Enhancement of the computing in an engineering environment by the installation of a virtual machine time-sharing system is discussed. This installation has been particularly useful in allowing the engineer to make the computer an integral part of a design cycle through the interactive use of graphic displays.

Described is a CP-67 system implementing the virtual machine concept. By using an operating system of his choice in his own virtual machine, the engineering user has great flexibility in the development of applications..

Virtual machine computing in an engineering environment

by M. McGrath

In the past few years, the virtual concept in computing has been used with several time-sharing systems.[1,2] There are several reasons for this recent trend, the most important of which is to provide a multiaccess computer system that can be shared at the same time by many users, including personnel not familiar with the details of data processing.

The requirements of data processing in the engineering environment have expanded into many different areas and disciplines during the last four or five years. This is in contrast to the first twenty-odd years of scientific computing that were characterized by single-CPU, single-thread, batch job streams. The introduction of the System/360 family of computers and multiprogramming was to some extent just an extension of the batch concept. Multiprogramming enabled the computers to perform better from a systems point of view but had no effect on the philosophy of the engineering user. Multiprogramming permitted the development of interactive programs that provided more flexibility for the engineering user but still did not offer a practical method for a large number of users to have access to different computer systems simultaneously.

Through time-sharing, computing for engineering is evolving into a system meeting the diverse needs of a variety of engineering personnel in real time. The engineer has been brought

closer to the computer through interactive program execution at his own terminal. Thus, some of the traditional and frustrating delays involved in the batch-processing mode have been alleviated by a man-computer relationship that enhances an engineer's problem-solving ability. The availability of time-sharing systems has also made it economical for the programmer to benefit. Development, debugging, and execution done interactively on these systems have reduced program development time. In addition to the user-oriented benefits, advances have been made in the management of computer resources such as use of less main storage and greater utilization of the CPU.

In this paper, CP-67 (Control Program 67),[3] is described as implemented in a large-scale engineering environment at Pratt & Whitney Aircraft Division of United Aircraft Corporation, a manufacturer of jet engines for many military aircraft and commercial transports. The installation is viewed from the aspect of its use of virtual machines and the resulting performance. Several engineering applications are used to illustrate the usefulness of the virtual machine approach.

System approaches

In 1968, the computing requirements of the engineering department at Pratt & Whitney Aircraft fell into several broad categories. One main requirement was to develop a graphics capability for the engineer using multiple cathode-ray tube displays, namely IBM 2250 display consoles, and the Operating System FORTRAN Graphics Subroutine Package (GSP).[4] The other major requirement was to provide the scientific programmers with an interactive programming system better suited to their needs than the RAX (Remote Access Computing System) they had installed on a System/360 Model 40.

The installation of a graphics system places many diverse demands on the computer system, and these may be characterized as three distinct states of the engineer and computer relationship. First, the engineer may be in a think mode – time he uses to consider which of several approaches to further explore based on the current graphic display. During this time, which may be a considerable portion of the elapsed time of a day, absolutely no demands are being placed on the CPU. The second state involves the interactive process in which a user interacts with the computer system by means of a light pen, an alphanumeric keyboard, or a number of program function keys. The engineer may be altering the shape of a display picture, changing his input data, viewing his output data, or selecting the next program to be run. This state puts very little demand on the CPU, but does require that when CPU time is requested as a result of a graphics interruption it be serviced as soon as possible. The third state is a pro-

gram execution phase in which it is desired to run a noninteractive portion of the program once the input data and shapes have been defined interactively. This state has a heavy demand for CPU time.

In the conversion from a batch to an interactive environment, the availability of main storage was a major factor that had to be considered since the graphics programs were essentially large batch jobs that had graphics capability added to them. In a batch environment, these programs could be run sequentially in a single region of 250K of main storage, but in an interactive system, it was estimated that 1000K of main storage would have to be dedicated to satisfy the storage requirements of three interactive display console regions.

MVT approach

To meet these engineering requirements, several possible systems were investigated. The first was a traditional batch approach with different configurations of the System/360 Model 65 being considered. The system would run under the MVT (multiprogramming with a variable number of tasks) option of Operating System/360 (OS/360), and would have many advantages. It would be compatible with the other computers in the installation, minimizing problems in operations and making backup possibilities more realistic. Installation of OS/360 would be eased by the considerable OS/360 expertise built up by the computing staff. In addition, MVT has a simple, straightforward scheduling system in which CPU time can be allocated on a strict priority basis making it possible to ensure satisfaction of all CPU requests from any graphics user before a request from any other task. This priority system would help ensure a minimum response time for all interactive graphics users taken collectively. However, the advantages of installing MVT had to be weighed against its limitations. Also, prior to TSO (Time-Sharing Option), OS/360 did not have a replacement for the RAX system that would provide the desired interactive programming facilities.

TSS/360 approach

The Time-Sharing System/360 (TSS/360) was also considered because it addressed many of the problems encountered in using a batch system. TSS/360 is a sophisticated time-sharing system designed for the Model 67 and has many facilities that make it very appealing to the time-sharing user. Among these is the table-driven scheduler that controls the task management functions in a paging environment and can be adjusted to meet the requirements of most installations. Also of note is the data management system provided by the virtual access method and designed specifically for a time-sharing system. TSS/360 was designed to encompass the widest range of computer facilities and satisfy simultaneously the batch, time-sharing, and data base requirements of an installation. Although TSS/360 has made advances in time-sharing systems, its ability to meet the requirements of the engi-

neering environment discussed here was limited by several factors. First was the lack of support for the 2250 displays, which was the main requirement of the engineering system. Other limiting factors include its incompatiblity with OS/360 and its complex structure that would have prevented early installation of the system and delayed development of the graphics applications.

chosen approach

The System/360 Model 67 hardware and CP-67 were chosen as the system that would best meet the interactive engineering requirements mentioned previously. In this system, the necessary support for the display consoles in available, real main storage is not a problem since a virtual memory technique is used, and the Cambridge Monitor System (CMS)[5] provides the required interactive programming facility. CP-67 will also dynamically allocate CPU time in a manner that will give a higher priority to the interactive user who requires a quick response time and a lower priority to the compute-bound user who can tolerate some delay in his program execution.

Graphics with CP-67

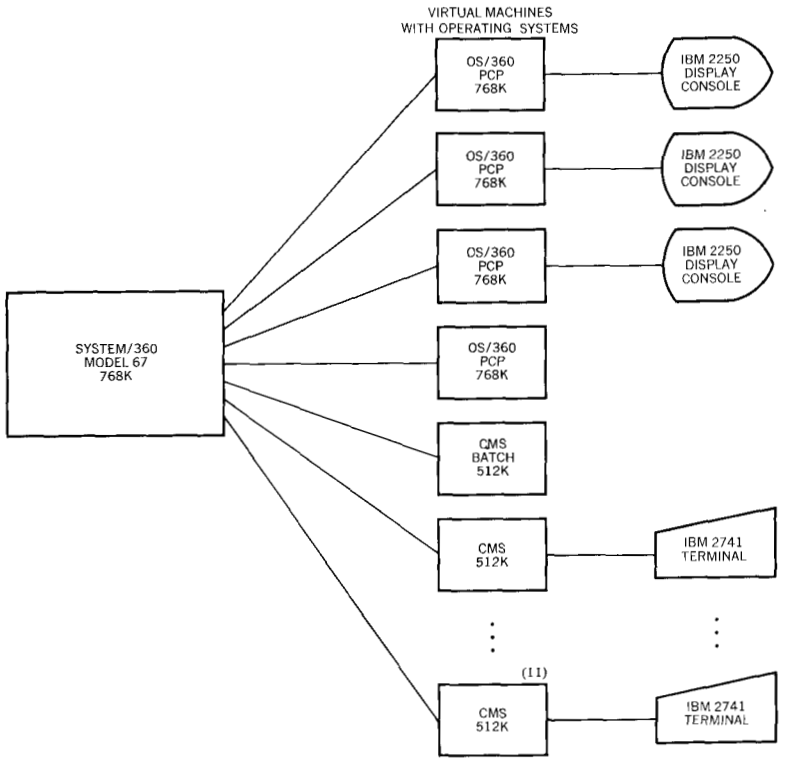
CP-67 is a control program that manages the resources of the Model 67 in a multiuser time-sharing environment.[3,6] CP-67 has implemented the virtual machine concept that provides each user with a replica of a complete System/360 consisting of a virtual CPU, virtual storage, virtual I/O devices, an operator's console, and certain additional features available only under CP-67. Each virtual machine can be loaded with the operating system of his choice. An example of the virtual machines run during prime shift on the installation being discussed is shown in Figure 1.

The virtual machine of each user is defined in a directory that is maintained as a utility function in CP-67. The directory will specify information such as the user's identification, password, priority, virtual storage size, and all virtual I/O devices attached to his virtual machine. Virtual disk drives that are specified for each user may be either a complete disk pack device of 203 cylinders, or just a portion of the pack. Data files may also be shared between virtual machines based on entries in the CP-67 directory. For example, some files may be completely dedicated to a user; others may be shared on a read-only basis, and access to other files may be on a password basis.

operating levels

The CP-67 system operates in three distinct levels. The first level is the CP-67 control program which runs in the supervisor state in a nonrelocatable mode. Some of the major functions performed by CP-67 include handling and processing of all hardware interruptions, scheduling and executing all real I/O operations, performing paging functions, allocating CPU time to the competing

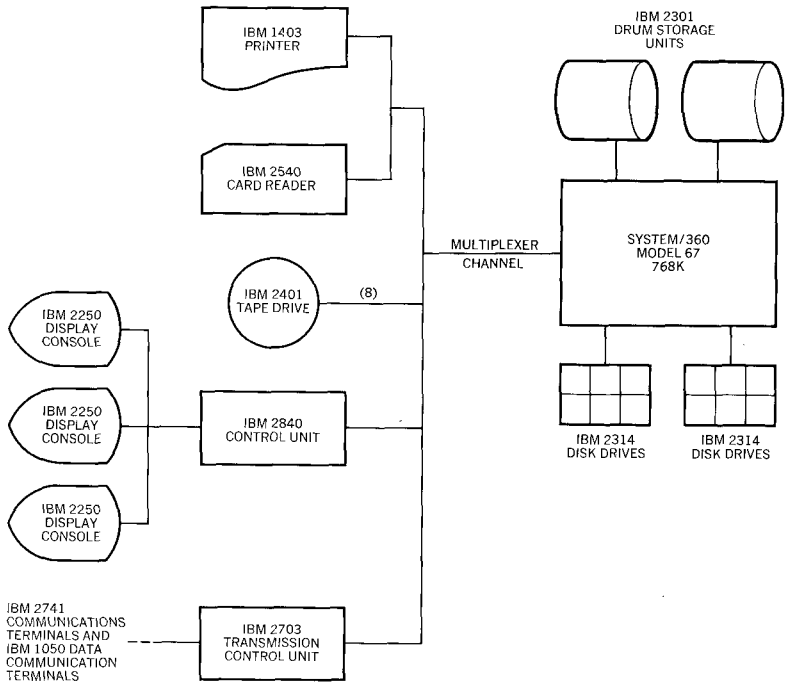
Figure 1 Typical virtual machines



users, and simulating virtual machine privileged instructions. The second level is used for the virtual machine operating systems such as OS/360 and CMS.[5,7] These operating systems are run by CP-67 in the problem state. The third level is used for the application programs that are run by the virtual machine operating system. This level is also run in the problem state.

The allocation of CPU time to the time-sharing users is performed by the dispatch function in CP-67. In Version 3 Level 1 the dispatching algorithm considers two main areas: overall system performance and the demand each individual user places on the system. In considering user demands, CP-67 classifies each user as being either interactive or noninteractive. The dispatcher allocates CPU time to an eligible user in the interactive queue who is performing terminal I/O operations before it allocates time to a user in the noninteractive queue who probably requires more of the system resources and whose terminal response time might not be critical. Some of the factors the CP-67 dispatcher considers in selecting a user for an execution queue include an installation-defined user priority, a system priority that ensures allocation of some CPU time to each user, and a paging activity index that reflects the paging demand a user is expected to put on the system.

Figure 2 System/360 Model 67 real hardware configuration



Overall system performance is considered by limiting the number of users demanding attention at any time in either the interactive or noninteractive queue. This reduces the possibility of poor performance caused by all users competing for real main storage at the same time. The number of users allowed in the interactive queue is fixed depending on the size of the real main storage. The number of users allowed in the noninteractive queue is dynamically adjusted, and it is calculated by comparing the number of available system pages with the sum of the paging activity values of each user active in the queue. Further details of CP-67 operations can be found in References 1, 3, and 6.

system configuration

The hardware configuration that is used in this CP-67 installation is shown in Figure 2. The Model 67 has 768K of main storage. The direct access storage consists of two IBM 2301 drum storage units that are used as paging devices and twelve modules of the IBM 2314 direct access storage facility. Installed on a selector subchannel of the multiplexer channel are eight IBM 2401 magnetic tape units and an IBM 2840 Model 1 display control unit with three attached 2250 Model 3 display consoles. Attached to the IBM 2703 transmission control unit are 20 slow-speed terminals consisting of IBM 2741 communications terminals and IBM 1052 printer-keyboard consoles with attached IBM 1056 card readers.

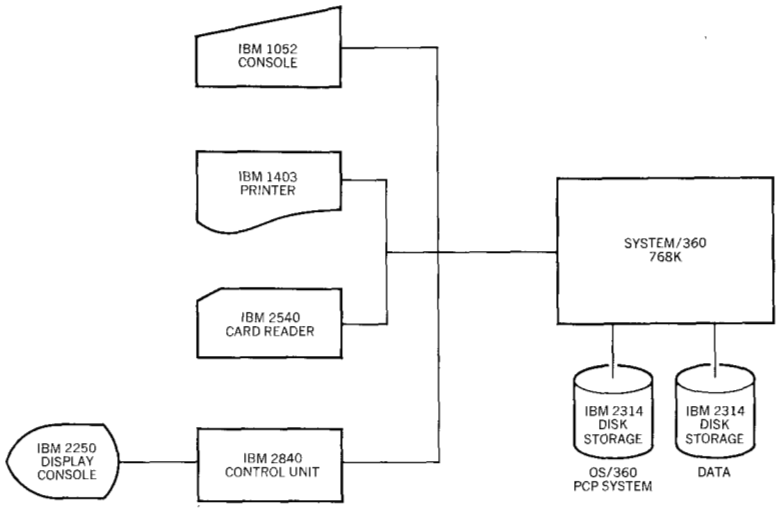
An interesting example of sharing devices between virtual machines under CP-67 arises in the use of the 2250 graphic display system to display graphic information associated with the engineering applications. The display control unit for the three display consoles contains a single buffer of 32K which is used to store the graphics order programs that generate the displays of all three consoles. These order programs are sent to the buffer by the FORTRAN application program running on the CPU. In a system like MVT, the operating system would manage the control unit buffer by allocating a portion of this buffer to each display console. However, under CP-67, three separate virtual primary control program (PCP) machines, each with one display console, share the single hardware buffer of 32K. This is made possible both by the way the PCP systems are generated and by the manner in which CP-67 processes virtual I/O operations. Each PCP nucleus has a unit control block generated for all three consoles, each with a specific 10K of the 32K buffer. However, in the CP-67 directory that defines each user's virtual machine, only one of the three consoles is entered for each graphics user. This will prevent one graphics user from interfering with a console or portion of the buffer of any other graphics user. CP-67 performs the I/O operation to the consoles for all the users on an individual basis and keeps a user identification associated with each I/O operation. It is not aware of the fact that the control unit buffer is being shared or that other consoles are on the system.

virtual machine setup

To install the display consoles and the related graphic subroutine package, an option of OS/360 had to be selected to run the graphic virtual machines.[7] It was first determined that three separate virtual machines, either PCP or MVT, would perform better than one large multiprogrammed MVT system with a region for each user. Both MVT and PCP systems were installed and tested with the PCP system outperforming the MVT system by a significant factor. MVT is a multiprogramming system, and in this case, would be run under CP-67, another system with multiprogramming capability. Overhead caused by facilities that are necessary in an MVT stand-alone system but are only a duplication of existing CP-67 facilities when used in a time-sharing environment resulted in the lower performance. It is more efficient to run a single user virtual system such as PCP and let CP-67 perform the multiprogramming. Another factor is that with MVT all card input is double-spooled, once by CP-67 and once by the MVT reader. In PCP, this problem does not exist because data and job control cards can be read and executed one step at a time, thus eliminating the OS/360 spooling operation.

The three PCP virtual machines each have 768K of main storage and an I/O configuration as shown in Figure 3. The virtual 1052 printer-keyboard that is used for the PCP operator's console is

Figure 3 OS/360 virtual machines



in reality a 2741 communications terminal that is physically placed next to each display console. Several changes to a normal PCP operation were made to improve performance in a virtual machine environment. Since there is practically no memory constraint, the PCP systems have resident access method (RAM), a function called BLDL to create a list of most used supervisor instructions in main storage, and SVC (supervisor call instruction) resident lists that are much more inclusive than would be feasible in a stand-alone system. By making many of these modules resident in virtual storage, a request for their use would be satisfied by either finding the PCP page already in real main storage, or at worst, by paging it in from the high-speed drum. However, if the routines have not been made resident in the PCP system, a request for their use will possibly involve a paging operation plus the additional overhead of reading the module from a slower-speed disk.

Another function of CP-67 that facilitates the operation of virtual machines is the procedure that initially loads the program by name. A CP-67 utility program saves on disk a copy of a user's complete virtual main storage as it exists anytime after the program is initially loaded. When a user subsequently issues the CP-67 command for the procedure to initially load the program, the saved copy of main storage is brought into the user's virtual machine, thereby eliminating the lengthy loading procedure. Changes have also been made in the nucleus initialization program code in OS/360 to automatically vary off-line any disk devices not defined in a user's virtual machine directory thus eliminating unnecessary mount messages. A modification to PCP has also been made to pick up the time and data fields from CP-67 dur-

ing the initial loading by name procedure. With these few changes, plus the OS/360 system generation operations of automatically starting the OS/360 reader and writer, the procedure to initially load an OS/360 system has been reduced to approximately 14 seconds and takes only two commands after a user has logged onto the system. The commands that must be issued are the CP-67 command to load the saved copy of PCP into virtual storage followed by the OS/360 command to initiate the first job.

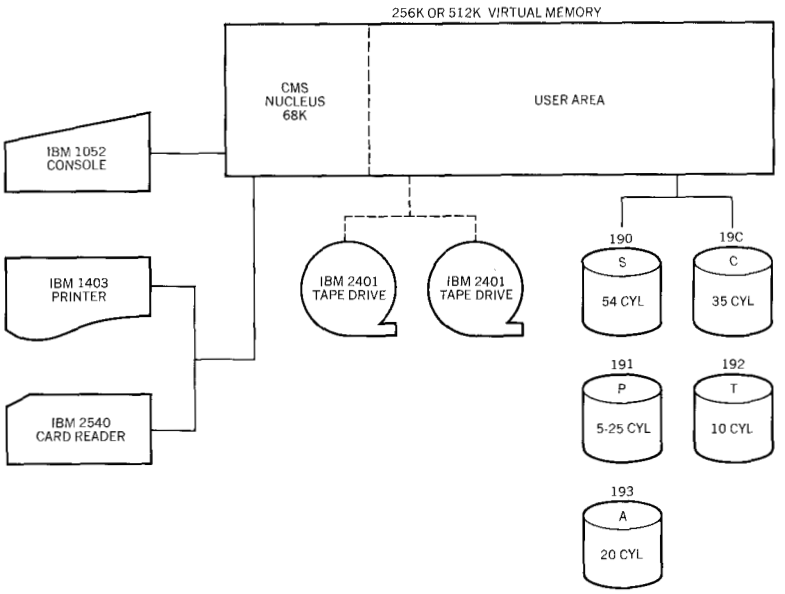
Despite these changes that have been made to ease the operation of the OS/360 virtual machines, an inconvenience still exists. A communications terminal beside each display console acts as the OS/360 operator's console. Thus an engineer at the display console must be somewhat familiar with the OS/360 operator messages and be able to respond to them when required. Since engineers have little inclination to become proficient OS/360 operators, they are occasionally confronted with messages that can only be answered by help from knowledgeable personnel. When this help is not immediately forthcoming, irritating delays may result. However, this problem is not as extensive as it might first appear because the command structure of PCP is not as complex as the other options of OS/360. Normally the engineer knows the responses to standard messages, and it is only the exceptional message that becomes a problem. Although this is a recognized deficiency, it has not hindered the development of graphics applications.

Another example of CP-67 performing a function that is not implemented in MVT is the command to save card information already read into the computer. By using this command, CP-67 will not delete a job or job stream from the user's virtual card reader even after it has been read by the OS/360 reader. This rereading of the card reader is especially helpful to a programmer at a terminal when he is debugging an interactive OS/360 program. Many times when a program terminates or ends abnormally, he would like to execute it again immediately. With this facility, all cards can be saved in the CP-67 spool area and submitted to the virtual machine without having the cards physically reread.

Use of CMS

CMS is a control program[5] designed for an interactive user that runs on any model of the System/360 or in a virtual machine under CP-67. When CMS is run under CP-67, many users have access to their own complete virtual machines, each with its own CMS control program. Since CMS was designed as a single user system, it is not encumbered by facilities such as multiprogramming or spooling that would only be a duplication of CP-67 func-

Figure 4 CMS virtual machines



tions. For this reason a CMS virtual machine will run very efficiently under CP-67. A typical CMS virtual machine configuration in this installation is shown in Figure 4. A user's communications terminal acts as a virtual printer-keyboard console that is attached to the virtual machine. All of a user's console commands and interactive communications will be entered and received through this virtual communication console. The facilities of a card reader/punch and a printer are available to the CMS user for large volume I/O operations through the CP-67 spooling function. The tape drives may be attached by the CP-67 system's operator directly to a user's virtual machine when needed.

Five types of virtual disk drives are attached to each CMS user's virtual machine. Each of these drives appears to the CMS machine as a separate device with the addresses shown in Figure 4. However, this disk space is allocated on "mini-disks," and in reality several of the virtual disk drives might be on the same physical device. In addition, the P, or permanent disks, of many users are on the same physical drive. The S, or systems disk, contains all the CMS systems modules such as the CMS nucleus, macroinstructions, and compilers. This disk is shared on a read-only basis by all CMS users. The C disk can be considered as a read-only extension to the S disk that contains installation-supplied routines available to all users. Included on this disk would be FORTRAN error-handling routines, scientific subroutines, and standard EXEC procedures. (The EXEC procedure is a method whereby a user can create his own command language by combining a series of

commands and logic statements that can be executed by typing a single line.) The P disk is space-designated to each user identification for permanent data or program files. It is also used as temporary storage space during an interactive programming session. The A disk contains files that are shared by users on a selective basis. In this installation, each major department will maintain a common data base and subroutine library that may be accessed only by the identification that is associated with their own department. The T disk is space assigned to eligible users on a temporary basis for the duration of their interactive session. CP-67 maintains a pool of temporary space and allocates it to the users as they log onto the system.

major CMS facilities used

The major facilities of CMS that are used in this installation fall in the three broad areas of file manipulation, compilation, and program execution. The file-handling commands allow a user to manipulate a disk file that is usually either a source program or engineering data. Through the context editor, the user can create a new source program or data file, or make changes to existing files contained in his library. Several of the commands that facilitate the editing of these files allow a user to insert or delete lines, locate occurrences of character strings, change character strings, print selected portions of the file, and store the changed file on disk.

CMS contains several language processors, some that are extensively used and others that are limited in use to the needs of special interest groups. The FORTRAN IV G compiler is the most widely used by the engineers and scientific programmers. CMS has accommodated the OS/360 compiler, and therefore, programs are compatible at both the source and object levels. This compatibility is extremely useful in developing and debugging programs under CMS. Then the object code can be transferred to a virtual OS/360 machine or punched into cards for use on nonvirtual computers. Assembler F and PL/1 F are additional OS/360 processors that are available under CMS, but their use in our application is primarily in systems programming and special application areas. Other processors available for use with CMS, which are used to a limited degree at this installation, include BRUIN (Brown University Interpreter, an interpretive language that performs desk calculator functions), SNOBOL (a string processor), and SCRIPT (a text processor). A debugging facility is also available with CMS to assist in determining program errors. Using this facility, a programmer may stop his program at any point during execution and inspect any storage location, register, or the program status word, make changes in any of these, and then continue execution.

The execution control commands of CMS allow a fully interactive program execution capability. Through the CMS loading com-

mands that may be used by programs during execution, complete overlay structures may be built and executed. A engineering user can also create his own command language by an EXEC procedure. This makes it extremely helpful to nonprogramming engineers because they are not interested in learning all the complexities of the full command structure.

CMS also provides a batching capability for running noninteractive jobs. The CMS batch monitor is run in a virtual machine and accepts input from a card reader or tape drive and writes output on tape, the printer, or the card punch.

Performance considerations

Although it has been discussed by many people at great length, performance of a computing system is still not a well-defined science. This is especially true in the case of time-sharing computers where even a definition of performance can include many different meanings. To the interactive programmer, a response time of five seconds to his terminal request might indicate poor performance. The criterion of measurement for the graphics user is the delay in generating graphic displays. The criterion of the systems group might be the total number of jobs run or system throughput. However, a programmer using the batch facility might be satisfied with a job turnaround time of several hours according to his criterion. Even if a performance criterion is agreed upon, a method to quantify and measure the results is still a major obstacle toward a solution that would improve performance. Another question that would have to be addressed concerns the trade-offs involved in maximizing any criteria; for example, how much system throughput degradation would be tolerated to obtain a measureable improvement in graphic response time?

response time

In this installation, graphic response time is considered the most important measure of performance. However, it is difficult to quantify or measure a user's degree of satisfaction with the system when response time is based on a variety of demands being placed on the system resources by many other users. Despite the difficulties in obtaining precise measurements, a graphics programmer or engineer sitting at a display console can judge whether the system is meeting his own performance criteria. For example, through experience he can give a good estimate of how much time it should take for compiling, or whether the delay in displaying certain graphic frames is excessive. The display console user's comments in this installation indicate that response and program execution time can vary according to the number of users on the system and the demands that they are placing on system resources. Under conditions prevailing in this

installation, performance of the virtual machine under CP-67 is satisfactory and consistent with the users' expectations.

Further improvements in graphics performance under CP-67 is still a goal that is being actively pursued. One of the more basic methods of improving performance is through external scheduling. Programs that are not interactive by design or jobs that put a heavy demand on the CPU are scheduled to be run in the OS/360 or CMS batch virtual machines that can be run during nonpeak periods of the day. It was found that two of the major graphics systems put a high paging demand on the system. They were then scheduled to run at different times during the day and a noticeable performance improvement was obtained. The dispatcher in CP-67 has been modified to give the OS/360 virtual machines priority. When the display consoles are in an interactive mode, they will be given priority over any other interactive user. When the display program becomes compute-bound, it will enter the noninteractive queue and compete for CPU resources with other compute-bound programs. As mentioned previously, CP-67 limits the number of users that can be active in this queue; however, modifications made to the CP-67 dispatcher ensure that the OS/360 virtual machines dedicated to graphics never have to wait to enter the queue or be forced out of the queue because of an accumulation of CPU time.

systems performance

Systems performance is another of the criteria mentioned on which a computer installation should be measured. By investigating systems performance a number of conclusions were reached regarding both the ability of CP-67 to meet the overall computer requirements and the ability of the hardware configuration to support the load placed on it by CP-67. Measurements of systems performance were obtained by two different methods. First, a hardware monitor, the Systems Measurement Instrument (SMI), was attached to the system; second, a software package was installed in CP-67. A summary of the results obtained by SMI are given in Table 1. These results were obtained on Version 2 of CP-67 during the nine-hour prime shifts of a one-week period. Software measurements have verified the measurements given in Table 1.

A few general comments may be made regarding the performance of the hardware configuration. The total CPU utilization is quite high considering the interactive nature and the number of users in this installation. A look at the channel statistics shows that the CPU was seldom in a wait state due to a pending I/O operation on any individual channel. Therefore, little would be gained by the installation of more channels. The effects of reducing the real main storage from 768K to 512K were also investigated. Software measurement showed that the paging rate jumped from approximately 25 pages per second to 75 pages per second. A dras-

Table 1 Performance results

	Average for entire shift	*Average for busiest 2 hours each day*
Total CPU execution time (percent)	74	88
Supervisor State (CP-67)	32	36
Problem State (virtual machines)	42	52
Total CPU wait time (percent)	26	12
CPU wait no channel busy	12	2
CPU wait Channel 1 busy	3	1
CPU wait Channel 2 busy	6	5
CPU wait Channel 3 busy	5	4
Number of users	10–17	13–17
Paging rate (pages per second)		
768K real main storage	25	40
512K real main storage	75	100

tic decrease in overall performance resulted according to users' estimates.

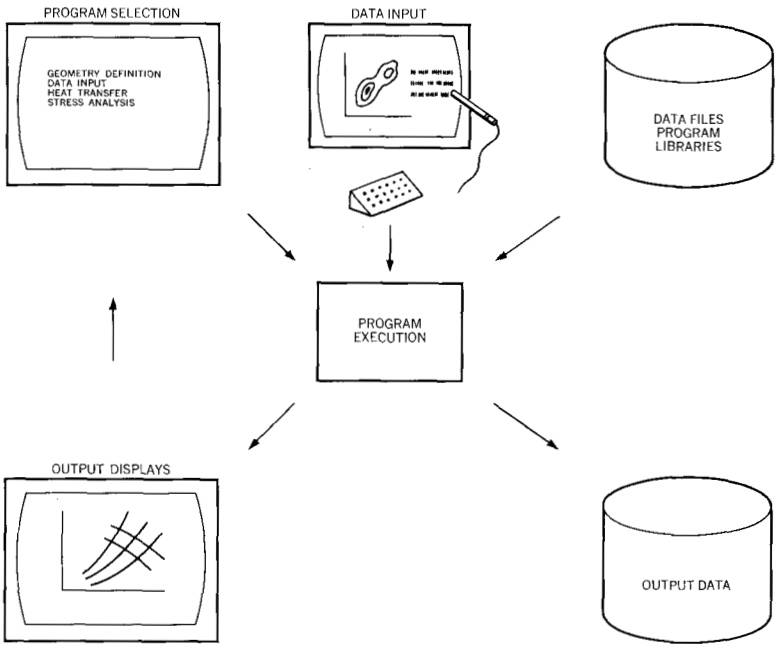
It should not be assumed from the supervisor time shown in Table 1 that an additional overhead factor of 32 percent is incurred by running CP-67. These measurements were made at a specific installation, and they might vary considerably depending on the work load and number of users on the system. Overhead, for the purpose of this brief discussion, may be defined as additional CPU time used that would not occur in the running of these jobs if they were executed in a nontime-sharing batch environment, such as OS/360. Certainly paging time and most of the time needed to translate channel command words would be considered overhead. However, CP-67 dispatching time and time taken to spool card and printer I/O would normally occur in any batch environment. Therefore, the additional overhead caused by the use of CP-67 is less than the 32 percent given in the table. Again, if system performance were a main criterion, a higher job throughput rate could probably be obtained by introducing more batch virtual machines into the system. However, this would be at the expense of terminal response time.

Applications of the system

Many applications have been developed for use in the environment of this installation under CP-67. Several of these applications are now described.

The major interactive application installed on this system is for computer-aided design of jet engine turbine blades using the

Figure 5 Interactive graphics systems flow



graphic display consoles. Historically, this design system had been a batch operation with all programs being run in a sequential manner. The output results of one program would have to be analyzed before a decision regarding the next batch run could be made. With the use of the interactive graphics system, the engineer can now communicate with the computer on a real-time basis. As a result of the previous display, the engineer at the console may decide which analysis to perform next, allowing him to dynamically iterate through difficult computations in one session at a terminal. The system also allows data to be transferred between analyses through a common data structure, eliminating the need for an engineer to prepare data between each computation. A display of input and output data in either curve or tabulated form is also another advantage. A simplified diagram of the design system is shown in Figure 5.

A typical session by an engineer sitting at a display console might go according to the following sequence. An engineer first selects the geometric definition program to define the shape of the desired air foil. He may enter preliminary geometric coordinates through the card reader and then dynamically delete, add, or change points or line that are displayed in picture form on the console. The engineer may then select a program such as a heat transfer analysis to be run, using the geometry just defined and

other data on a disk data set. The results of this program will usually be displayed in graphic form on the console. After studying these graphic data, the engineer might decide to go back to the initial input phase and redefine his geometry, or he may decide to proceed further with the analysis and select another program to be run.

By making many of the engineering programs interactive in nature, the graphics system has substantially reduced the design cycle time in the development of air foils. However, installation of the graphics system was made practical because of the time-sharing and virtual machine facilities of CP-67.

program development

Interactive program development under CMS is another major use of the Model 67. At present, this facility is used almost exclusively in the scientific environment with most of the programs being written in the FORTRAN language. In some instances, the development time of batch scientific programs has been substantially reduced through the use of CMS in comparison to the previously used batch method of development. This reduction can be attributed to the advantages of time-sharing systems in which a programmer at a single terminal session can perform many iterations in the compiling, debugging, and testing phases of program development. Once the batch programs have been developed under CMS, they can be put into production on a batch computer.

CMS is also used to develop the OS/360 graphics programs. Since the virtual OS/360 machines and the display consoles are primarily used as a production tool, very little time is available for program testing. To conserve virtual OS/360 machine time, all graphics programs are compiled and initially debugged under CMS. When the programs have been successfully compiled, the object code can then be tested on the virtual OS/360 machine. Using the EXEC procedure facility of CMS, the object deck can be combined with the proper OS/360 job control language cards into a single file. This file can be transferred to the CP-67 spooling disk where it can be read by the OS/360 virtual machine when test time is available.

systems programming

System programmers have made excellent use of the virtual machine facilities. Modifications and additions to PCP and CMS are made by the systems programmer at his terminal. These changes can be completely tested and debugged in a virtual machine that is isolated from any of the production systems. Once the test results are satisfactory, the modifications can be incorporated into the other production systems. New releases of operating systems such as OS/360 can be generated in a virtual machine during the prime shift. If it is desired to debug or test a new release of CP-67, the new release can be run in a virtual machine under the existing production version of CP-67.

program interchange

Several engineering departments have the requirement to interchange programs and data with other manufacturers and vendors in the industry. In some cases, these programs were written at other installations where the hardware configuration enabled these programs to be developed using a large amount of main storage. On smaller systems, these large programs were very inconvenient to schedule or they required modifications to fit within existing main storage limitations. With a virtual storage system, size is no longer a factor that has to be considered when running these jobs.

interactive program execution

An interactive program execution facility at low-speed terminals has been provided for the use of the engineers. CMS EXEC procedures have been written to assist the user in executing programs during a terminal session. The engineer is led through a series of questions about a program he wants executed, the input media he would use, and the device that would be used for his output. An engineer may choose to submit input data to his program from his terminal keyboard or its attached card reader, the card reader at the computer installation, or from a stored data set on disk. The output may be received at the terminal, or if the volume is large, it may be directed to the CP-67 spool file, then printed on the printer. A typical user of this facility is a design engineer who may execute approximately 30 programs that are stored on the CMS A disk library which is maintained by his group. These programs are usually of short duration and may include moment of inertia or gear calculations, stress or vibration programs, or a weight analysis. This application is designed primarily to give the engineer a quick turnaround capability for small batch jobs rather than providing an interactive capability during program execution.

plot facility

A spooling capability has been developed for use under CMS to facilitate the operations involved in creating tapes that are to be used with off-line plotters. In the past, a CMS user who wished to create a plot tape would have to wait until a seven-track tape became available. This might have taken a considerable amount of time. A systems operator would then dedicate a free tape drive to the user, usually for the duration of his terminal session. The spooling capability was implemented by using many of the facilities of virtual machine computing in the following manner. The original plot subroutines contained in the user's program have been modified to punch card images into a virtual card punch instead of creating plot records directly on tape. When the plot deck has been completely punched into the virtual card punch, the card image file is transferred to the virtual card reader of a virtual CMS machine dedicated completely to the plotting application. The virtual machine will read the card images as if they came from the physical card reader and then write the plot data onto tape. This virtual plot machine runs in a disconnect mode

which means that the physical terminal that was used to initially load the system may be disconnected from the operation of the virtual machine. This disconnect feature thus eliminates the need for a physical terminal being tied up whenever the virtual plot machine is active. This virtual machine is usually started by the systems operator at the beginning of the day and remains on as long as CP-67 is running. When there is no plot work being performed, the CMS machine places no demands on any of the computer resources and is probably completely paged out of main storage within minutes after it has become inactive. It will remain in this dormant state until it has been notified that there is a file to be processed in its card reader. This system has improved the utilization of tape drives and also increased operator efficiency since the plot files of many users may now be spooled onto a single physical tape.

remote batch job entry

A CMS batch machine is also available to run programs that are not interactive in nature or do not require a quick turnaround. These jobs may be submitted from the local card reader or by a user at a terminal who transfers an input file from his virtual machine to the CMS batch virtual machine. This batch machine runs in a disconnect mode, not requiring a terminal to be dedicated to its use.

OS/360 usage

An OS/360 batch system is available to run special processors that are supported only in an OS/360 environment. These processors include the General Purpose System Simulator (GPSS) and the FORMAC interpreter, which is an extension to PL/1 and provides for the symbolic manipulation of mathematical expressions. A user at a CMS terminal may interactively create or modify the GPSS or FORMAC parameters and then use an EXEC procedure to combine the processor input and the OS/360 job control language into a single file. This file can then be transferred to the OS/360 batch machine for execution.

Although the development of many applications has been made possible by the installation of CP-67 CMS, a major limitation is processor speed. For example, in a complete air foil analysis, several programs are still not included in the interactive graphics system because they would require several hours of CPU time even if no other user was on the system. A more powerful CPU would be needed to make these programs interactive in a practical sense.

Summary

The Model 67 with CP-67 was installed at Pratt & Whitney Aircraft to meet specific requirements of their engineering department that could not be met by a traditional batch computer sys-

tem. An extensive interactive graphics system, using graphic display consoles, has been developed to assist the engineer and reduce the development cycle time of an engineering design. The interactive programming and job execution facilities provided by CMS have given the engineers and programmers simultaneous computer capabilities at local terminals through the use of virtual machines. The concepts of time-sharing and of virtual machines have certainly proved beneficial for the present applications and will definitely be a valuable tool in implementing many engineering applications in the future.

ACKNOWLEDGMENTS

The author would like to thank Thomas Barry of the System Programming staff at Pratt & Whitney Aircraft for his major contribution in the areas of the CP-67 installation and systems performance. Discussions with Jeanne Bourque, also of Pratt & Whitney Aircraft, were very helpful in areas concerning the engineering users and applications.

CITED REFERENCES

1. R. P. Parmelee, T. I. Peterson, C. C. Tillman, and D. J. Hatfield, "Virtual storage and virtual machine concepts," in this issue.
2. P. J. Denning, "Virtual memory," *Computing Surveys* **2,** No. 3, 153–189 (September 1970).
3. R. A. Meyer and L. H. Seawright, "A virtual machine time-sharing system," *IBM Systems Journal* **9,** No. 3, 199–218 (1970).
4. A. D. Rully, "Interactive graphics in data processing: A subroutine package for FORTRAN," *IBM Systems Journal* **7,** Nos. 3 & 4, 248–256 (1968).
5. *CP-67/CMS Version 3 System Description Manual*, GH20-0802-1, International Business Machines Corporation, Data Processing Division, White Plains, New York (1970).
6. *CP-67 Program Logic Manual*, GY20-0590-0, International Business Machines Corporation, Data Processing Division, White Plains, New York (1970).
7. *CP-67: Operating Systems in a Virtual Machine*, GH20-1029, International Business Machines Corporation, Data Processing Division, White Plains, New York (1971).