Time-sharing has resulted in the development of methods to increase the utilization of computers. In this paper, one such method employing the concept of the virtual machine is discussed.

Described are the design objectives of CP-67/CMS, *a multi-access system that manages the resources of a computer set up for time-sharing such that each user appears to have a complete, dedicated computer at his disposal. Also discussed are the system operation and some of its applications.*

A virtual machine time-sharing system

by R. A. Meyer and L. H. Seawright

Time-sharing systems are now an important factor in the computer industry. Because they allow a multiplicity of users to have access to a computer by means of a terminal, they encourage increasing numbers of people to utilize computers. Thus, various ways of increasing the capability of a computer in this area are being sought. One such system is the Control Program-67/Cambridge Monitor System (CP-67/CMS), a multi-access system that manages the resources of a computer set up for time-sharing such that each (remote) user appears to have a complete, dedicated computer at his disposal. This concept is known as a virtual machine and allows each user to select the operating system he wishes to run because concurrent operation of several operating systems is possible.

In this paper, the history and design objectives of the system are discussed as well as its present capabilities and some of the applications for its use.

Development history

Development of CP-67/CMS was based on a research effort with the following objectives: (1) research in various time-sharing techniques and methods, (2) the examination of hardware require-

ments for time-sharing, (3) the development of a time-sharing system for in-house use, and (4) the development of a method of observing the dynamic interaction between operating systems and their hardware environments.

The fourth objective led to the idea of simulating other computer systems, namely configurations of the IBM System/360, to measure various combinations of hardware and software. With simultaneous simulation of several terminal-oriented computers, the objective of providing a time-sharing capability could also be achieved.

Implementation of a virtual machine system was then begun, and a specially modified System/360 Model 40 was designed and built. This modification provided a memory address translation mechanism in the form of a 64-register associative memory, thus allowing the memory of 256K bytes to be divided into 64 pages of 4096 bytes each.[1,2]

The software was composed of two independent components. The first component, called the Virtual Machine Control Program, provided the functions of time-sharing and resource allocation. In this arrangement, the user interface and data and task management are the responsibility of the operating system of the virtual machine. The second component, the Cambridge Monitor System (CMS), was designed as a single-user, conversational monitor system that provided the functions necessary to operate a computer in a convenient manner from a console typewriter. It was to be written such that it could also run on a standard System/360 without the virtual machine control program.

This functional division approach has several advantages. With the responsibilities divided sharply between the control program and CMS, each part is less complex and easier to implement. In addition, development was able to proceed in parallel, since CMS was to be capable of running by itself. Indeed, CMS was running by itself on the Model 40 long before it ever "saw" a virtual machine environment.

The feasibility of such a system was shown when the control program for the Model 40 supported 14 virtual machines using CMS and fulfilled its goal of providing a productive in-house time-sharing system. Shortly afterward, development of a virtual machine system for the System/360 Model 67 was begun, called CP-67. Since CP-67 and the control program for the Model 40 provided the same basic interface, the same CMS was used on both systems. Thus, in the present system, the control program (CP), which is CP-67, creates the time-sharing part of the system that allows simultaneous operation of several virtual machines, and CMS is a conversational operating system for terminal operations.

The virtual machine environment

The expression, virtual machine, is now generally accepted as a software replica of a complete computer system such as System/360. It consists of a data structure describing the memory size and the input/output configuration of the simulated system. There is a master control block in which are stored the registers, program status words (PSW's), status information, and appropriate pointers to storage blocks describing the input/output devices and associated segment and page tables concerned with management of virtual memory. This "machine" appears to its user as a standard System/360 and can execute any standard instruction except DIAGNOSE.

operating system support

The virtual machine can support many operating systems (including OS/360, DOS, RAX, DOS/APL) as long as they do not include any rigid timing dependencies or dynamically modified channel programs. (Several operating systems can be used simultaneously if we are considering the simultaneous operation of several virtual machines.) The interval timer presented to the virtual machine does not reflect true time-of-day, since the CP (and the time absorbed by it) must be transparent to the virtual machine. Thus program code which depends on precise operation times may not work. Real time interruption conditions (such as the System/360 Program Control Interruption) will not, in general, be reflected to the virtual machine quickly enough to be used for the intended purpose due to the competition for the computer by other virtual machines. Dynamically modified channel programs are those that are changed between the time the start input/output (SIO) instruction is issued and the end of the input/output operation, i.e., changed by the channel program or the central processing unit (CPU). A channel program can modify itself by having a read operation use another part of the channel program as an input buffer. However, certain simple types of self-modifying channel command sequences can be translated, such as those generated by the indexed sequential access method of Operating System/360 (OS/360).

Since the virtual machines are simulated, their configurations may differ from each other and from the real machine in terms of virtual memory size and number and type of input/output devices.

Like real machines, virtual machines will operate most efficiently under operating systems. CMS is designed to allow control of a System/360 through a simple command language entered at the console. Since each user has his own virtual machine with his own copy of an operating system residing in it, nothing he does should affect any other user. If his operating system terminates abnormally or destroys some part of itself, it will affect only that virtual machine and no others. The affected user can simply re-initiate his virtual system without disturbing other users. In particular, users cannot get outside their virtual machine. Due to the uniqueness of their

virtual storage and their virtual input/output devices, not only are the users protected from each other, but the CP-67 system itself is protected from user error.

To run as a replica of a System/360, the virtual machine must have counterparts to all the components of the real machine. Therefore, a virtual System/360 is comprised of an operator's console represented by the remote terminal, a virtual memory, a virtual CPU, and virtual input/output devices.

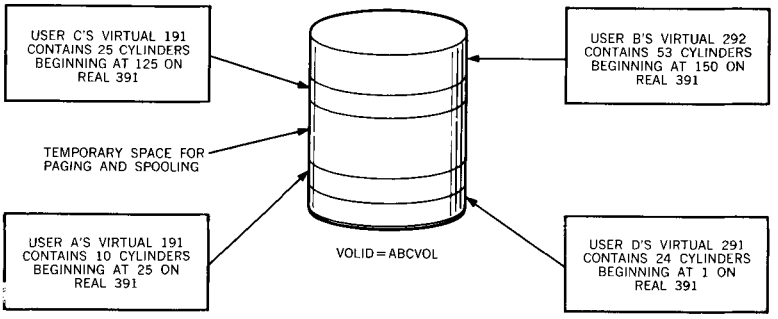
console functions

The remote terminal represents both the on-line terminal and the entire console display of a System/360. By means of appropriate terminal commands, the switches and lights of the console are simulated, thus providing control of a virtual machine. These console commands constitute the basic command language of CP-67 and are called console functions. There are three types of console functions. The first type simulates the control panel and includes such commands as DISPLAY—to examine the contents of virtual memory and registers, EXTERNAL—to generate an external interruption, BEGIN—to start the virtual machine, and IPL—to load the virtual machine. A second type is called extended console functions because they go beyond the simulation of the console and include commands such as DETACH—to remove input/output devices from a virtual machine, MSG—to communicate with other users, QUERY—to interrogate users and status of files, SET—to control the printing of messages at the terminal, and LINK—to add a subset of a direct access device to a virtual machine. A description of the first and second types of console functions are presented later in the paper. The third type provides the real system operator with special control features such as ATTACH a real device to a given user, SHUTDOWN the system, ENABLE telecommunication lines, and TERM (terminate) output on the unit-record devices.

CP-67 uses the dynamic address translation feature[3] of the IBM System/360 Model 67 to provide up to 16 million bytes of virtual memory for each user (the full range of 24 bit addressing); consequently, the user's effective memory space can be larger than the real memory of the Model 67. Since inefficient use of very large virtual memory can put an extremely heavy burden on the paging facilities of the CP and seriously degrade system performance, virtual memory is normally assigned in sizes ranging from 256K to 1024K bytes. It can be defined in 8K bytes and any multiple thereof. The size of the virtual memory is defined for each user in a table called the system user directory; thus virtual memory can differ for each virtual machine. Virtual memory is normally completely private to each machine, although certain areas of virtual memory, such as in the nucleus of CMS, can be shared by different users.

The virtual CPU, of course, is just the real CPU of the Model 67 shared by the different virtual Systems/360 using time-slicing tech-

Figure 1 Shared disk for permanent user files and CP-67 files

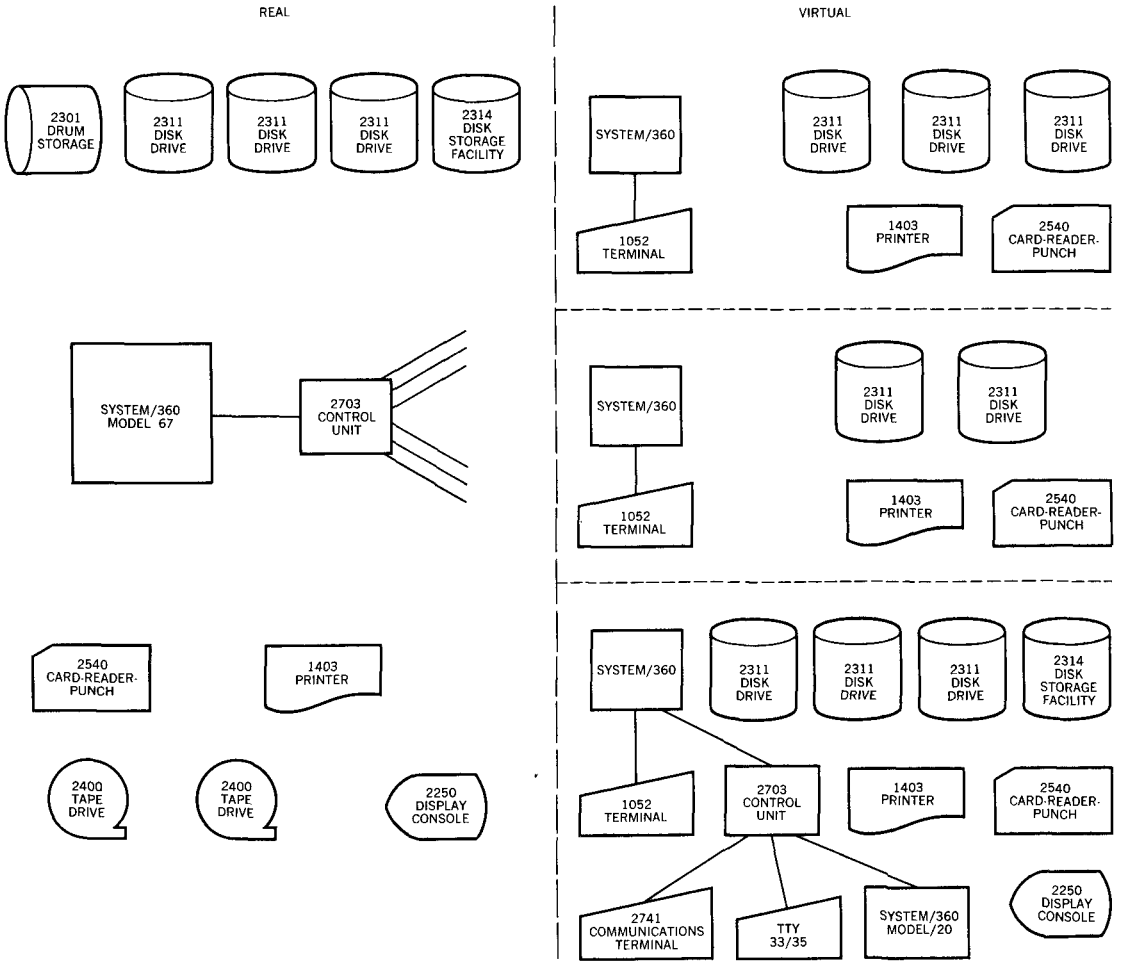


niques. The present system provides a virtual CPU representing a standard System/360. The concept could be extended to include the relocation and paging hardware of a Model 67, but it is not currently supported by the system.

Every computer, of course, must have input/output devices. A data structure containing control blocks for each virtual channel, control unit, and device is maintained within the CP for each virtual machine. These input/output devices are controlled by the virtual system, not CP-67; therefore, the software which supports the virtual input/output device must be present in the virtual machine's operating system. An input/output device may exist on the virtual machine with a different address than that existing on the real Model 67. Such an input/output device can also be defined as having a size not available in the real world, such as seven cylinders of a disk pack or a subset of the lines on a transmission control unit. There may be a virtual machine that contains only four lines of a transmission control unit, whereas the real machine has a different transmission control unit with 88 lines. Similarly, a disk pack of a large disk storage facility such as the IBM 2314 may be suballocated into "mini-disks"; for example, several virtual 10-cylinder IBM 2314 disk packs. In this way, many mini-disks can reside on the same physical disk pack, allowing many more virtual machines (each requiring one or more private disks) to be run at a given time than there are real disk drives available. Figure 1 portrays the arrangement of mini-disks on a disk pack, along with temporary space provided for use by the CP. In the present implementation, it is generally true, however, that the type of virtual device must exist on the physical machine before it can exist on a virtual machine.

Figure 2 illustrates the virtual machine concept showing varying configurations. Figure 3 shows multiple virtual machines running different operating systems.

Figure 2 Virtual machine concept using IBM System/360

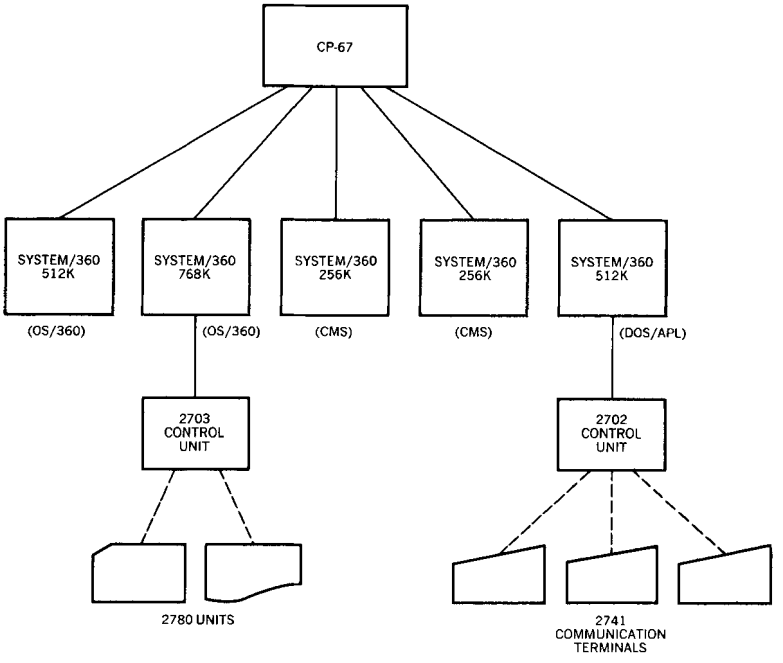


Hardware requirements

Implementation of a practical virtual machine system depends on the host computer having certain hardware features to ensure proper control by the supervisor and to avoid excessive overhead in the translation or simulation process. These hardware features are as follows:

- The virtual machine and host machine should have as nearly identical an instruction set as possible so that nonprivileged instructions can be executed directly by the virtual machine. Nonidentical, nonprivileged instructions could be simulated by the control program provided an interruption mechanism exists to activate it at the proper time.

Figure 3 Virtual System/360 computers created by CP-67



- The host computer must have ways of distinguishing between privileged and nonprivileged instructions. The supervisor must be made aware of any attempt by a virtual machine to execute a privileged instruction, or change its mode of operation. This requirement is satisfied on a System/360 by the "supervisor" and "problem" states of operation. The virtual machine created by CP-67 is always run in the problem state, and any attempt to execute a privileged instruction causes a trap to the supervisor state, or CP-67. The trap must not cause an unrecoverable error, and the CP must then simulate the effect of the instruction.
- The CP must be protected from the user programs (i.e., the memory assigned to the CP must be protected in both read and write mode).
- The virtual machines must be protected or isolated from each other. This means that the memory or input/output devices assigned to one virtual machine must be inaccessible from any other virtual machine, unless proper safeguards are provided. Isolation and protection of memory areas can be accomplished by some form of memory protection, as in a standard System/360 (up to 15 users), or an address translation scheme as in the Model 67. However, the normal memory protection feature does not adequately solve the problem of nonrelocatable programs and absolute addresses. All System/360 operating systems contain absolute addresses, since the architecture of the Sys-

tem/360 assigns certain memory locations in lower main storage to particular functions such as the timer, channel status word, channel address word, and new and old program status words.[4] Relocatability of absolute memory addresses is provided by the data address translation feature of the Model 67. Isolation and protection of input/output devices and data sets residing on input/output devices is accomplished by trapping input/output instructions (which are privileged instructions in the System/360) and simulating them within the CP. Thus all actual input/output instructions are executed by the CP.

The Model 67 meets the above hardware requirements without any modification; therefore CP-67, with its virtual machine capabilities, was implemented on the Model 67.

System usage

Before a user is authorized to use CP-67, he must be assigned a USERID, which is a name that identifies him to the system, and a password, which is checked when he "logs in" to initiate use of the system. Associated with each user identification is information concerning accounting, privilege class, options desired, and a table describing the virtual machine assigned to that user. Whenever he logs in, CP-67 sets up this virtual machine for him. Although all the virtual machines may be different, most are set up with the configuration expected by CMS, which is the most commonly used operating system. They include at least 256K bytes of main storage, two disk drives, an operator's console (the terminal), a card-reader-punch unit, and a printer.

paging

Because there is not room in real main storage for all users' virtual main storage, a technique called *paging* is used by the system. Virtual main storage is divided into 4096-byte blocks of storage called *pages*. All but currently active pages (i.e., those in current use) are kept by the system on direct access secondary storage; because active and inactive pages change status, they are "swapped" between real main storage and secondary storage on a demand basis. While the swapping operation is being performed for one virtual machine, the CP runs one or more virtual machine(s). The paging operation, and resultant allocation of real main storage to a given user's pages, is transparent to the user. The dynamic address translation feature of the Model 67 translates, at execution time, the user's (or user's program's) addresses into the current real addresses of the relocated pages.

The implementation of the virtual machine concept employed by this system requires that only the CP may operate in the supervisor state on the real machine. All programs other than CP—i.e., all programs being executed on virtual machines—operate in the prob-

lem state on the real machine. However, the virtual machine operating system has to use privileged instructions; therefore CP supports a virtual supervisor state in the virtual machine by intercepting and simulating the privileged instructions. All user interruptions, including those caused by attempted privileged operations, are handled by CP, which then reflects to the virtual machine the results to be expected from a real machine. The user may expect his programs to be executed on his virtual machine in a manner identical to their execution on a real System/360, provided he has not violated the basic restrictions concerning timing dependencies and channel programs.

input/output operations

Because the input/output instructions are privileged, all virtual machine input/output operations are intercepted and handled by CP, which must then translate them into real machine input/output operations. This requires two translations, accomplished as follows: CP intercepts all user input/output when a start input/output (SIO) instruction is issued. It translates virtual device addresses into real device addresses, translates virtual main storage addresses into real main storage addresses, ensures that all necessary pages are in real main storage, builds a channel command word (CCW) string for the user from his original CCW sequence, and starts the input/output operation when the channel is free. The virtual machine is not given control from the time it issues an SIO instruction until the CP issues the real SIO instruction and delivers the condition code to the virtual machine. In the meantime, one or more virtual machine(s) may be operating. When CP receives an interruption indicating completion of input/output activity, it saves information to that effect in the user's virtual machine status table; when control is returned to the virtual machine, the proper input/output interruption is simulated.

Virtual machine unit record input/output activity is normally collected as a disk file (i.e., spooled) by CP. Thus, any card deck to be "read" by a virtual machine would in the normal case have been read by CP prior to the user's call for it from his virtual machine, or transferred to that user from another user's files via the XFER console function in CP. The card deck in question must be preceded by a card containing the user identification, so that CP can know by whom the card-image file is to be read. Later, when the virtual machine has "read" the card deck, a card reader end-of-file interruption is simulated. Card-punch and printer output, similarly spooled, is not queued for physical output until CP is "notified" of an end-of-file interruption. Further output for a closed device is assumed to start a new file. So that the system operator can separate physical output, CP-67 precedes all printed and punched output files with a record containing the user identification.

A virtual machine can have dedicated unit-record devices instead of sharing the system's unit-record devices. With these dedicated

devices, spooling is not performed by CP-67; the normal input/output requests are issued by the virtual machine, intercepted and translated by CP-67, and then the real input/output sequences are issued. For example, if OS/360 having multiprogramming with a variable number of tasks (MVT) is run under CP-67, double spooling will occur, once for OS/360 and once for CP, if the unit record devices are not dedicated to the OS/360 MVT virtual machine.

Virtual machine input/output to the on-line terminal console is simulated, because CP must translate that input/output into sequences for the remote terminal from which the user has logged in.

error conditions

If input/output errors occur as a result of input/output activity initiated by the virtual machine, the error conditions are reflected back to the virtual machine. The error recovery procedures provided by the virtual machine's operating system will then handle the error conditions as they would on a real System/360. Errors occurring during CP-initiated input/output operations (such as spooling and paging) are handled by CP itself.

The CP console functions allow the user to control his virtual machine from the terminal much as an operator controls a real machine. To perform an initial program load, for instance, the user types IPL and a device address or the name of a "named" operating system, such as CMS. The user can stop his virtual machine at any time by depressing the ATTN key on his console and request display of any portion of his storage. To start his virtual machine running again, the user can issue the BEGIN console function of CP. Each of the CP-67 console functions that can be issued by any user from a terminal is described in Table 1. The privileged operator commands are not included in this list.

As well as running single-user systems, CP-67 can run multi-access systems. The multi-access systems being run under CP can have high-speed and/or low-speed communication lines. In the case of low-speed lines, once the multi-access system has been loaded into a virtual machine and the communication lines prepared, users at remote terminals can become connected to this virtual machine by issuing the CP console function DIAL. When the remote terminal is connected to that system, it is treated by CP-67 as an attached transmission line. The existence of a terminal is then known only by the virtual machine, and thus CP-67 console functions are not available. Only when that multi-access system disables the communication line will CP regain normal control of the terminal.

Some of the multi-access systems that have run under CP-67 are DOS/APL, RAX, and CPS. Additional operating systems that have run under CP-67, in addition to CMS and the CMS Batch Monitor, are OS/360 (with its options such as MVT, PCP), DOS, and many diagnostic programs. Note that although the basic operating sys-

Table 1 User console functions

Function	*Definition*
BEGIN	begins execution at the specified address or, if no address is given, at the location at which execution was interrupted
CLOSE	releases the disk areas (spooling areas) containing input from the card reader or output to the printer or card punch
DETACH	removes the specified device from the user's virtual machine configuration
DIAL	is used in place of LOGIN to connect a user's terminal with a virtual telecommunications operating system or a virtual time-sharing system
DISCONN	allows a user to disable the terminal and leave his virtual machine running
DISPLAY	types at the terminal the contents of the specified register(s), main storage location(s), or program status word
DUMP	prints the contents of the specified register(s), main storage location(s), or program status word on the off-line printer
EXTERNAL	simulates an external interruption to the virtual machine
IPL	simulates the initial program load sequence on the specified unit
LINK	attaches (after log in) virtual disks in accordance with information contained in the system user directory
LOGOUT	releases the user's virtual machine, including input/output devices, and closes any spooling areas which have not been released
MSG	types the specified message at the terminal of the person whose user identification is specified
PURGE	erases spooled input or output files by device
QUERY	types out status information concerning active users, spooled files, and time used
READY	simulates a device end for the specified unit
RESET	simulates the system reset key on the System/360 console
SET	controls the saving of virtual card-reader files and the typing of messages at the terminal
SLEEP	invokes a special console function mode to facilitate reception of messages; the virtual machine is in a dormant state
SPOOL	directs spooled output and controls the reading of spooled input
STORE	replaces the contents of the specified register(s), main storage location(s), or program status word with the specified information
XFER	establishes a logical connection between a user's spooled output device and another user's spooled input device

tems have been run successfully, particular applications may violate the timing and input/output restrictions described earlier and cannot be used.

CP-67 is currently totally main-storage resident and is approximately 80K bytes in size. Additional main storage is assigned as CP free storage areas depending on real memory size. The remaining main storage is made available for paging the users' virtual memory.

The minimum configuration required to run CP-67/CMS is a System/360, Model 67, simplex or half-duplex, 256K memory, three IBM 2311 disk drives or two IBM 2314 storage modules, a nine-track, 1800 bpi tape drive, a card-reader-punch, and a printer. Typical configurations also have storage drums plus additional memory and disk storage module capacity.

system performance

Running an operating system under CP-67 will, of course, add additional overhead, thus increasing the elapsed time in comparison to stand-alone elapsed time. The primary source of overhead for a single virtual machine is in the input/output processing function. As far as degradation is concerned, an input/output bound task will suffer the most.

The number of virtual machines supported by CP-67 must be considered in the context of the type of operating systems being run from each virtual machine, as different operating systems put different loads on CP. The simpler the structure of the virtual operating system, the lighter the load imposed on the CP. OS/360 job streams, for example, may suffer a degradation in the range of 30 to 100 percent. The way to maximize total throughput under the CP is to exploit its capability of efficiently multiprogramming a virtual machine. In cases where job streams can be run from two virtual machines, running multiple OS/360 systems can provide under the best conditions more OS/360 throughput under CP-67 than a single OS/360 system.

As far as CMS under CP-67 is concerned, CMS is designed to communicate with a single user at the console; it knows nothing about multiprogramming or multiple users. Therefore, because of its simple design, CMS does not place a heavy load on CP-67. There are installations currently running 30 to 40 CMS virtual machines on a 512K Model 67.

The monitor configuration

The Cambridge Monitor System is a single-user, conversational operating system, capable of running on a real machine as well as on a virtual machine. It interprets a simple command language

Table 2 Machine configuration for CMS

Device type	*Device name*
1052	console
2311 or 2314	system disk (read-only)
2311 or 2314	permanent disk (user files)
*2311 or 2314	temporary disk (work space)
1403	line printer
2540	card-reader-punch
*2400	tape drives

at least 256K bytes of main storage.

*optional devices

typed in at the operator's console (when used with CP, the user's remote terminal).

Whether running on a real or a virtual machine, CMS expects the machine configuration to be as shown in Table 2. Under CP, of course, these devices are simulated and mapped to different devices. For instance, CMS expects an IBM 1052 printer-keyboard operator's console, but most remote terminals are different; CP handles all channel program modifications necessary for this simulation.

CMS takes advantage of the CP environment by executing all programs in the "pseudo-supervisor" state; thus the user is free to use all System/360 instructions. This also allows the user to add his own programs for input/output devices not supported by the standard system.

file management under CMS

Each CMS user is assigned two disks (a third disk is optional), one of which is shared with other CMS users. These disks, under CP, are seldom complete disk packs. At the time a user is authorized to use CP-67/CMS, the size of each disk area is set by the system administrator, according to the needs of the user and the total amount of disk space available.

The shared disk contains the CMS nucleus, which is loaded into the virtual machine by the IPL console function. Also on this disk, referred to as the "system disk" are disk-resident programs and system macro and program libraries. The CP prevents any user from writing on this disk as it is read-only. Any attempt to modify the system files results in an error message.

The two other disks are known as the "permanent" and "temporary" disks. The user does not normally share these disks with any other user, as they are accessible only to him after he has logged in with the correct user identification and password. The permanent disk is used for files that are to be saved from one terminal session to the next. The temporary disk, which is optional, provides space for work files which need not be retained between sessions. This disk is erased whenever the user logs out.

All CMS disk files are written in 829-byte physical records. The system input/output routines place logical records into this format, and the records are allocated only as needed. CMS maintains chains of disk addresses to keep track of the files. These chains are linked to the user file directory, which CMS maintains on each disk to record information on the file formats, sizes, and locations. The user file directory is brought into storage when the user logs in and is updated whenever files are modified. Periodically, and if the files have been modified, the updated directory is written onto disk, so that the permanent copy is as current as possible. This ensures an accurate directory if it is necessary to reload CMS during a terminal session.

The directory handles files up to 25.24 million bytes in length, which is 203 cylinders of an IBM 2314 disk pack and is beyond the capacity of a smaller storage unit such as an entire IBM 2311 disk pack. In practice, the user's disk will not normally require files of that length, since the typical user needs less than 25 cylinders. Whenever CMS detects that only a few tracks are left on the user's disk(s), a warning message is typed, and the files currently open are closed. A program or command in execution is halted, so the user may create more free space on the disk by erasing some files, or by copying them from disk to other media.

Although most of the CMS commands operate on disk-resident files, the user also has access to the card-reader-punch, printer, and tape drives. The commands, in general, create sequential files of fixed-length logical records; however, the programmer using the CMS disk and tape input/output support routines is able to use any logical record format with either fixed-length or variable-length records. Fixed-length record files may also be read or written by direct access.

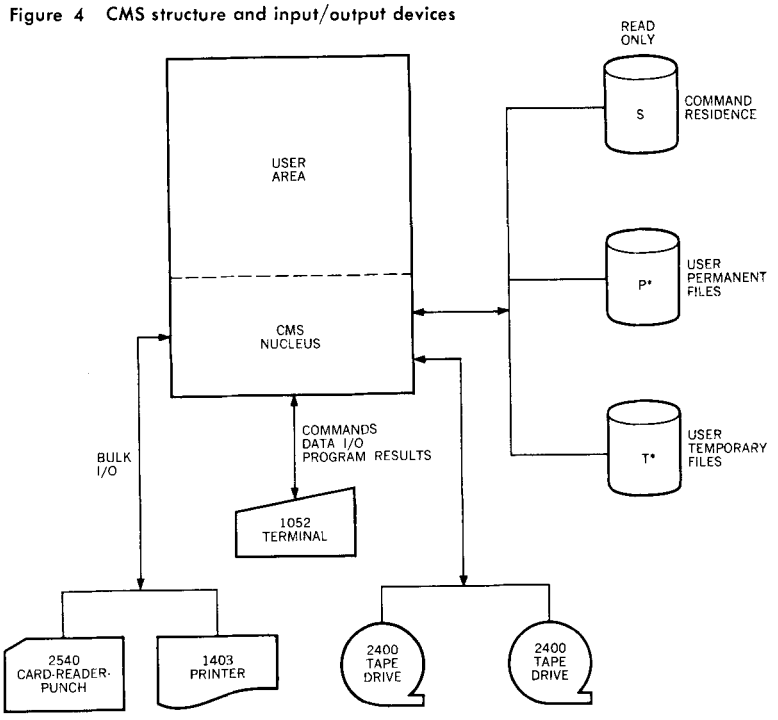
Files are automatically "opened" for reading or writing when the first read or write command is issued. CMS routines automatically close files after every command. Files must be closed between writing and reading within the same program.

Figure 4 illustrates the CMS input/output devices and their use.

CMS commands

CMS commands fall naturally into five categories: file manipulation, compilation, execution control, debugging aids, and utilities.

Figure 4 CMS structure and input/output devices



*THIS CMS DISK IS NORMALLY A SUBSET OF A PHYSICAL DISK

The file-handling commands allow the user to create, copy, move, combine, update, print, and erase disk files. Other commands provide access to the tape units, printer, and card-reader-punch. The editor provided with CMS for creating and maintaining disk files is a context editor; it allows the user to utilize character strings for locating and changing records in addition to working with entire logical records. Under the CMS linkage scheme, all of these commands are available to programs being executed as well as to the user at the terminal.

The OS/360 language processors, Assembler (F), FORTRAN IV (G), and PL/I (F), are used by CMS, and the object programs produced may be executed under either CMS or OS/360 depending on the operating system facilities requested. Diagnostics from the compilers are printed at the terminal unless suppressed by the user or directed to disk. Because the CMS file system does not provide as many access methods as OS/360, some features of PL/I are not supported at program execution time.

Three other processors are also included: SNOBOL, a string processing language; BRUIN, an interpretive language; and SCRIPT, a text processor. BRUIN (Brown University Interpreter) was adapted

from the OS/360 version of BRUIN developed at Brown University, Providence, Rhode Island. BRUIN provides both immediate (desk-calculator-like) and deferred (stored-program) modes. The SCRIPT processor is designed for text-formatting functions.

The execution control commands allow the user to load his programs from single object decks or from a library of programs. He can pass a list of parameters to his program from the terminal and specify the point at which execution is to begin. To avoid using the relocating loader for each execution of the program, he can create a file consisting of an image of the portion of main storage containing his program and load that nonrelocatable copy back at any time. Since the loading commands can be accessed by programs being executed, overlay structures may be set up, and dynamic loading can occur. There is full interactive execution capability in CMS.

The user can also create a file containing a command structure ranging from a simple series of consecutive commands to a complex logical structure and then execute these commands by typing a single line. This capability is called EXEC and allows a user to develop his own command language as well as to automate complex tasks.

The debugging command in CMS is called DEBUG. It allows the user to stop his programs at predetermined points and examine his registers, program status word, and storage, and modify these if he so desires. This information may be typed out at his terminal or printed off-line. A program interruption also gives control to DEBUG, as does the external interruption caused by the EXTERNAL console function. The user may invoke routines to trace SVC calls, as well as the FORTRAN debugging package.

The utility functions in CMS provide tape-copying facilities, disk file comparing, a disk file sort, and the dumping of files either by name onto the console or by cylinder locations onto the off-line printer. There are commands also for converting files of fixed length records to variable length records and for converting files in BCDIC code to EBCDIC code.

Other miscellaneous commands give the user the facility to suppress the typeout at his terminal, to restore typing at his terminal once the typeout is suppressed, and to terminate program execution. The user can also obtain statistics on his file space.

System applications

Following are some of the ways to utilize the virtual machine capabilities to take advantage of the features of CP-67.

Virtual storage can provide up to 16 million bytes of addressable main storage for those who have the need to run large programs or store large structures of data in storage.

Virtual Systems/360 can be utilized with varying configurations. The ability to define virtual device addresses differing from the real addresses on the Model 67 gives CP-67 the ability to serve as backup for other Systems/360, even though the software may be address dependent. Combined with the concept of mini-disks, a large data base system requiring 100 disk drives might be developed and tested with only one large disk storage unit.

The hardware can be checked on-line by running diagnostic programs from a virtual machine along with the other time-sharing users.

Running multiple operating systems concurrently provides the installation a choice of features and eliminates scheduling conflicts. Such variations are: utilizing multiple protected copies of a given system; operating with different versions of releases as well as with specially modified operating systems; training operators and programmers on-line; and allowing each user to control his own system and effectively debug a program from his operator's console. System development work, including testing of "privileged" code can be performed during regular production hours since virtual machines can execute any privileged instructions, and users are protected from each other's mistakes. An example is generation of OS/360 during the prime shift without stopping regular OS/360 service.

Running multiple CMS machines provides a powerful general purpose time-sharing system with its context editor, language processors, interactive execution, and on-line debugging facilities. Languages and services not available in CMS, such as APL, RAX, and CPS, are available under CP-67. To provide such services normally requires a computer and all its resources to be dedicated to the particular service, whereas under CP-67 such a system could be provided by dedicating just one disk drive and a terminal. Figure 5 shows a mix of systems which were demonstrated simultaneously under CP-67 during the 1969 Spring Joint Computer Conference.

The virtual machine environment can also be utilized in the development of computer-to-computer communications. Figure 6 illustrates the technique used to test a computer network system. Each virtual OS/360 system contains a program controlling an attached data adapter unit line. The two adapter unit lines have been connected externally via the telephone system. In this case, the two operator consoles can actually be physically adjacent for control by a single programmer. Figure 7 shows the use of the resulting program between two CP-67 systems. The only effective difference

Figure 5 CP-67 as set up at 1969 Spring Joint Computer Conference

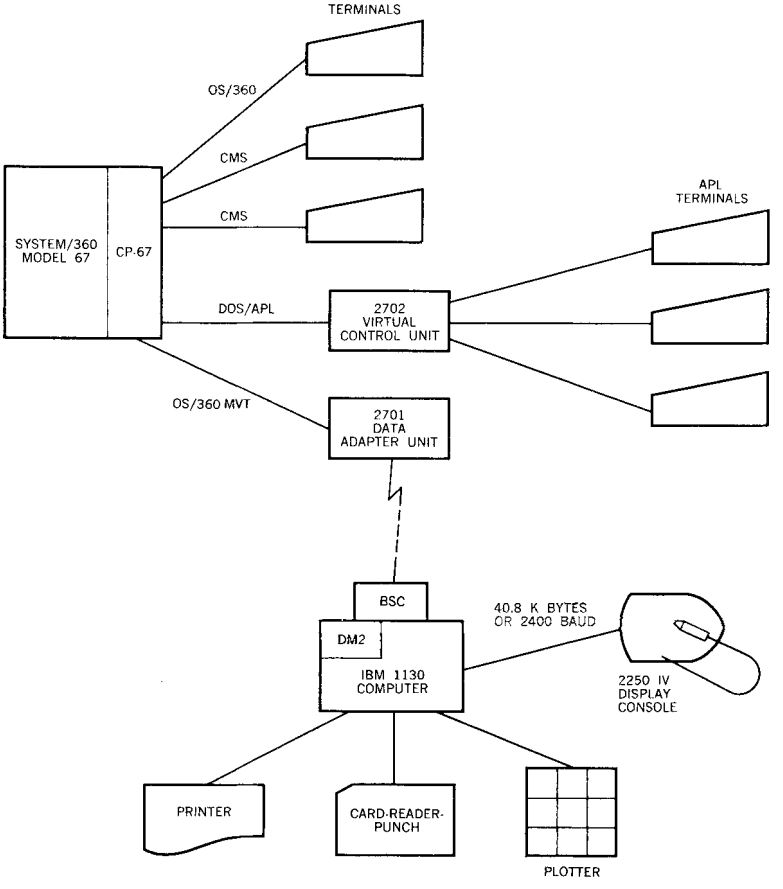


Figure 6 Testing teleprocessing on one machine under CP-67

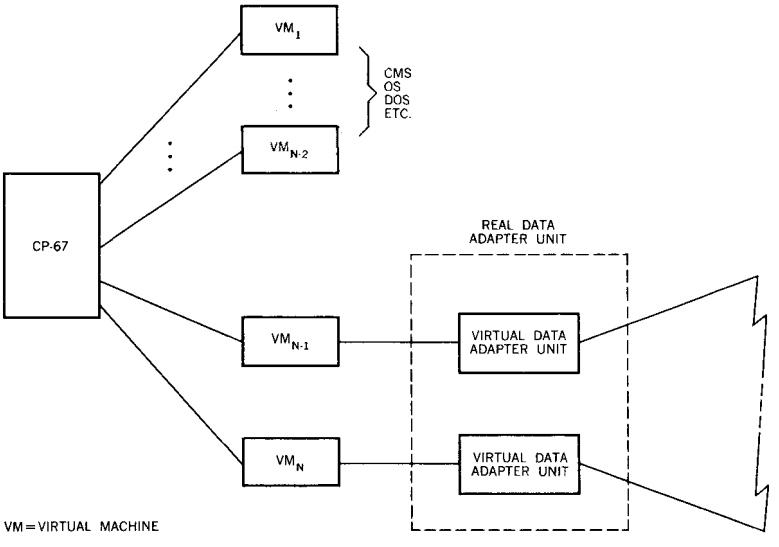
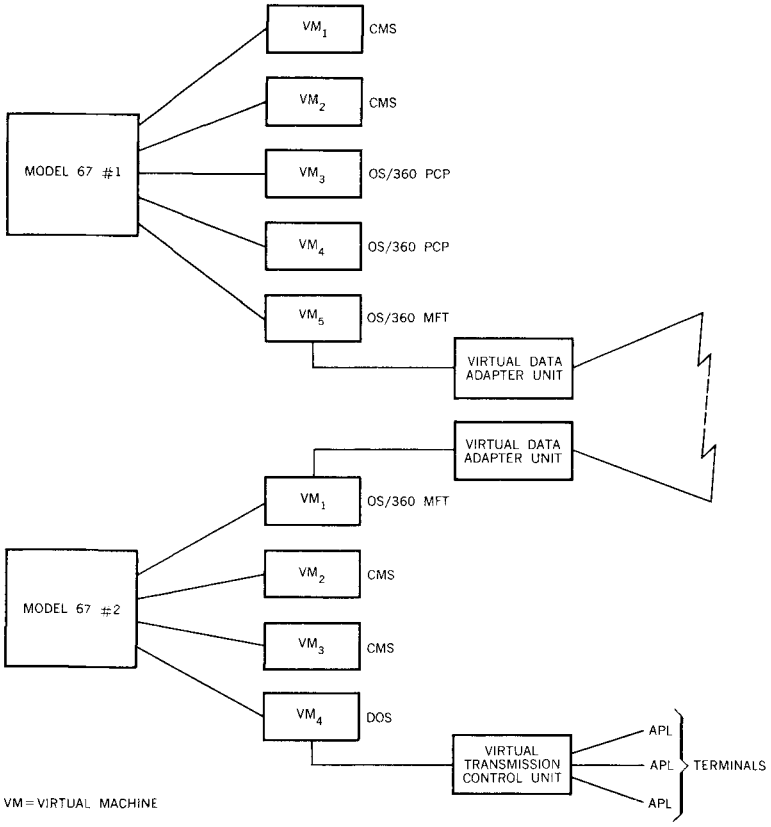


Figure 7 Teleprocessing checkout during normal operation of CP-67



between Figures 6 and 7 is the different telephone lines used in the connection. The final communications package, based on OS/360 can, of course, be used under normal System/360 operating conditions without CP-67.

CITED REFERENCES

1. L. W. Comeau, A. B. Lindquist, and R. R. Seeber, "A time-sharing system using an associative memory," *Proceedings of the IEEE,* **54,** No. 12, 1774–1779 (December 1966).
2. R. J. Adair, R. U. Bayles, L. W. Comeau, and R. J. Creasy, *A Virtual Machine System for the 360/40,* Cambridge Scientific Center Report 320–2007, International Business Machines Corporation, Cambridge, Massachusetts (May 1966).
3. *System/360 Model 67 Functional Characteristics,* A27–2719, International Business Machines Corporation, Data Processing Division, White Plains, New York.
4. G. M. Amdahl, G. A. Blaauw, and F. P. Brooks, Jr., "Architecture of the IBM System/360," *IBM Journal of Research and Development* **8,** No. 2, 87–101 (April 1964).

GENERAL REFERENCES

1. J. N. Bairstow, "Many from one: the 'virtual machine' arrives," *Computer Decisions* **2,** No. 1, 28–31 (January 1970).
2. M. S. Field, *Multi Access Systems—The Virtual Machine Approach,* Cambridge Scientific Center Report 320–2033, International Business Machines Corporation, Cambridge, Massachusetts (September 1968).
3. D. D. Keefe, "Hierarchical control program for systems evaluation," *IBM Systems Journal* **7,** No. 2, 123–133 (1968).
4. *CP-67/CMS,* Program 360D–05.2.005, International Business Machines Corporation, Program Information Department, Hawthorne, New York (June 1969).