System performance analysis techniques have been applied to support the development of a data-communication, data-base system. These techniques have been applied continuously from the system planning phase through system testing.

Computer simulative models and computer measurement tools were used in this analysis.

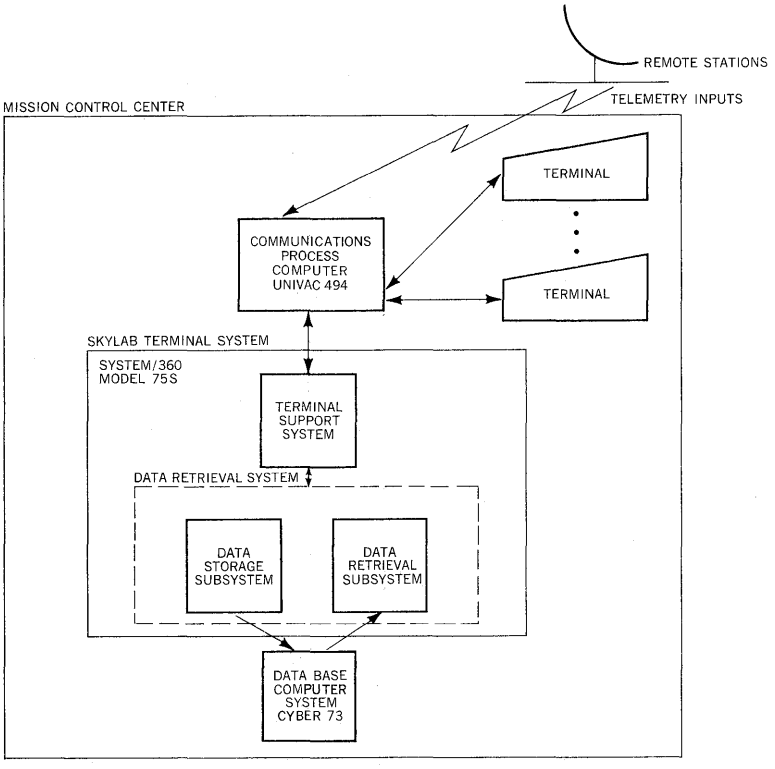
Performance analysis for the Skylab terminal system

by R. J. Mancini

The National Aeronautics and Space Administration (NASA) required computing systems for ground support of the Skylab space station project (1) for controlling the intricate aspects of manned space flight as in the Apollo missions, (2) for scheduling mission activities, and (3) for processing a large variety of specialized experimental data. The system for processing experimental data, the Skylab Terminal System, is discussed in this paper. This system permitted scientists and engineers to view telemetry data (i.e., data transmitted from the spacecraft by radio signals to receiving ground stations) after the data have been processed and formatted for output.

The Skylab spacecraft data were transmitted from the orbiting laboratory and received by remote tracking sites. From these sites, the data were transmitted over communication lines to the Goddard Space Flight Center for relay to the Mission Control Center in Houston, Texas. At Houston the data were received by a front-end UNIVAC 494 Communications Process Computer. This computer functioned as a message switch that routed the data to the Skylab Terminal System within one or more of the five IBM System/360 Model 75 computers in the Mission Control Center. In the Skylab Terminal computers, the data were routed by the Terminal Support System to the Data Retrieval System as shown in Figure 1. Data were transmitted to the CDC CYBER 73 Data Base Computer System for storage by using the Data Storage Subsystem. Data are retrieved from the Data Base Computer System in response to terminal requests

Figure 1 Interactive scientific system for the Skylab project



by using the Data Retrieval Subsystem. Scientific and engineering application programs were part of the Data Retrieval and Data Storage Subsystems. Included in this paper is an analysis of the Skylab Terminal System and its interfaces with the CYBER 73 Data Base Computer, and with the UNIVAC 494 Communications Process Computer.

The system software within the IBM System/360 Model 75 computer represents a three-year development project by IBM. The resultant large, complex system (approximately 2.5 million bytes of code) is required to receive and transmit large volumes of telemetry data according to time constraints concurrently with the requirement to responsively process terminal user requests. Therefore, continued performance evaluations were essential throughout the planning, design, implementation, and testing phases (summarized in Table 1) to determine whether the given hardware configuration and the software design satisfactorily meet the operational system objectives. Such evaluations benefit the developers through improved system design, and benefit the project through a reduction in the amount of development rework to meet system objectives. The intent of this paper is to

Table 1 System development and evaluation phases

Development phases	*Functions of phases*	*Performance evaluation*
Planning	Define requirements	System feasibility Configuration alternatives
Design	Develop architecture Develop baseline design	System performance verification Design feasibility
Implementation	Create integrated program system	Identify problems Compare implementation with requirements
System testing	Verify system meets requirements	Verify system performance Tune system

demonstrate the benefits of using performance evaluation techniques throughout a specific system development. Therefore, the paper uses examples that illustrate how these techniques can influence the software design and development rather than emphasizing the description of the application or the specific techniques used. The use of such techniques does not guarantee that all performance problems can be identified early and precluded. However, continuing system analysis does aid in producing a system more capable of meeting system requirements within the schedule constraints than otherwise might be possible.

Growing and changing requirements were a characteristic of the Skylab Terminal System development project. An example of this was that although the initial terminal capabilities were envisioned as limited, quick looks at incoming scientific (and trajectory) data, the final sytem had extensive terminal support. Included in the added support was the capability of entering batch requests for large amounts of processing, and Input/Output (I/O) from a terminal. The fact that application requirements changed points to an increased need for performance analysis of the system to evaluate the impact of such changes on the total system.

Both modeling and measurement techniques were used to support the performance analysis effort. The techniques that were used are categorized here according to the terminology definitions listed by Pomeroy.[1]

Self-driven simulative models were used during the project planning, design, and implementation phases. During each successive phase, more detailed application design information was included in the simulative models, ranging from the functional design level to the individual program level of information. Both Stanley[2] and Seaman[3] have illustrated some specific uses of this technique.

In the area of measurement techniques, both hardware and software monitors were used during the project implementation and

testing phases. Hook-catching software monitors[1] were used to measure Central Processor Unit (CPU), Input/Output (I/O), and main storage usage, as well as transaction response time. Activity was measured down to the task, program, and control program service levels. A statistical sampling software monitor[1] was used to measure Large Capacity Storage (LCS). Stanley,[2,4] Bonner,[5] Hobgood,[6] and Margolin[7] have illustrated some uses of these measurement techniques.

System planning phase

Every system development goes through a transition period during which raw ideas, needs, and concepts are formulated and translated into one or more general functional embodiments that satisfy the requirements. Even during this early system planning phase, we found it to be both possible and practical to make predictive evaluation analyses of proposed systems and, thereby, to make a positive contribution to the success of the project.

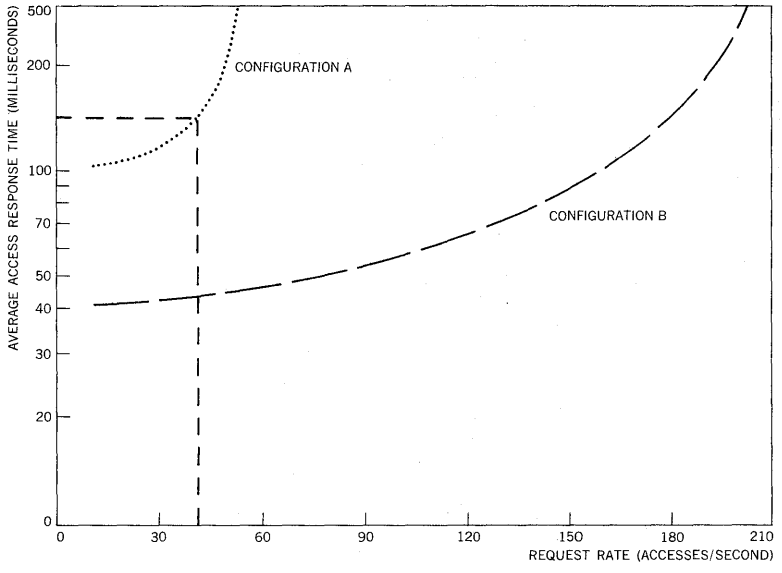
The scientific nature of the Skylab project established a requirement for an interactive terminal and data base system for the use of ground-based scientists and other mission personnel. Since a data base system was not available, much of the performance analysis work done during the planning phase was the evaluation of data base configuration options that were based, in turn, on a conceptualized system design.

Cost versus performance tradeoffs were used in deciding among data base configuration options. Because of a requirement to randomly access a data base of over one billion bytes, direct access disk storage was selected as the data base storage medium. It was also known at this time, however, that at least thirty scientific and engineering terminal users were to access the data base with an average expected response time of five seconds for the quick-look requests for data. The lowest cost configuration option was simply to add sufficient disk storage devices to the existing System/360 Model 75 computer systems. What was not known was whether this configuration wold be responsive to the terminal users. As a consequence, the following study of this option, compared to a more costly configuration option, was undertaken.

data base configuration options

Two principal data base configuration options (designated A and B) were considered. Configuration option A would obtain the necessary data base system at minimal cost by using existing System/360 Model 75s to manage the planned data base and by connecting additional disks through selector channels to make up the required storage capacity. The data base disks would then be switchable to any one of the System/360 Model 75 computers.

Figure 2 Data base access response time

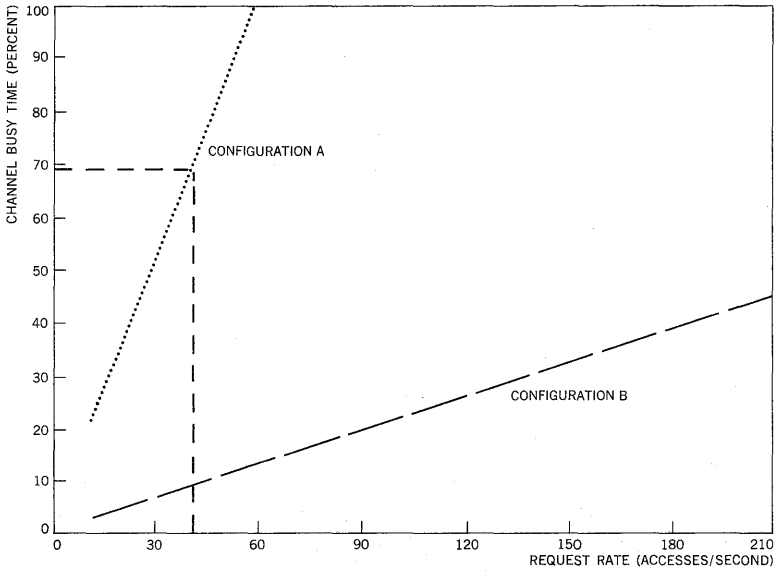


With the use of a two-channel switch, the disks could also be dynamically shared between any two of the computers. The choice among various types of disks was limited to those types that were compatible with the given computer system.

Configuration option B specified a separate data base computer that would be linked to the System/360 Model 75 by way of a selector channel. The B configuration would be a more open-ended approach relative to performance considerations. The disks could be selected for superior performance capabilities, since they would be required only to be compatible with the configuration B data base computer. Greater flexibility in channel configurations would also be available. Configuration B could thereby offer improved performance characteristics over those of configuration A, but only at added cost. A second data base computer would be required for a backup in configuration B because there would be no existing system to fall back on.

The approach used in our data base configuration evaluation had two basic parts: (1) to estimate and compare the expected performance of the data base hardware devices and configuration options; and (2) to estimate the expected performance of both configuration options relative to the terminal response requirements for the total systems. The data base hardware evaluation was necessary to gain insight into the advantages, limitations, and maximum data access capability of each data base configuration to handle a storage requirement of one billion bytes of data. Generalized disk usage assumptions, such as that of the use of

Figure 3 Data base channel usage



random seek functions, were made for the first part of the analysis. Also, an exponential distribution of data base request interarrival times was assumed in the hardware study so as to determine the overall capability and flexibility of each data base configuration. The scope of the second half of the study included the total data retrieval system, with a definition of projected terminal activity, to determine the distribution and frequency of disk activity. Simulation models were used in both parts of the study.

data base system evaluation

For the data base hardware analysis, a fixed data-request size and a variable request rate were used for each data base configuration. The maximum request rate capability for configuration option A was greater than fifty requests per second, but, as the request rate increased beyond forty requests per second, the request response time rapidly increased to a half second per request. The configuration A access response time function is shown by the dotted line in Figure 2. The steep increase in response time was caused by channel busy times that approached one hundred percent, as shown by the dotted line in Figure 3.

The request rate for configuration option B reached nearly two hundred requests per second before the response time approached one-half second. This request rate was nearly four times that achieved by configuration option A. For a request rate of forty accesses per second, the response time was 42 milliseconds for option B as compared with 135 milliseconds for option A.

There would have been little advantage to adding more disk packs to design A, because of the high channel loading (69 percent at 40 accesses per second). More disks could be added to Design B because the channel would be only 40 percent occupied at the peak request rate as shown in Figure 3. In design B the high utilization of the disks was the limiting factor. More disks, however, would provide improved performance.

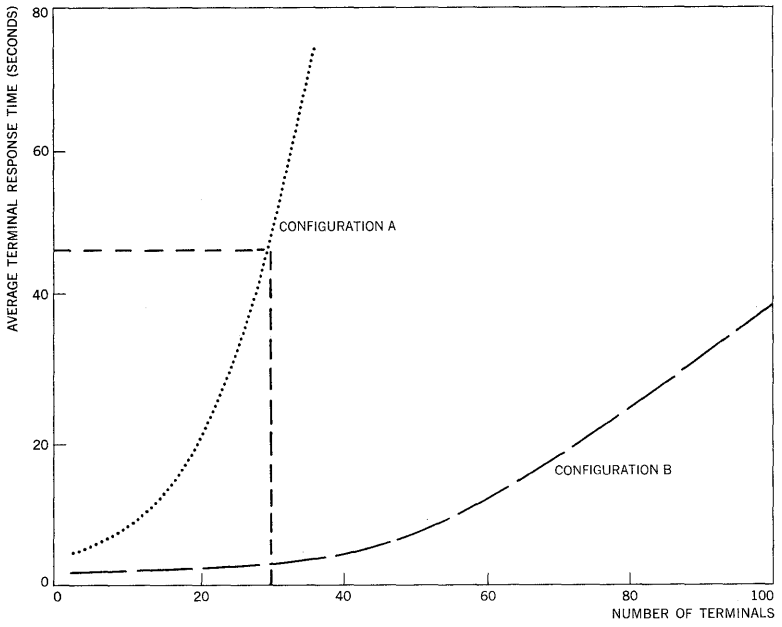
Two aspects of the configuration behavior are clear: First, configuration option B is always more responsive to data base requests than is configuration option A; and, second, configuration option B permits a far greater data base request rate than configuration option A, should the terminal workload require it. Since we have gained an understanding of the basic behavior of the two data base configuration options, it is necessary to decide which one to select for development. Even though data base configuration option A performs less well than option B, it is less costly and could be selected if it provides proper response to terminal users.

System evaluation information about terminal user requests and computer resources required to handle those requests was collected to evaluate the performance of the data base configurations in context with the total system performance. Only a best-estimate type of information could be obtained during the planning phase. A simulation model was designed to represent a gross computer system design that would be necessary to service the terminal workload. Further, the simulation model included representations of human interaction at terminals, transmission of remote data, and other factors that might influence total system performance.

The procedure to study the performance of data base configuration options A and B in the total system environment was that of a sensitivity analysis. The number of terminals to be supported by the system was progressively increased and the average response time was determined. The terminal response function for configuration option A is shown as a dotted line Figure 4 and as a dashed line for configuration B in the same figure. If the system supports 30 terminals with configuration option A, the average response time would be 46 seconds. This is unsatisfactory relative to the five-second terminal response requirement. On the other hand, configuration option B could easily support 30 terminals and still meet the response requirement. In light of this study, a decision was made to use a separate data base computer, as reflected by data base configuration option B.

The terminal response sensitivity study not only showed the resultant effect on the terminal user, but also provided insight into planning the organization of the data base. The initial con-

Figure 4 Terminal response sensitivity analysis



cept of organizing the data was based on the chronological sequence in which it was generated aboard the Skylab spacecraft. It was also assumed that there would be greater interest by users in the more recently acquired data. Our evaluation of the simulation results showed that data organized by time of origin could lead to potential performance problems with either of the candidate design configurations. Disks with the more recently acquired data would tend to be more heavily used than disks with older data. The predicted result was that there would be access contention for the recent data. Therefore, the plan to organize data solely by time was dropped to avoid such performance problems.

System design phase

With the aid of planning-phase study results, system planners were able to define the system in more formal terms. When the system developers received the formal system requirements, basic design work was begun. Although additional requirements would follow, the basic system architectural design could proceed, based on the set of baseline requirements derived from the planning phase. A major part of the system architecture within the System/360 Model 75 was planned to be a Terminal Support System. The Terminal Support System was to provide interactive terminal services (e.g., paging, terminal output queueing,

and temporary report storage) for the various systems to be supported. Another major architectural decision was to split the Data Retrieval System into two separate subsystems – the Data Storage and the Data Retrieval Subsystem. The Data Base Computer System would interface with these subsystems.

The Data Storage Subsystem was designated to process incoming telemetry data and transmit the data to the Data Base Computer for storage. The data would be processed to detect incorrect data points and would then be logically organized to facilitate retrieval. The Data Retrieval Subsystem would retrieve data from the data base and process it in response to specific terminal users' requests. Requested data would be processed by performing certain tests such as limit checking on the data points before doing special mathematical computations. Results would then be prepared for output in the form of tables or plots. To provide for data base integrety, only one system would be able to store in the data base; the majority of terminal users would be able to retrieve data only. This design would also provide the flexibility to run the Data Base Storage and Retrieval Subsystems in separate computers for possible load sharing or multijobbing with other applications.

Performance evaluation during the system design phase focused on system architecture and software functional design. The purpose of the analysis was to assist the designers in assessing the implications of their design decisions on software design adequacy relative to total system performance constraints. Furthermore, early identification of potential problems provided management with information on which to base decisions to change design, with the intention of avoiding the complications of making changes after designs had reached a firming-up stage. These evaluations covered such areas as task structure, disk I/O, and computer-to-computer interfaces. Analytical support was provided by a simulation model that included representations of the multicomputer configuration, the proposed system hardware and architecture, and the projected software design. System requirements for terminal inputs and data transmission from the remote sites were used to drive the model.

task structure

Evaluations of system performance revealed potential performance bottlenecks related to worker task management. *Worker tasks* service terminal user requests to generate reports via the Data Retrieval Subsystem. Analysis of terminal response times showed that, in a multiterminal environment, users might experience excessive response times. If the number of available worker tasks was low, requests from terminals would be delayed unnecessarily while waiting for a task to become free. This would be upsetting to the terminal user. On the other hand, a large number of tasks would place excessive demands on other system re-

sources. Since the final terminal workload and the precise computer resource available to the Skylab Terminal System were not known at this time, the final design provided an optional number of worker tasks to be created at system initialization time.

Another task problem showed up during the evaluation of telemetry message inputs from remote sites. Experienced judgement indicated that the amount of processing done by the Terminal Support System to handle input messages was large compared to the work accomplished. Detailed performance data showed that almost one half of the CPU resources were being used by the terminal task. This task had been designed to handle communications with each terminal, but it would serve no functional use for telemetry input messages other than routing them to the appropriate subsystem. As a result of this study, its inadequacy to handle high speed cyclic telemetry messages became obvious. The projected CPU usage was reduced by modifying the task design.

The Terminal Support System also had potential local disk I/O performance problems. This disk would be used to store completed reports and allow terminal users to retrieve them on request. The disk would also provide a temporary storage facility for each terminal user. Simulation results revealed that, with multiple terminals active, this disk could become the most highly used system resource. Disk-busy time was greater than eighty percent, which was an unacceptable design level. For certain terminal requests, disk I/O waiting time became the largest single component of overall terminal response time. The remedy provided early in the design process was the use of a larger data blocking factor, which would avoid significant device contention.

A two-task structure to facilitate temporary disk storage was developed for the Data Storage Subsystem to service the incoming telemetry data. Through this design, a system requirement could be met to examine the input telemetry data in context with the previously received data to filter out incorrect data points. A local disk, accessed directly by the System/360 Model 75, was used as a buffer to accumulate each group of data before the final processing could be done. The first pass task would do the initial processing on the messages and temporarily store them on the local disk. When a group of data were completed, the second pass task would do the final processing on the data and then transmit the data to the Data Base Computer for permanent storage.

disk I/O

It was found that the proposed design for the Data Storage Subsystem could have potential performance problems related to the flow of telemetry messages. An average of twelve messages per second for sustained periods of time was expected, but this rate could go as high as twenty-seven messages per second. To

assure the flow of messages through the system, an analysis was made of contention for the channel, control unit, and disks to determine whether the proposed system could handle the required telemetry rates. Results showed that the proposed design would be feasible if first a direct access method was used; i.e., if first, no indices were used, and second, if the telemetry messages were blocked (grouped together) with a minimum blocking factor.

computer interface

Potential performance problems were investigated for both the interface of the Skylab Terminal System in the System/360 Model 75 computer to the Communications Processor Computer and to the Data Base Computer System. The performance of these interfaces had to be understood in order to design the interface routines and establish computer interface performance requirements. On the Communications Processor interface, performance analysis showed that an output message handling convention then being considered would unnecessarily throttle the outputs from the System/360 Model 75 computer. This would be particularly critical in the case of a system requirement to transmit large volumes of data from Houston to Huntsville. The convention was that successive output messages could not be sent from the System/360 Model 75 computer destined for a particular terminal or remote site until a demand message had been received from the communications processor. A performance problem would arise if a heavy volume of input telemetry messages was being received at the same time the System/360 Model 75 computer was trying to transmit messages to another remote site. These longer telemetry messages would have caused heavy usage of the input channel to the computer, which could result in high channel contention between input telemetry messages and the demand messages needed to enable the output. This situation was subsequently corrected to give the shorter demand-message channel priority over the longer telemetry messages without any noticeable ill effects.

The design of routines used by the Data Storage and Data Retrieval Subsystems that would interface with the Data Base Computer System were also analyzed. One design possibility would be to project potential performance improvement on the basis that tasks within the System/360 Model 75 computer could continue processing while they had unsatisfied data base requests. The inclusion of such an overlap capability appeared to be desirable, but it would require a more sophisticated design. An evaluation was made to determine the potential performance improvements that would be attributable to the additional capability. The results showed that the design of the Data Storage Subsystem could be expected to keep up with the volume of input telemetry messages only if its data storage requests could be overlapped. In the case of the Data Retrieval Subsystem, certain

classes of terminal requests showed improvement when their I/O requests were overlapped. Some performance degradations resulting from increased device and CPU contention in the data base system would be expected, but the overall performance was projected to improve. On balance, the evaluation indicated the desirability of incorporating the overlap capability into the data base system interface design.

Implementation phase

After completing a preliminary design review, design implementation could begin. The goal of this phase was to create an integrated-program system that was ready for system testing. Along with the system implementation, tracking was required to be sure that system requirements were being met in a manner that was consistent with the system design and ultimately with a fully operational system. Changing operational requirements and system definitions also had to be dealt with throughout this phase.

During system implementation, detailed design information was available at the load module level. For example, the design of the Data Retrieval Sybsystem would now specify the various load modules required to support the required functions. Terminal requests would pass through the Terminal Support System and would then be processed by the message processor load module. If data were required from the data base, the parameter fetch load module would communicate with the Data Base System through interface logic to retrieve the data. As the data segments were retrieved, the parameter fetch module would search for and collect the requested parameters. Other load modules would then perform the various data manipulations requested and prepare the data for output via the terminal support system. An understanding of the functional relationships and design of each component was necessary to evaluate the system and to address the performance problems likely to arise during the implementation phase. A similar level of detail was also available for the Data Storage Subsystem.

This increased level of design detail was incorporated into the simulation model to describe load module structure and interaction. As components of the system were implemented, computer monitors provided current information on the use of such resources as the CPU and I/O by the load modules as they came online. These monitors also provided information on interactions among the load modules so as to verify the original system design and to understand the performance implications of these interactions. The appropriate use of the simulation model and the computer monitor made it possible to track both the projected

and the current uses of the system. We now discuss the evaluation of particular system functions during the implementation phase.

terminal activity

The terminal response time that a user would experience was the most visible means of assessing the effectiveness of the system design. A user forms his opinion of an interactive system on the basis of his expectations of terminal response time. Therefore, it was necessary to thoroughly investigate interactive system behavior. Since the terminal workload characteristic had been defined originally several months prior to this time, a comprehensive reevaluation was made of the intended use of terminals to access scientific and engineering data stored in the Data Base Computer System. We discovered that only one quarter of the terminal requests were expected to be the quick-look type; the other three quarters were to be batch processing requests that would involve significantly more CPU and I/O resources.

Results then showed that the System/360 Model 75 computer would become computation bound because of the high number of batch requests entered at terminals. Since all terminal requests were to be considered and handled with equal priority, batch requests significantly degraded the simulated system performance for quick-look terminal transactions. As system loading by batch requests increased, simulated response times grew to as high as ten minutes when, under more favorable conditions, they could have taken only ten to fifteen seconds. Such degraded system response was judged to be unsatisfactory to meet the needs of those terminal users.

Given the response problems, the Data Retrieval Subsystem design was reexamined. In addition, a study was undertaken to determine the characteristics of a terminal workload that still met the basic requirements of the terminal user but did not impose such a severe process load on the Skylab Terminal System. A new terminal load was defined to include the following characteristics: (1) only one-half of the terminal users would make batch requests; (2) the scope of the batch request requirements would be reduced; and (3) the number and frequency of terminals making quick-look requests would be increased so that the total number of terminal users being serviced would remain constant. Also, quick-look requests were simulated as being processed at a higher priority than batch requests. The system model was run on the basis of these new conditions, and showed system performance and terminal response to have returned to within reasonable ranges. Under these conditions, the model projected an average CPU utilization of less than fifty percent. At this point two recommendations were made: first, operational procedures should be initiated that would constrain terminal user requests,

so as to process the terminal workload in a single System/360 Model 75; and second, quick-look requests should be handled at a higher priority than batch requests.

LCS usage

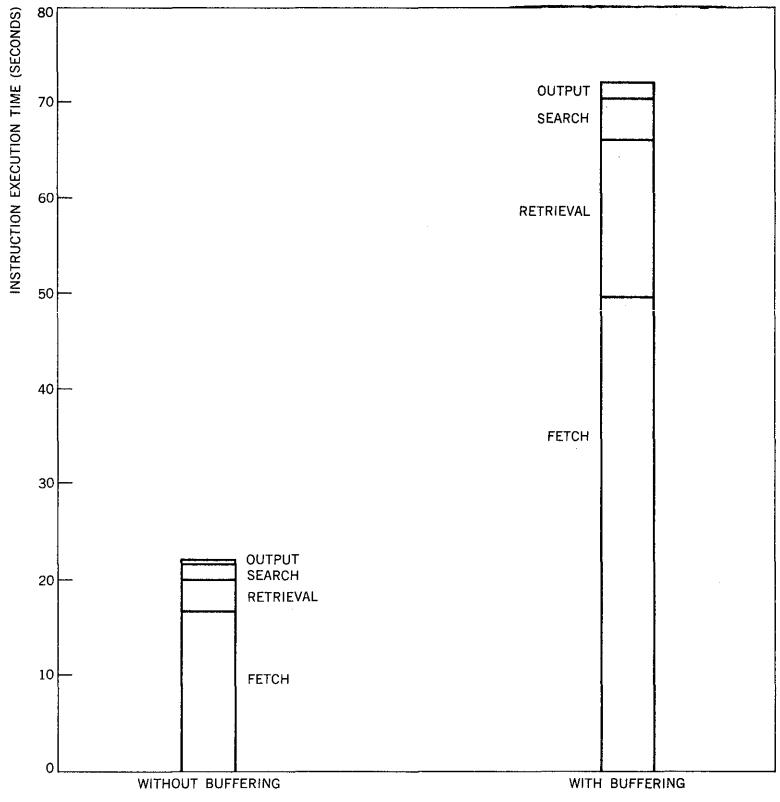
The advantages of using Large Capacity Storage (LCS) were the higher-speed peripheral storage capability available as an alternative to disk storage, and the extension of main storage at slightly reduced speed. Both load modules and data buffers could be placed in LCS. However, misuse of this capability could cause unnecessary performance degradation because the LCS cycle time was between four and eight times slower than that of main storage. The LCS evaluation effort provided general LCS usage guidelines. As the development proceeded, the specific LCS-related performance problems of each application were studied. Examples of some of the studies follow.

A general study was made of the ways in which specific LCS configurations affect program execution, and guidelines were proposed for avoiding performance degradation. Since these studies were basic to developing the system, they were covered in the system design phase. It was found that load modules should not be indiscriminately placed in LCS for execution simply to reduce main storage requirements. The best load module candidates should have exceptional characteristics such as low CPU usage relative to size, infrequent execution, and minimal response constraints. This study also provided designers with an easy-to-use analytical technique for estimating the increased CPU usage that would result from executing load modules in LCS or from placing data buffers in LCS. Although this analysis could only be used for rough estimates of actual increases in CPU usage, it was a valuable design guide and an aid in obtaining performance evaluation tradeoffs before the system components were implemented.

This LCS analysis information was used in the performance tuning of the Data Retrieval Subsystem to determine possible reductions in CPU usage due to improper placement of data buffers in LCS. A study was made of the major application load modules to determine the effect of LCS buffer placement. In the case of batch terminal requests that would require buffering, there was a tripling of instruction execution time, as shown in Figure 5. Information from the simulation of the frequency of execution of each load module, along with a breakdown of the components of each type of terminal request, were used to recommend the placement of specific load module buffers in main storage.

LCS analysis was also applied to a performance problem in the Data Storage Subsystem. In this case, there was a question of the system's real-time ability to handle the flow of telemetry data from the remote sites. CPU measurements had been taken from an early version of the system when there was a low message in-

Figure 5 LCS buffering of a batch terminal request

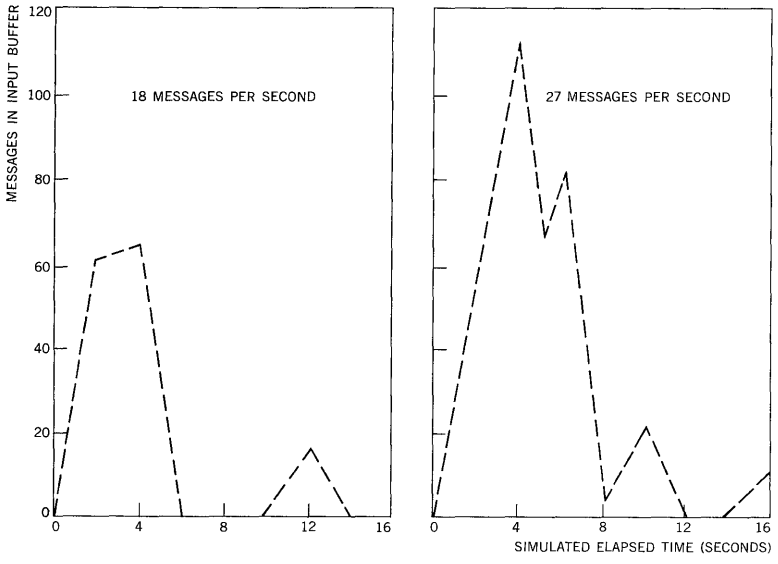


put rate from remote sites. Analysis showed that, at a nominal loading rate of 18 messages per second (an increased requirement), there would not be adequate CPU capability to handle the load. This was caused by the initial placement of buffers and tables in LCS because of their anticipated massiveness. To correct this situation, analysts and designers worked together to identify critical buffers and tables of reasonable size to be moved into main storage. Measurements were taken on the modified system, and the CPU usage was found to have been reduced to acceptable levels.

event analysis

The expected performance of the Data Storage Subsystem had to be continually tracked as the implementation proceeded, because of the requirement to handle a real-time flow of data from remote sites. An added complexity was the occurrence of such specific events as the selection of new sites, the discontinuation of other sites, and periodic deletions of data from the data base. Event analysis was done to determine the effect of these occurrences on the capability of the Data Storage Subsystem to handle real-time data flow. The objective of the analysis was to determine

Figure 6 Input message queuing during site selection



the performance of both a first-pass task and a second-pass task, since each was affected differently by the events being studied. Because of the requirement to transmit data to the Data Base Computer System, the analysis of the second pass task had to consider data base interface performance.

The purpose of the first-pass task was to store the input data on a local disk in an organized format. This established a rate equivalent to the flow of messages into the computer. The task worked on one message at a time. If the task was busy, additional messages coming into the computer would be placed in an LCS storage buffer. The number of messages queued in this buffer could not exceed the capacity of the buffer or there could be a loss of messages. (Manual intervention would then be required.) Since loss of messages had to be avoided, events—such as site selection—that required the exclusive use of the first-pass task were a particular concern of the analysis.

The analysis revealed that site selection was the most critical event that could affect the performance of the first pass task. A method of studying the dynamic nature of site selection was to simulate the event in a multitasking system environment with typical message rates (18 messages per second) and the worst case (27 messages per second). Results of site selection simulation in terms of the buildup of messages in the input buffer are shown in Figure 6. We concluded from the analysis that a significant buildup of messages could be expected during site selection. This buildup was used to establish the size of the input

buffer. We also concluded that the task as designed could work off the buildup of input messages and store them on disk well before another site selection event.

The purpose of the second pass task was to retrieve data from a local disk, do secondary processing, and then transmit the data to the Data Base Computer System. The execution of this task was asynchronous with the input message rate in that data were retrieved only after a complete grouping of the data that were to reside there. Because of this type of execution, a separate analytical approach was needed to determine whether the design could maintain the over-all input flow through the system. By simulating the design in a typical multitasking environment, it was found that the second pass task, once activated, could service input data at a rate equivalent to 33 messages per second. This rate appeared to be satisfactory because it was well above the worst-case message loading rate. However, there remained a question of performance relative to such events as site selection, site discontinuation, and data-base deletion. A determination was made of the rate at which the second pass task must handle messages to be sure that the system could service the nominal rate of input, even when these events were occuring simultaneously. To answer this, various events were simulated to provide information needed to establish a design acceptance rate at which the second pass task must be able to perform. This acceptance rate was set at 30 messages a second. The then current evaluations showed that, in a normal case, the design could exceed this performance acceptance rate.

System testing phase

System testing was the final and independent evaluation of the Skylab Terminal System in a near-operational environment. All components had to be integrated into a working system. Acceptance testing was also performed to be sure that the system satisfied all the functional requirements. During this period, future operating personnel were being trained. Because additional system capabilities being implemented much of the testing period was overlapped with development and integration activities.

Performance is key to testing because a system should perform the way the user requires. System performance monitoring and evaluation in parallel with system testing can answer performance questions. During the testing phase, much of the performance analysis activity was directed toward the tracking of known potential resource problems, checking out specific problems, and analyzing operational procedures that might affect performance. A study of the operational environment was made to determine the compatibility of the subsystems with other applications in Houston that would be sharing the same resources.

Tracking the CPU usage of the Data Storage Subsystem was of particular importance because of the requirements to handle input messages in real time. The projected CPU usage also varied with the input message rate and the specific type of data being transmitted. Analysis of the first system test series showed that CPU usage would increase by an unexpected 30 percent at the nominal message input rate as compared with previous projections.

CPU usage

This CPU usage increase was traced to extensive use of a special facility to determine the Greenwich Mean Time for tagging input messages. Time tagging had required little programming effort, and it had quickly become part of the system. But this minor modification had been made without considering its performance implications. Analysis of the time tagging method revealed that the facility to determine time had not been designed to be executed at a rate equivalent to the input message rate. Further investigation revealed that this time tagging was a duplication of a similar time tagging already being done by the Terminal Support System with much less CPU usage per tag. As a result of the analysis, the system was modified so that the original time tagging operation could serve the new requirement as well. Although the solution was obvious after the problem had been discovered, the implications of a seemingly minor programming change can easily be overlooked in a large development effort. System performance tracking during system integration, and testing can be a safety check on such changes.

CPU usage projections for the Data Storage Subsystem were based on early measurements of a particular type of data. Therefore, it was necessary to determine whether the CPU usage would vary significantly with other types of input data. Analysis and measurements resulted in CPU projections at nominal loadings for various types and mixes of data. The projected CPU usage for one type of input data was high enough that special operational restrictions were established to handle it.

One of the functions performed during acceptance testing was to identify discrepancies so that they could be corrected. System funcitonal capabilities and responsiveness to users' requests were both assessed. In one case, computer measurements showed that two load modules were looping—giving control to one another—and thereby causing excessive CPU usage. Corrective programming was initiated as a result of analysis.

problem checkout

A different type of performance problem arose with an application that was designed to send a large volume of data over a transmission line to a remote location. The application controllers said they could not drive the transmission line at the capacity they wanted. Three major causes of performance degradation

were identified and the degree of each degradation was determined. Inefficient coding and program logic errors were found and corrected. Also, the then current method of display output that monitored the progress of data transmission was found to be reducing system performance. An immediate solution was a relaxation of a stringent cyclic display update requirement. A long-range solution that was implemented was to redesign the display task structure. It was also found that the computer to computer interface response time was limiting the transmission efficiency.

operation aspects

An analysis of certain of the operational procedures of the system showed that the user himself could cause performance degradiation. In the Data Storage Subsystem, the user could indirectly cause cyclic displays to be generated in excess of those required, thereby increasing CPU usage unnecessarily by five to ten percent. Operational procedures that induced the extra use of the CPU were subsequently modified to eliminate such problems. Another operational procedure that involved the logging of input and output messages was analyzed. Computer resources required for the logging option were identified and put in perspective with the use of those same resources for vital processing. With this information, the user could evaluate his need to log data during critical time periods against the total system objectives. A decision was taken to eliminate data logging during critical mission support periods.

Investigations were made of all applications using the five available System/360 Model 75 computers, to determine their most efficient usage. Both the data storage and data retrieval subsystems had to be analyzed to determine the ability of each subsystem to run with one or more other applications in one computer. Prime areas considered in the study were computer resources, modes of operation, and performance requirements of each of the applications. Results of this study were presented in a co-user matrix that showed which modes of operation for each application were compatible with other applications. Supporting information was accumulated to back up this matrix, such as resource and performance requirements for each application and each proposed combination of applications. The co-user matrix provided an easy-to-use source of information to develop computer scheduling options and alternatives.

Performance evaluation done in parallel with testing aided in producing a system that was highly tuned to the user's operational requirements. It also provided a method of answering the user's questions about hypothetical or actual performance problems. Computer monitors were the principal tools used for the performance evaluations. Simulation models, when used, played a more limited role when the operational system was available for testing. Simulation models would be valuable in the system

testing phase to simulate operational environments that could not be tested directly, such as the full complement of terminal users who would want to use the system.

Concluding remarks

Analytical techniques applied throughout the development cycle can contribute significantly to the development of a successful computer system. In this paper, we have tried to illustrate the value of doing the appropriate level of performance evaluation at each stage of a development cycle. The particular types of techniques used depend on the level of complexity and performance considerations associated with a particular project. A simple pencil and paper approach with observations of program execution may suffice for a small project. On a large system development project where the performance of more complex computers and other resources are critical, more extensive techniques are usually required. Here it may be necessary to judge the adequacy of system design, details of software design, and computer configurations in the expected total system environment. The analyst, using the techniques of digital simulation models and computer system monitors, can take this perspective. This was the course followed in the development of the Skylab Terminal System described in this paper.

The preventive nature of performance analysis often makes it difficult to assign actual cost savings for performance problems that are avoided. This paper has tried, through an illustrative system development example, to show how one can improve the visibility and control of a system development effort by applying performance evaluation and analysis. Experience with the Skylab Terminal System shows that system developers can depend on predictive techniques and the kind of analyses described to guide complex system development efforts.

ACKNOWLEDGMENT

The author is pleased to acknowledge the supportive assistance of his colleauges who have who have contributed to the overall system evaluation effort discussed in this paper. He especially wishes to acknowledge Wayne Stanley.

CITED REFERENCES

1. J. W. Pomeroy, "A guide to programming tools and techniques," *IBM Systems Journal* **11**, No. 3, 234–254 (1972).
2. W. I. Stanley and H. F. Hertel, "Statistics gathering and simulation for the Apollo real-time operating system," *IBM Systems Journal* **7**, No. 2, 85–102 (1968).

3. P. H. Seaman and R. C. Soucy, "Simulating operating system," *IBM Systems Journal* **8**, No. 4, 264–279 (1969).
4. W. I. Stanley, "Measurement of system operational statistics," *IBM Systems Journal* **8**, No. 4, 299–308 (1969).
5. A. J. Bonner, "Using system monitor output to improve performance," *IBM Systems Journal* **8**, No. 4, 290–298 (1969).
6. W. S. Hobgood, "Evaluation of an interactive-batch system network," *IBM System Journal* **11**, No. 1, 2–15 (1972).
7. B. H. Margolin, R. P. Parmelee, and M. Schatzoff, "Analysis of free storage algorithms," *IBM Systems Journal* **10**, No. 4, 283–304 (1971).