief discussion of the functions in the system is helpful in understanding the need for flexibility in its design.

The point-of-sale terminal performs such cash register functions as automatic accumulation of totals, calculation of change due, printing of cash receipts and sales checks, printing a journal of all sales transactions, and maintaining audit controls. In addition, the point-of-sale terminal performs calculations including multiplication, division, automatic tax calculations, discount calculations, and group pricing. As an interactive terminal, the point-of-sale terminal is used for on-line credit authorization. This interactive ability also allows automatic price look-up (where the

store controller supplies the point-of-sale terminal with the price of an item upon receipt of the stock number) and flexible interactive functions, e.g., warehouse stock inquiry.

The ticket unit operates on-line to the store controller, receiving ticket-preparation or ticket-reading requests from user-written programs within the store controller or at the host processor. In addition to program-requested ticketing, ticketing control is provided from the display station.

The display station is used for general keyboard communication within the store system (as opposed to the specific function of the point-of-sale terminal). It may be used on the selling floor for data entry and information retrieval, in the back room for invoicing, receiving, ticketing, etc., or as an administrative device in the office environment. It also serves as the main on-line communication device for user-written programs.

The store controller is the center of the Retail Store System. It performs the functions of terminal control, transaction logging, credit checking, and departmental totals updating. It also acts as the communications controller for all attached terminals and the message router for sending inquiries to, and receiving responses from, the host processor. In addition to these standard basic functions, it can also execute user application programs written at the central computer location, thus achieving flexibility for certain store functions.

The System/370 host processor is necessary to the installation and operation of the Retail Store System. At the host processor, user programs can be written to interactively communicate with the store controller for stock inquiry and reservation programs, positive credit checking, and interactive merchandise-processing applications such as purchase order entry or receipt data entry. Other user-written application programs can be executed at the host processor, using data collected in the store controller and transmitted to the host processor in a batch mode. The host processor also generates and maintains the store system in the store controllers. Basic system generation functions such as controller disk and storage allocation are performed by the host processor under a set of programs referred to as *subsystem support services*. Additionally, user programs designed for execution in the store controller are translated from macroinstructions, written under the subsystem program preparation support, to object code executable in the store controller and then transmitted to the appropriate store controller.

Performance objectives

System functions are divided between two areas: the selling floor and the back room.

The performance objective of the point-of-sale terminal is to provide cash register functions and complete point-of-sale data capture on the selling floor without limiting the input rate of the operator by responses from the system. Meeting this objective causes the throughput at any terminal to be limited only by operator capability. The system is said to be "operator bound" when this objective is achieved.

selling floor

For the functions that require the operator to wait for a response before continuing, such as price look-up and negative or restrictive credit checking, response time objectives developed from human factors studies have been established.[2]

In back-room operations, the performance objective of the ticket unit is to provide the capability to make or read tickets at a rate that will handle the throughput requirements of the specific retailer, whether the retailer has centralized or decentralized receiving of merchandise. The performance objective of the display station is to provide the capability to interact with user programs and complex data entry and inquiry routines, providing acceptable response times for these functions. Response time criteria have been developed from human factors studies.

back room

Specific performance objectives are generated for each installation by combining the general objectives stated above with the specific attributes and requirements of the installation. This permits appropriate cost/performance tradeoffs to be examined for each installation.

System generation

The system designer exerts control over system performance by (1) allocation of system functions between the controller and host processor and (2) preferential allocation of available storage and disk resources within the store controller to functions of assigned priority. These allocations permit cost/performance tradeoffs among controller resources, communications facilities, and the host, reflecting the inherent flexibility of a system designed with distributed intelligence. The specific requirements of the various merchandise-processing applications and their data bases, the complexity of processing requirements for local reports, and available transmission time all influence the allocations.

Once functional allocation is established, controller performance depends on the priorities assigned in storage and disk allocation during the system generation by the subsystem support services. The user specifies the selection of standard functions, the presence and priority of his programmed functions, and his disk requirements in terms of record organization, access technique, and

storage allocation

number of records. The creation phase of the subsystem support services generates up to five different storage configurations (maps) for selective Initial Microcode Loads (IMLs) at the controller. For example, the user could generate a storage map oriented toward point-of-sale for daytime operation and batch program processing for nighttime operation. He could alternatively set different priorities for ticketing operations dependent on anticipated concurrent point-of-sale activity.

This process will determine which of the microcode modules will be resident in storage and which microcode modules will be transient (residing on disk). Preferably, the most frequently executed modules should be resident. However, there is always some balancing required to provide satisfactory system performance.

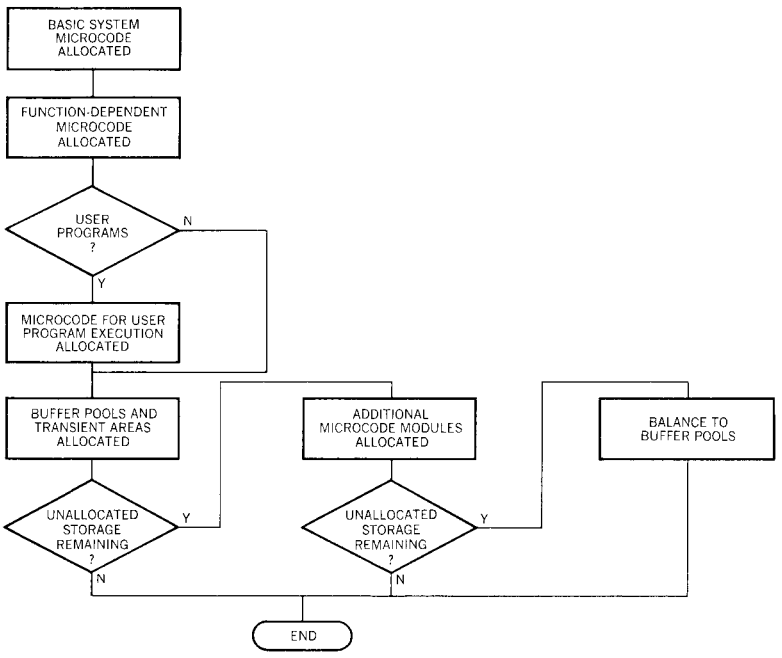
When storage allocation is complete, the transient modules are stored in a library on disk. The transient modules will be moved into transient areas of storage as they are needed for execution. The time required to load microcode modules is added to the execution time of a transient function, thus increasing the response time when a transient function is used.

An algorithm for allocating storage is shown in Figure 1. The basic system microcode modules are allocated first. Function-dependent microcode modules are allocated next. Thus allocation is dependent on the particular functions included in this IML. If user programs are to be executed, supporting microcode modules will receive space allocation next. The last basic allocation is for buffer pools and transient areas. The amount of storage allocated for these functions is determined by the store configuration and the amount of traffic predicted for the store by the user. If available storage has not been completely allocated, further allocations are now made of additional microcode modules in a prespecified sequence. Finally, if storage still remains after assignment of all specified microcode modules, the balance is distributed to the buffer pools.

The disk is divided into two areas, a drum-like area under fixed heads and an area under a movable head. Files are allocated to minimize expected disk service time. Disk performance is affected by the allocation of files to tracks under fixed heads and the allocation and organization of files to tracks under the movable head. The former is generally pre-established. The files under the movable head are allocated during the subsystem support services creation phase.

The number of files required and the size of each file will affect its location on the disk, which will affect the average number of cylinders traversed for disk access, influencing disk utilization.

Figure 1 Storage allocation



disk allocation

The disk is divided into 256-byte sectors. Each track contains 60 sectors and each cylinder contains two tracks. A disk containing either 5 or 9.3 megabytes may be used.

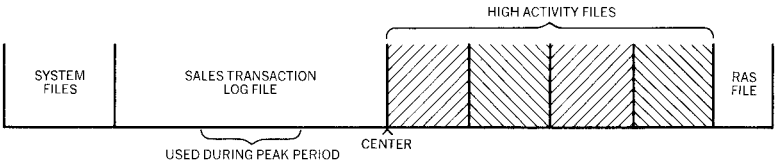
Three types of files are required for the Retail Store System:

1. System files contain the system storage maps for each IML, microcode libraries, diagnostics, and maintenance files. These files are normally assigned to prespecified areas of the disk.
2. Retail data files contain price look-up records, negative/restrictive credit records, batch print records, ticket unit records, and sales transaction log data for which the user specifies the number of logical records and the number of files required.
3. User files include the customer program library and data files required for user programs.

Three types of data organization are provided:

1. Sequential organization supports either fixed length or variable length logical records containing from one to 256 bytes.
2. Keyed organization supports either fixed length or variable length logical records containing from one to 254 bytes. A key is the argument for all file accessing. Record expansion is accommodated by using packing factors when calculating file size.

Figure 2 Disk allocation without second technique



3. Partitioned organization provides a means of maintaining microcode modules and user programs.

The file names and sizes are placed into a table, which the user has ordered by frequency of use as part of his store environment. The table is used to construct a disk map.

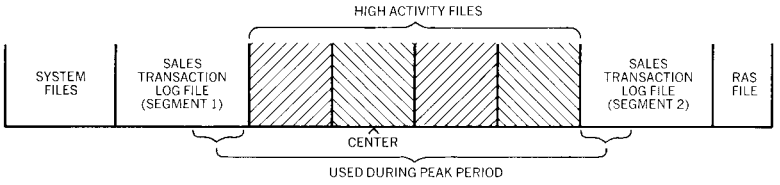
Disk service time consists of three components: seek time (disk arm movement), rotational delay (wait time for addressed sector to be positioned under read/write heads), and data transfer time (256 bytes from controller to disk or disk to controller). The only component that can be controlled is seek time, which is dependent on the number of cylinders over which the arm must move from file to file which, in turn, depends on the file allocation. On each system generation, two allocation techniques are used.

The first technique is based on frequency of use. The customer assigns appropriate priorities to his files, which are then allocated in order of decreasing priority to an increasing distance from the center of the disk.

The second technique takes advantage of the infrequent use of large sequential files, (e.g., sales transaction file), which are split into two segments. The two segments are allocated on opposite sides of the disk with the sequential access method reading from or writing to an outside track when the file is opened. Subsequent accesses will be progressively toward the center, allowing the disk arm to be stationed near the middle of the disk when the file is half full. This is the time of day when the store expects its highest frequency of use from the sales transaction file. The number of cylinders over which the disk arm must travel from file to file is bounded by a progressively smaller number. Figure 2 shows a disk allocation without the second technique, and Figure 3 shows a disk allocation using the second technique.

The algorithm for allocating files starts with the system files which are assigned fixed locations on the disk. The next files to be allocated are the large, infrequently used sequential files described in technique 2. The remaining files are allocated based on the priority given by the customer described in technique 1.

Figure 3 Disk allocation using second technique



The allocation starts at the outside and moves toward the center in a balanced fashion as shown in Figure 4. At the end of the allocation process, the files are shifted to close up any unallocated space as shown in Figure 5, moving unused space out of the high-traffic disk center.

Performance projection techniques

The following analytic and simulative system performance projection techniques were developed concurrently with the actual system design to measure its ability to satisfy the range of operational environments anticipated.

Alternate designs could be modeled to evaluate the adequacy of system resources prior to final commitment to a specific design. Then, before installing a particular system, alternate configurations could be evaluated and a preferred cost/performance balance established. Therefore, the same models initially used for system development could be utilized to evaluate particular installations.

The analytic techniques were designed to provide a rapid evaluation of system capacity versus projected load. Simulation techniques were developed to provide detailed insight into the internal characteristics of system operation. Both techniques were designed to accommodate the broad usage of functional applications and configuration choices anticipated.

Simulation techniques provide comprehensive determination of resource utilization, response time projections, and other performance attributes of the system, but the design detail included in these models requires large amounts of computer time to simulate much smaller segments of system elapsed time. While the results of analytic techniques provide less detail, they suffice for purposes of configuration evaluation. The analytic model is executed interactively on a terminal-based system and requires relatively small amounts of computer time for execution.

In general, the system developer requires the detail provided by the simulation model output in choosing a design. The system

Figure 4 Movement of allocation

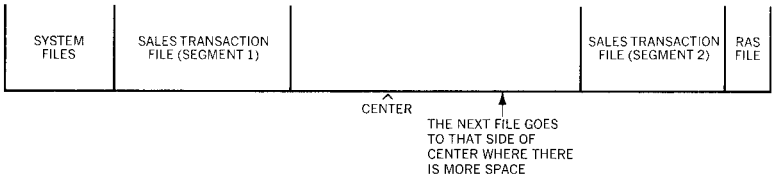
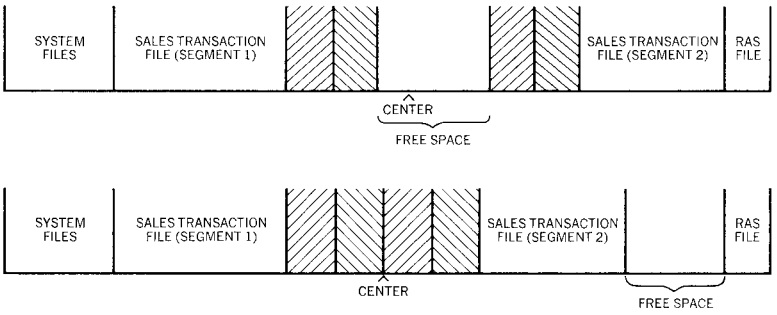


Figure 5 End of allocation process



designer evaluating configurations for installation recommendation generally needs only the level of information provided by the analytic model. However, it was convenient to select some features from each model for both purposes. Analytic techniques are required to emulate the storage and disk allocation algorithms implemented within the subsystem support services prior to performance evaluation by either modeling technique. Therefore, the analytic tool is used to generate a specific storage map and disk map as basic input to the simulation model or for further evaluation within the analytic model. Conversely, direct projection of controller utilization and certain component response times by analytic means is extremely complex. In those cases, a set of simulation runs could be executed and data points interpolated for use in the analytic model.

The remainder of the discussion will concentrate on the analytic techniques applied, emphasizing those that may be of general use beyond the retail system.

environment statement

The environment statement provides basic input to the analytic model in a form similar to the input requested by the subsystem support services at the time of system generation. The user defines his hardware configuration, choice of system functions to be included, and characteristics of user-written programs to be executed. The user will also specify certain options that will determine the specific set of microcode modules assigned to resident storage during the storage allocation process. Where system-generated files are to be included by selection of system

functions, the user need only specify variable characteristics such as number of records. The major file characteristics are prestored in the model. When the user defines his own files for execution, all characteristics must be specified. These characteristics include record size, number of records per file, file organization, and access technique.

The user is also asked to specify the traffic generated by each terminal and/or function and for characteristics of certain functions needed for detailed evaluation. As an example, the user is asked for information in ticketing dealing with the number of identical tickets per batch and for selection of ticket bursting characteristics by batch, line item, or purchase order. In point-of-sale operation, the user is asked for transaction distribution by transaction type, number of items sold per transaction, etc.

In essence, the environment statement is a comprehensive statement of system configuration and events anticipated in a specific operational period of the store. All information requested in the environment statement is stored for subsequent model usage. Evaluation of alternate configurations and/or alternate operational periods of the store are implemented by modifying the appropriate sections of the environment statement and re-executing the model.

storage and disk allocation

These sections of the analytic model calculate and allocate space according to input contained in the environment statement. The algorithm used is identical to the subsystem support services algorithm used during the actual system generation process. The model calculates the amount of storage required for each component allocated to resident storage and provides other generation detail required as input to the simulation model. The model also provides a listing of specific file size and location by file name. This information also provides direct input to the simulation model. The resultant storage and disk maps are stored within the model for use in the subsequent performance analysis.

disk utilization

Disk utilization for a given store environment is an important measure of the adequacy of the system to meet the concurrent application and traffic demands. In the design process, environment statements reflecting different seasonal peaks and functional mixes are evaluated to determine if a suitable performance level can be achieved in each case. The disk utilization and a summary of the disk activity for each file will indicate either satisfactory performance or the desirability of either reconfiguration or job rescheduling. The latter is primarily directed to backroom activity, since sales floor activity is determined by environmental circumstances.

Disk utilization is defined as the long-run proportion of time that the disk is being accessed. Disk access time includes movable head seek time and record search and read/write-check times under both fixed and movable heads.

Criteria used in setting limits for acceptable disk utilization are related to the queuing characteristics of the system that yield the response times to disk activity. The queuing characteristics of the disk are best examined by simulation.

Disk utilization is calculated by summing the individual contributing disk utilization of each system function. For example, if a function that uses an average of 45 milliseconds of disk time per access is being executed and the traffic rate generates 10 accesses per second, the contribution to disk utilization of that function would be 45 percent. The general equation can be written as:

$$U_d = \sum_{i \in I} S_{d_i} T_{s_i}$$

where

U_d = disk utilization

S_{d_i} = mean number of accesses per second for the ith function

T_{s_i} = mean service time per access for the ith function

I = the set of functions associated with the given store environment

However, the mean service time is dependent upon many factors including:

- File allocation under fixed and movable heads.
- File size and allocation under the movable head.
- Distribution of read and write accesses.
- Storage allocation.
- Traffic.
- Function mix.

The most difficult component of the mean service time to calculate is the average seek time encountered in movable head accesses. Head movement across the disk is determined by the size of, and relative activity to, each individual file (e.g., transaction log, credit authorization records, price look-up records, library, etc.) and the actual location of the file on the disk. Relative file activity is determined by the transaction mix (i.e., the distribution of cash sales, charge sales, price look-up items/transaction, voids, etc.). File size is determined within the system generation process and is based on the store environment statement. In

essence, the specific file allocation on the disk and the transaction mix determine the average seek time. It is important to note that while the transaction rate affects the total disk activity in terms of accesses per second, it does not affect the average seek time. The average seek time would be affected only by a change in transaction mix (assuming a given system file allocation).

To calculate the average total seek time the weighted seek time for each file is separately calculated and then summed.

Weighted seek time $T_{sk} = \sum_{i=1}^{n} W_i$

The weighted seek time for file i, W_i, is found by computing the sum of the time $\bar{t}_{ij}$ to traverse the distance between the file i and all other files j weighted by the probability of actually making that transition, p_{ij}, and multiplying this sum by the probability of being at that file, P_i.

$W_i = P_i \sum p_{ij} \bar{t}_{ij}$

We assume p_{ij} to be independent of i, hence equal to P_j and estimate P_i by

$P_i = S_i / S_t$

where

S_i = accesses per second to the ith file
S_t = total accesses per second on the movable head

$\bar{t}_{ij}$ is calculated by assuming head movement between the midpoints of file i and the midpoint of file j. In the case where $i = j$, $\bar{t}_{ii}$ is calculated by assuming head movement over one-third of the file if that file is organized for keyed or partitioned access. No head movement is assumed ($\bar{t}_{ii} = 0$) if the file is organized for sequential access. $\bar{t}_{ij}$ is calculated from the acceleration/velocity curve of the access mechanism.

The read and write access time to records under the movable head can be calculated as follows:

Movable read time $t_{mr} = t_{sk} + t_l + t_d$
Movable write time $t_{mw} = t_{sk} + t_l + t_r + 2t_d$

where

t_l = latency
t_r = rotational delay
t_d = time to read or write one sector

The read and write access time to the fixed heads can be calculated as follows:

Fixed read time $t_{fr} = t_l + t_d$
Fixed write time $t_{fw} = t_l + t_r + 2t_d$

The only variable in the above equations is t_{sk}. All other parameters are constants and are determined by the characteristics of the disk hardware. Once the store environment is specified and the disk map generated, t_{sk} is calculated and remains a constant.

The next step in the calculation of disk utilization is the counting and listing of all disk accesses in the transactions executed. While the microcode or user program specifies all data accesses, additional system-initiated accesses will occur and must be included. The most prevalent will be disk accesses due to load transient microcode modules or customer programs. The storage map must be reviewed to determine which microcode modules have been fixed in storage and which modules are transient. The most frequently accessed microcode modules are stored in the library placed under fixed heads. Therefore, suitable fixed head read accesses must be included for those transactions utilizing transient modules. User programs are generally stored in a program library placed under the movable head. Therefore, suitable movable head read accesses must be included. Additional accesses will be encountered also due to synonyms in keyed files. These will be significant only if the packing density of keyed files is high.

To complete the calculation of disk utilization, the system transaction rate is translated into accesses per second for each type of disk access by multiplying the number of accesses per transaction for each data, microcode, or user program access by the appropriate transaction (message) rate. The total disk utilization U_p is

$$U_p = A_{mr}t_{mr} + A_{mw}t_{mw} + A_{fr}t_{fr} + A_{fw}t_{fw}$$

where

A_{mr} = mean number of movable head read accesses per second

A_{mw} = mean number of movable head write accesses per second

A_{fr} = mean number of fixed head read accesses per second

A_{fw} = mean number of fixed head write accesses per second

It is important to note that the calculations above assume that the transaction rate is independent of system response time and throughput. This is generally true for point-of-sale functions and conversational back-room activity. This, in essence, is an alternate statement of the overall performance objective of the system (i.e., that the system be "operator bound"). However, in batch-oriented functions such as ticketing and batch print, disk utilization (and other resource utilization) is dependent upon system response time and resulting device throughput. This will be discussed later.

store loop utilization

Store loop utilization is another measure in evaluating a system configuration. As in the evaluation of disk utilization, store environments are analyzed. In this case, the analysis determines the level of usage of each of the three store loops. The model calculates the activity on each store loop and indicates the contributing utilization of each terminal (or set of terminals). The utilization of the system can be adjusted by modifying the number of store loops, by reconfiguring terminals in varying combinations among the store loops, or by job rescheduling.

Store loop utilization is defined as the long-run proportion of time that data is being transmitted. An alternate definition which is equivalent to the basic definition is as follows:

$$U_s = \frac{R_D}{R_T}$$

where

U_s = store loop utilization
R_D = aggregate data rate (characters per second)
R_T = maximum transmission rate (characters per second)

In the general case:

$$U_s = \frac{1}{R_T} \sum_{i=1}^{n} R_{D_i}$$

where R_{D_i} is the contributing data rate of the ith terminal on the store loop.

In calculating the data rate, control characters must be added to actual information characters transmitted and received. The contributing data rate is calculated by dividing the total number of transmitted and received characters generated during the transaction by the transaction period.

controller utilization

The definition of controller utilization is simply the long-run ratio of the average time used for instruction execution in a time interval to the total time interval. Analytic calculation of controller utilization is extremely difficult because of the complex microprogram data flow and multiple priority interruption design of the controller. Controller utilization can best be obtained by simulation. To maximize the usability of the analytic model, simulation runs are made for several store environments felt to provide a broad range of controller stress conditions. Controller utilization can then be interpolated for the specific store environment being evaluated by the analytic model.

response time

The disk, store loop, and controller utilizations affect the response time to transaction-generated messages. One or more queues are associated with each of the aforementioned re-

sources since each resource is shared by all message segments being processed within the system. Since the data flow is designed to optimize multithread processing, complex queuing structures are encountered in many points of the system.

Response time is determined primarily by the time spent waiting in queues associated with each resource and the time spent in execution within the resource (e.g., disk record access, store loop transmission). The time spent waiting in queues is the major component contributing to the variation in response times. The variability of the resource service time is related to the function being executed and the nature of the physical resource involved. Disk service time can be expected to vary considerably due to the random file processing under both the fixed and movable heads. Variability in store loop service time is generally small since the transmitted message sizes do not vary widely. Controller service time varies widely due to a broad range of path length variability and the priority interruption design of the controller hardware.

Priorities for all functions are equal in terms of disk processing and store loop transmission output. The disk queue is disciplined in a FIFO (first-in, first out) sequence. Input store loop priorities are established by the relative position of the terminal on the physical loop. Controller processing priorities are established in both hardware and microcode. The hardware priorities are ordered to assure timely data service for portions of the data flow with minimal buffering capacity. The microcode priorities are ordered to minimize the response time to point-of-sale-related messages.

The queuing characteristics of the disk may be calculated simply from basic single-server queuing equations due to the FIFO structure.[4] Simulation studies confirm the reasonableness of these assumptions. The nature of the polling and message concatenation technique used on the store loop and the complex hardware and microcode interruption structures of the controller render direct analytic queuing calculations impractical for these resources.

A hybrid technique consisting of both simulative and analytic steps is used to provide response time estimation for the store loop and controller. A range of store environments is studied in the simulation model and the resultant utilization and response time components for various message segments are noted. The response time component of the controller is derived in terms of controller utilization and priority. The response time component of the store loop is derived in terms of the store loop utilization. The derived curves are then analytically described and used for analytic calculations. The response time for a given message is

calculated by summing all resource queuing and service times encountered in the system data flow in processing that specific message. In executing the calculation, average queuing time for each resource is combined with the actual time required to service that specific message type.

throughput

Throughput and response time are inextricably related in batch-oriented functions. Throughput can be evaluated both with respect to a single terminal and to the total system. If a given function being executed at a terminal is single thread, i.e., there is no overlap between actions at the terminal and system message processing, throughput is inversely proportional to the response time. In many cases, however, terminal function is overlapped with system message processing. In essence, the achievement of the "operator-bound" performance objective is dependent upon a large degree of overlap between the salesclerks' actions and system message processing. The system is designed such that the response time is generally smaller than the parallel time expected for the salesclerk to complete the operation. In this case, throughput is determined primarily by the operator, hence the term "operator bound."

In the case of batch operations at a terminal, e.g., batch printing and ticketing operations, the degree of overlap may vary considerably. Ticketing throughput will be determined by both system response time and the mechanical capabilities of the ticket unit. The latter capability will control the situation when many identical tickets are being made; the response time will control the situation when many different tickets are made.

Calculation of ticketing throughput demonstrates a challenging problem in those cases where throughput is limited by response time. As discussed earlier, response time is dependent on disk, store loop, and controller utilizations. Conversely, the calculation of each of these utilizations requires knowledge of transaction rates which depend on throughput. This leads to an iterative calculation in which initial estimates need to be made for each resource utilization. Response time calculations are then made based on the assumed utilizations, and finally throughput is calculated to revise the utilizations for each of the resources. The iterative process is repeated until the utilizations converge. The analytic model calculates throughput on the basis of the procedures described.

The relationship between single terminal throughput and aggregate system throughput is determined by the terminal application and the degree of system overlap provided. In the case of point-of-sale, the aggregate transaction throughput is simply the individual terminal throughput multiplied by the number of terminals. In ticketing applications where throughput is determined

by system loading characteristics, increases in system load will cause a reduction in throughput due to the resultant response time increase. The analytic model reflects these factors in the throughput calculations.

Summary

The ability to make performance projections for the retail system requires a detailed knowledge of the range of retail functions that are executed within the system and a detailed knowledge of the internal design of the system. It is essential that the performance of the system be evaluated over a range of environmental assumptions anticipated in the annual operating cycle of the store.

Analytic and simulative techniques have been developed concurrently to evaluate performance. The two techniques are mutually dependent. The analytic model provides storage map and disk map input to the simulation model. Simulation studies are required to derive response time curves for input to analytic calculations. The resulting hybrid technique provides a very powerful set of tools and permits the use of the analytic tool as the primary evaluation technique for configuration evaluation. The simulation tool is the primary evaluation technique used in the system design and development process.

With complex system designs and continually changing environments, performance evaluation techniques are essential to determine limitations of system capacity. These tools are beneficial during the system development cycle and are basic to the system design and installation process.

CITED REFERENCES

1. P. V. McEnroe, H. T. Huth, E. A. Moore, and W. W. Morris, III, "Overview of the Supermarket System and the Retail Store System," in this issue.
2. D. C. Antonelli, "The role of the operator in the Supermarket and Retail Store Systems," in this issue.
3. *IBM 3650 Retail Store System Introduction*, GA27-3075, IBM Corporation, Data Processing Division, White Plains, New York (1973).
4. D. R. Cox and W. L. Smith, *Queues*, 54, John Wiley & Sons, Inc., New York, New York (1961).

Reprint Form No. G321-5005