To support the smooth running of a VM/370 installation, performance measurements of various types are desirable. This paper describes a range of measurement facilities that have been developed for VM/370 for use both on-line and off-line at the level of the users (general user, operator, and system analyst) and the installation management.

Performance measurement tools for VM/370

by P. H. Callaway

Measurement facilities, broadly speaking, have been implemented at several levels of the Virtual Machine Facility/370: the user level, the system operator level, the system analyst level, and the installation management level. The user needs a simple measure of the utilization and contention for resources on the system so that he may plan for a productive terminal session. The system operator needs to know when the system is overloaded or performing in some anomalous fashion so that he may take corrective action or summon a system analyst for an in-depth analysis of the situation. The systems analyst, when equipped with a good working knowledge of the system, may be able to diagnose some system problems using on-line snapshots of critical system parameters, but he will need a detailed measurement facility for a wide range of tasks, for example, tuning the system, validating and testing performance sensitive changes or enhancements, evaluating alternative scheduling or paging algorithms, or supporting virtual machine users whose programming systems need measurements on their mode of operation in the virtual machine environment. Programmers, involved in system modeling, simulation, and other areas related to computer science, may require the same kind of data as the system analyst but in more detail. Finally, installation management needs a wide range of data of a summary statistical nature to track the utilization of all the hardware and ensure that the installation is correctly configured to handle the normal load in an efficient and responsive manner.

The Virtual Machine Facility/370 (VM/370) performance measurement tools are three separate entities. The first, known as

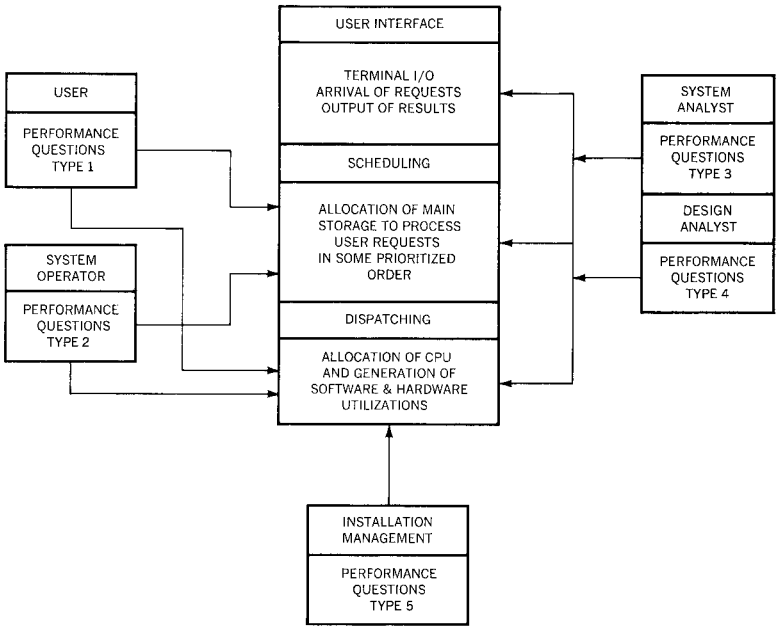
Load Indicators, is a control program service providing low-overhead console functions for observing various aspects of load on the system in real time. The second, known as VM/Monitor, is a general purpose control program service for collecting a wide range of data relating to most aspects of performance measurement. The first two services are known collectively as the VM/370 Measurement Facility and are used via the INDICATE MONITOR console functions. The third service is an optional data reduction system known as Statistics Generating Package for VM/370, or VM/SGP. It is available as an Installed User Program. VM/SGP consists of a data selection and reporting language, a translator, and a library of reduction programs to handle most classes of VM/Monitor output.

The purpose of Load Indicators is to provide the users, analysts, and the operator with the means to observe the load conditions on the system, to varying degrees depending on necessity and the levels of authority granted them, such that they may base their respective dealings with the system on data obtained in real time, not on guesswork. In the case of performance problems, real-time data may be of limited usefulness and may only serve to confirm that a problem exists. There may be no way of understanding, from a small amount of on-line data, the exact nature of the problem, how it arose, or how it may be resolved.

VM/Monitor provides a general-purpose mechanism for collecting a large variety of data on magnetic tape for later reduction and analysis. In most cases, data collection takes place with insignificant overhead and, thus, minimal impact on the system being monitored. (Reduction of the data may, of course, constitute a significant load but may be done with batch facilities.) Therefore, together with a reduction facility, permanent records of summarized measurements of load and performance may be maintained, and problems may be thoroughly analyzed and resolved with definitive results. Less critical concerns such as characteristics of user load and ways of improving them may be studied at leisure but in great detail if necessary. The great flexibility of VM/SGP may be employed to massage the raw data into forms most acceptable by management or which carry the greatest impact for the given circumstances. It is impossible to predict all the ways in which it may be desirable to present data, and so in place of a fixed set of report generating programs, VM/SGP provides a reduction language allowing each user to tailor reports to his own liking.

The facilities described have, to a large extent, been developed from the experience gained in measurement of the virtual storage systems, CP-67 and TSS/360, and the batch system, LASP/OS/MVT, at the computing center of the Thomas J. Watson Research Center.[1,2]

Figure 1 Schematic of the performance questions that may be asked of the VM/370 system



Categorization of measurement requirements

Readers first exposed to the wide range of measurement requirements likely to occur in a VM/370 time-sharing system might find it useful to have these requirements categorized and tabulated. The distinctions drawn between different classes of personnel and different types of performance questions are somewhat artificial and not at all rigorous but are introduced solely for the purpose of providing some organization in what may otherwise appear to be a confusing picture. What is important is that a variety of measurement facilities have been developed that allow gathering of data at varying levels of detail, for analysis in real time or off-line, as required by the situation at hand. In fact, users or system analysts or operators may be capable of making use of the full range of data.

Questions that may be asked relative to the performance and utilization of VM/370 are depicted schematically in Figure 1. The VM/370 control program is represented by three components that perform the functions of controlling interactions with the user, scheduling the user for main storage occupancy, and allocation of the CPU. In the schematic, the questions asked are linked to these three components according to the likely source of the answers. The questions themselves are grouped according to type of user posing them and are tabulated in Table 1. The list is not intended to be complete, rather it is intended to provide representative examples within each category.

Table 1 A categorized list of questions relating to the performance and utilization of a VM/370 system

Types of questions	*Questions*	*Comments*
Type 1: User-Level Questions	What is the mean utilization and contention for major resources? And therefore what sort of response should I expect in my forthcoming terminal session? During execution, what are my storage and CPU requirements, and how may I reduce these requirements to improve my performance?	The first requirement is easily met by a control program console function, and with experience a user may soon learn to relate response time to observed load conditions for his particular application. The second requirement may be met in a limited fashion by giving the user the ability to examine at will the total resources used by his program at any point in time, but the detailed tracing of the user's execution characteristics using a systems analyst tool may provide the only complete answer.
Type 2: System Operator-Level Questions	Is system overloaded or performing in some anomalous fashion? Is any one user contributing in an exceptional way to the situation? Can I take effective corrective action?	These questions may be answered by observation of control program load monitoring figures and use of further on-line diagnostic aids. The effectiveness of corrective action will then depend largely on the control facilities of the Scheduler and Dispatcher. System analysts may need to be consulted on interpretation of data.
Type 3: System Analyst-Level Questions	What are the work demands being placed on the system? What are "think times" and system response times, and how are they related to the observed utilizations and performance of the system? How well are resource allocation algorithms working? Can they be improved and can the system be tuned for the observed loads? Can I measure improvements in performance that may result? Are there any performance bottlenecks in the system or are there any anomalous performance situations that may require detailed analysis before diagnosis is possible?	These questions may be best answered by software monitoring with sampling and trace capability. Analysis of such data requires detailed knowledge of the control program internals. With experience, a knowledgeable systems analyst may be able to diagnose bottlenecks or anomalous performance situations with simple on-line diagnostic aids, and the ability to take immediate corrective action may outweigh the hazards associated with a certain degree of guesswork.
Type 4: Design Analyst-Level Questions	Can I gather data characterizing the loads put on the system by various categories of users? Similarly, can I collect data on the overall utilization of the hardware and performance of the software, thus providing a load profile of the system as a whole? Can I trace the execution of a particular program running in virtual storage and find out how reordering and restructuring of the program may lead to faster execution with less paging? How does the design of the system page replacement algorithm affect the performance of individual programs, and which is more important to the overall performance of the system, the algorithm or individual program design?	These questions may be answered by a software monitor with detailed trace capability. Analysis of such data requires detailed knowledge of the control program internals.
Type 5: Installation Management-Level Questions	Is the configuration optimum for the observed loads? In fact, are utilizations and responses acceptable? What are the usage growth rates and what are the expected configuration extensions likely to be?	These questions may be answered by a software sampling monitor or, to some extent, by a hardware monitor. System analysts may need to be consulted on interpretation of data.

The components of the VM/370 control program as depicted in Figure 1 may need further explanation:

User interface. Input data collected at this interface is the primary source of information defining the load on the system; output data is the system response. These data (when identified by the user's identification (userid) and time of day) are essential to an understanding of the relationship between user activity and system performance, and are of interest to psychologists as well as system analysts.

Scheduling. For installations with small amounts of main storage, the scheduler is effectively a funnel, and contention for main storage is handled in some priority order according to installation-defined biases. Contention of this kind may be the primary cause of delays to responses, and an understanding of the conditions of the various queues in the scheduler is a prerequisite to understanding the performance of the system.

Dispatching. Once the scheduler has chosen the (multiprogramming) set of users to occupy main storage, the dispatcher will prepare the users and give them the CPU in some priority order calculated by the scheduler. For installations with large amounts of main storage, there is no funneling effect in the scheduler, and the task of scheduling is reduced to that of determining dispatch priority in order to share the CPU in some equitable way among the users. Under these circumstances, the responsiveness of the system is more related to expansion factors[3] in execution because of the sharing of system resources (and possible contention for them). Observable conditions of hardware and software utilization are generated by the activities of the multiprogramming set of users. Their density of privileged instruction execution helps determine the ratio between problem time and supervisor overhead for simulation of the virtual machine environment. Their use of virtual storage space determines the system paging activity and associated overhead. The I/O subset of executed privileged instructions, together with the I/O activity due to paging, determine the utilization and possible contention for channels, control units, and devices.

Implementation

The VM/370 Measurement Facility may be thought of in terms of function being provided at four user levels and is described in the following sections. Since the system analyst and design analyst both use the same service, the measurement facilities available to them are covered in this same section.

user level—the INDICATE command

Performance has been defined as the way in which a system meets the throughput and response expectations of a user.[4] Consequently, it is important, for two reasons, that a user of an interactive system should have some means of assessing the current load and contention on a system that he is about to use. Primarily, he should be able to plan his terminal session to match the indicated load, thus making most productive use of his time. Secondarily, his expectations should not be disappointed, such that performance from his point of view may seem to be poor. (It has been shown that degradation in the response of a system results also in degradation of user think time.[5]) It is generally agreed that programmer productivity is more important than numerically high and efficient utilization of computing resources. On this basis, the small amount of overhead required to support the INDICATE function is easily justified.

INDICATE is a console function with two options available to the general user of a VM/370 system.[6] (Other options available to the operator and systems analyst if authorized are described in a later section.) The first option (and also the default option) is INDICATE LOAD. Its response provides a smoothed measure of the utilization and contention for the major resources of the CPU and main storage. More specifically, INDICATE LOAD gives a measure of CPU utilization and the level of multiprogramming (the degree to which the CPU is being shared, and therefore an indicator of possible program execution stretchout due to time-sharing alone). INDICATE LOAD also gives a measure of ma storage utilization and contention for that resource as indicat by the scheduler queues. (Long lists of users waiting to be allocated space in main storage represent long delays before any of the CPU is received by a particular user.)

A sample reply to INDICATE LOAD might be as follows:

CPU — 100% Q1 — 05 Q2 — 02 STORAGE — 42% RATIO — 1.0

The smoothed value of CPU utilization is updated every 30 seconds and has a time constant on the order of five minutes. Its value is obtained from the total wait state times maintained by the control program and is computed from interval mean wait time. Its maximum value is 100 percent.

The contention for the CPU is represented by the smoothed values of the number of users in Q1 and Q2. The actual counts of users in queue, maintained by the scheduler, represent, for Q1, users engaged in trivial interactive work, and for Q2, users engaged in longer batch-type tasks. Users about to begin nontrivial computations, such as large compilations, will estimate system response based on the Q2 (and not Q1) value in conjunction with the remaining response variables. A user planning more interactive use of the system will estimate system response

based on the Q1 (and not Q2) value in conjunction with the remaining response variables.

The smoothed value of storage utilization is updated immediately prior to a user being dropped from queue, and thus its time constant varies with the rate of queue manipulation activity. Its value is obtained by sampling, not by integration, and is thus more accurate under heavy activity than under light activity. The value of STORAGE in the reply is computed from the sum of the estimated working set sizes of the users resident in main storage as a fraction of the number of main storage page frames available for user paging. Its maximum value may exceed 100 percent when the control program occasionally overcommits storage.

RATIO (an abbreviation of scheduler contention ratio) is arbitrarily defined as follows:

$$\text{RATIO} = (E + M) / M$$

where E = total current eligible users (waiting for response but not currently receiving resources) and $M = \text{Q1} + \text{Q2}$ = number of users in multiprogramming set (resident in high-speed storage receiving resources).

The smoothing effect for Q1, Q2, STORAGE, and RATIO is achieved by the following calculation:

$$\text{VALUE}_{(\text{NEW})} = 1/16(15 \times \text{VALUE}_{(\text{OLD})} + \text{VALUE}_{(\text{CURRENT})})$$

RATIO is also evaluated prior to a queue drop, and its resulting time constant is variable. When $E = 0$, then RATIO = 1.0, all active users are resident in high-speed storage, and the scheduler discrimination controls and user priorities are not in use. When $E \geq M$, and RATIO = 2.0, users begin to spend significant amounts of time in the scheduler eligible list, the user priorities and interactive and paging biases are in effect, and various rates of iteration through the queues will be experienced; throughput for users with the larger working set sizes will be noticeably degraded. When RATIO is greater than 3.0, some users performing nontrivial computations would be wise to seek alternative work activities or suffer low productivity. With the interactive bias applied, trivial interactions should not be so noticeably affected by the RATIO value.

A second option of the INDICATE command is available to the general user; it is INDICATE USER. The purpose of this option is to allow a user to find out more about the activity of his virtual machine in terms of the resources used and occupied and events that have taken place. The data obtained in response to the command are as follows:

PAGES: RES − 045 WS − 025 READS = 000114
WRITES = 000067 DISK − 0128 DRUM − 0000
VTIME = 000:01 TTIME = 000:09 SIO = 000032
RDR − 000000 PRT − 000000 PCH − 000000

The first line of the response gives all data from the user's virtual machine control block (VMBLOK) relevant to his paging activity and resource occupancy. RES is a snapshot of the current number of user's virtual storage pages resident in main storage, and WS is a snapshot of the most recent system estimate of the user's working set size. READS is the total number of pages read for this user since he logged on or since the last ACNT command. (When the operator issues ACNT, the account command, most major resources used by each active virtual machine up to that point in time are punched onto account cards and the VMBLOK fields from which they were derived are reset to zero or an initial value). WRITES is the total number of pages written over the same time period. DISK is a snapshot of the current number of virtual pages allocated on system disk paging space for this user, and DRUM is the number allocated on system drum paging space. (RES, WS, DISK, and DRUM are not reset by the ACNT command.)

The second line of data gives user CPU usage and accumulated I/O activity counts. VTIME is total virtual time since logon or since the last ACNT command. TTIME is the total virtual time plus simulation time for the user for the same time period. SIO is the total number of nonspooled I/O requests issued by the user over the same time period. RDR, PRT, and PCH are counts of the total numbers of virtual cards read, virtual lines printed, and virtual cards punched over the same time period.

By recording the system response to the INDICATE USER command at frequent intervals throughout the execution of his program, the user may obtain a picture of his execution characteristics in the virtual machine environment.

load monitoring and diagnosis

When an interactive system is overloaded and responds poorly, the operator is often subjected to calls and messages from irritated users. It is, therefore, prudent to provide the operator with definitive indications of overload conditions and with facilities for determining which users are contributing most to the situation.[7] The operator may then ask the dominating users to postpone or moderate their activities until the response to the INDICATE command shows that a lighter load is being handled, or he may lower their priority. Furthermore, a persistently overloaded system is probably undesirable from an installation management point of view as well as from a programmer productivity point of view. There should therefore be simple ways of communicating to management just how often a system is in such a state, so that adjustments in configuration or reduction in user load may be planned.

Two possibilities exist for the monitoring of overload conditions, using the VM/370 Measurement Facility. First, by periodically comparing the response to the INDICATE LOAD command with some arbitrarily defined limits for the configuration, overload may be detected in real time and reported accordingly. The alternative is to use VM/Monitor to collect utilization data and to tailor a reduction report, using the same overload criteria, to summarize the amount of time spent in overload. The first method is more appropriate for use by the operator; the second would be used by the system analyst and will be discussed further in a later section.

An installation may therefore provide its operators with arbitrarily defined overload criteria and request that periodic checks be made using the INDICATE LOAD command. VM/Monitor data has been used to determine these overload criteria for the VM/370 system running on a two-megabyte System/370 Model 158 at the Thomas J. Watson Research Center (see Appendix A). Once a persistent overload condition is identified, a number of steps may be taken to clarify the situation, using some additional options of the INDICATE command. These options should be available to the operator and system analyst and may be used as follows:

Use INDICATE QUEUES to find out who the current active users are and how much real main storage (one major resource) they occupy. (The reply to this command contains a good deal more data and a full explanation will be given in a later section.)

Use the INDICATE USER command to display the resources used by each of the active users. (The operator should have the authority to look at the resource usage fields in the VMBLOK of any user on the system.) Determine if anyone is receiving a disproportionate share of the CPU – the other major resource besides main storage. The task of determining what resources the active users have received over a given interval is best handled by creating spool files of the above responses at the beginning and end of the interval, and then reading them into a program. This program, running in the user's virtual machine, then calculates the resources used by these tasks active over the whole period, that is, the long-running Q2 users. Such a program would also be widely used by the system analyst while looking for exceptional users in real time.

If any user appears to be receiving an inappropriately large share of the resources, inform installation management. A study of that user's application may reveal ways of reducing its load or, in general, improving its performance in the virtual machine environment. Alternatively, it may be apparent that the application is more suitably run on a batch system and not on a

time-sharing system, since with the former the large resources required to run may be scheduled appropriately with less noticable impact on other users. It may be desirable to run certain applications during low load or nonprime shift periods (with time-sharing, each user wants to get his work done in prime shift, often treating the computing resources as infinite and ignoring resulting load conditions and inefficiencies).

If RATIO is greater than 1.0 and the discrimination abilities of the biased scheduler are in effect, then user directory priorities may be changed to force the more demanding users into the background. The INDICATE LOAD response will not show any relief of the overload condition; however, certain users may be now cycling through the scheduler lists at a slower rate than others, and more reasonable expansion factors should be experienced by the less-demanding users. The INDICATE USER command should reflect the change in resource usage of the various users.

real-time and off-line measurements

Early use of experimental versions of the VM/370 Measurement Facility confirmed the desirability of being able to study the various aspects of a load both in snapshot, or summary, fashion in real time and in detail off-line at leisure. Several options of the INDICATE command provide some on-line diagnostic capability for the system analyst. These options enable him to check for obvious bottlenecks without having to resort to more lengthy and detailed analysis. VM/Monitor provides the ability for more thorough and detailed analysis off-line when considerable thought and planning are required to massage many types of data into the form necessary to either solve a problem or present an argument in a convincing manner based on data. The experience gained in the development and use of these tools has led to the VM/370 Measurement Facility now generally available. Considerable experience has been gained in the performance analysis of VM/370 systems based on sampled data obtained by a virtual machine. The general principles of analysis are well-documented in Reference 8.

INDICATE command for system analysts

A system analyst, when confronted with a load situation requiring some analysis, would first ensure that the VM/Monitor data collection tool had been started appropriately and then would proceed to attempt to understand the problem on-line using the INDICATE command.[9] If the problem cannot be solved on-line or simply goes away, then VM/Monitor will have captured all the relevant data for later analysis.

Next, the INDICATE LOAD option would be issued to obtain values for the current utilization and contention for the CPU and main storage. Additionally the QUEUES option of the INDICATE command would provide a sample of the current active users,

their userids (user identifications), their execution status, the amount of main storage occupied, and the current working set estimate. This command should be issued several times so that only persistent conditions may be acted upon. Alternatively, the resources used by the continually active users over some interval (say, one minute) may be determined by processing spool files of the responses to the INDICATE commands as mentioned previously.

A sample response to the INDICATE QUEUES command is as follows (page frame counts are given in hexadecimal):

```
useridA Q1 PG 001/010   useridB Q2 RU 045/025
useridC Q2 IO 055/024   useridD E2 -- 000/045
useridE E1 -- 000/010
```

This response indicates that there are five active virtual machines, three of which occupy main storage (one in the interactive queue (Q1) and two in the more compute-bound queue (Q2)). UseridA, in Q1, has so far brought in one page, has a working set estimate of 10 pages and is currently in page wait (PG) state. UseridB, in Q2, is executing and is the current RUNUSER (RU); he has 45 pages in main storage and has a working set size estimate of 25 pages. UseridC, in Q2 also, is currently in I/O wait state (IO) with 55 pages resident and a working set of 24 pages. UseridD and useridE are waiting in the eligible lists of Q1 and Q2 respectively, have no pages resident, are waiting for the CPU (--), and have the indicated working set size estimates. An additional status flag, PS, indicates that a virtual machine has issued an enabled wait PSW (program status word). This PSW wait state may be indicative of I/O contention.

If the CPU is the only bottleneck, then RATIO will be 1.0 (all active users resident in main storage), and the majority of active users will be waiting for execution, as indicated by the '--' status flag, and, of course, CPU utilization will be close to 100 percent.

If main storage is the bottleneck, then RATIO will be greater then 1.0 and a corresponding proportion of the active users will be displayed in the scheduler eligible lists. Storage utilization will be relatively high (except on small systems with low levels of multiprogramming). If response is sporadically poor, then it is likely that one or more users have working set estimates comparable with the number of pageable page frames, and under storage contention conditions, are periodically blocking the scheduler eligible lists.

If I/O is the bottleneck, then a significant number of the multiprogramming set in each sample will be displayed in I/O wait or PSW wait status, and the I/O option of the INDICATE command

should be employed to display the real devices being waited on. If more than one user is waiting persistently on the same device, then the contention condition is clear.

A sample reply to the INDICATE I/O command, displaying such a condition might be as follows:

useridA 212 useridB 210 useridC 212 useridD 212

In this example, three out of four users currently in I/O wait state are queued on the same device.

If paging is the bottleneck, then a significant number of the multiprogramming set in each sample will be in the page wait state (PGWAIT), and the PAGING option of the INDICATE command should be employed for further analysis. This option displays user paging space residency counts and is only relevant when the installation is equipped with drums as primary paging devices and with other direct-access facilities as secondary paging devices. When the primary device is full and performance is degraded by users spilling over to the slower devices, the INDICATE PAGING ALL command will show which users are occupying space on which device. Consider, for example, a virtual machine running a large operating system that was allocated large amounts of primary paging space at IPL (initial program load) time, but then became inactive. This user is occupying a critical resource but is not putting it to good use, and, in fact, may be contributing significantly to system performance degradation. (The INDICATE PAGING WAIT option displays the same data but only for those users in the multiprogramming set currently in PGWAIT). The form of each entry in the reply to an INDICATE PAGING command is simply:

useridA 128/000

where 128 is the number of pages resident on drum and 000 is the number on disk.

In addition to the above, the system analyst has the authority to use the INDICATE USER option, specifying the userid of the virtual machine whose resource usage he wishes to study in more detail in relation to the problem at hand.

VM/Monitor and VM/SGP

The VM/Monitor and VM/SGP tools provide a general purpose data collection mechanism and a general purpose data reduction language and report generator, which has been used to build a library of reduction programs in common format. The data collection tool runs as a privileged component of the control program, providing trace and sampling data using the MONITOR

CALL instruction and facilities for specifying timing interruptions. The monitor call instruction available on System/370 CPUs greatly facilitates the collection of event-driven trace information. By imbedding monitor calls at strategic places throughout the control program of VM/370, critical transient performance information may be gathered, such as the way in which user working set sizes are estimated, and the scheduler queues may be manipulated to establish an optimal level of multiprogramming for the given system. A monitor call instruction may be used to specify a class number in the range zero to 15 and a code number computed from base and displacement fields. The monitor call instruction is fully described in the *System/370 Principles of Operation*.[10] The implementation of VM/Monitor[9] takes advantage of this structure by categorizing data collection by function into separate classes with each class identified by a suitable keyword.

The execution of a monitor call gives rise to a program interruption (code hexadecimal 40) if that class of monitor interruption is enabled. The classes of monitor call interruptions enabled at any time are defined by the contents of a 16-bit mask field in one of the control registers. Thus, a simple set of commands is implemented to permit an operator to define the contents of that control register via selection of the appropriate keywords and thereby enabling data collection of the type best suited to solve the problem in mind at the time. The operator command MONITOR ENABLE followed by a string of keywords results in the storing of a mask field that is loaded into the control register later when data collection is actually initiated with the MONITOR START command. The command MONITOR DISPLAY is provided to remind the operator which classes of data collection have been implemented and what are the corresponding keywords. Modifications to the program interruption handler have been made to divert supervisor state monitor call interruptions to a new program that performs the functions of decoding, data collection, and data output. (Problem state monitor calls are simulated and reflected back to the issuing virtual machine.)

The class and code numbers of the monitor interruption are placed by the hardware in reserved locations in page zero of main storage. Thus the monitor interruption decoder simply uses these stored values to index into branch tables and reach data collection routines unique to any particular interruption. Additional subroutines provide means for creating standard header records and handling the buffering and output of the data. I/O supervisor facilities are utilized to actually perform the output and handle any error conditions. A tape drive is used as the recording medium since large volumes of data may be collected when using trace techniques. The actual initiation or

termination of data collection is controlled by the MONITOR START or STOP commands with an additional parameter with which to specify the address of a tape drive (which is reserved for the system for the duration of the data collection session). Sampling data collection is implemented by the specification of a timer interruption and an interruption return address via the control program timer request block facility. This procedure provides a means of passing control to a given data collection subroutine at precise intervals of time. The subroutine then acts as though a particular class and code of monitor call has been executed and samples all relevant system event counters and accumulators, thus generating a record of sampled statistics relevant to the utilization and performance of the system.

All records created by the collector are prefixed with a header section containing the class and code of monitor call and the time of day of the occurrence of the interruption. All records are in standard variable length format and are blocked in a page size of 4096 bytes. The time stamp provides a simple mechanism for measuring the elapsed time between the occurrence of any two events, and, in fact, has been used for measuring control program overheads directly. A typical monitor tape generated in the above manner usually contains a mixture of variable length records of different classes and codes both sampled and traced. The general purpose design of the collector ensures that extensions to its data collection capabilities can be very simply achieved by the astute placing of monitor call instructions in the VM/370 control program, and by definitions of new classes of monitor calls or additions to existing ones.

classes of data collection

The classes of data collection implemented and the uses to which they may be put are as follows:

The PERFORM class provides sampled data only, which when reduced, yields summaries of the overall utilizations of the CPU and main storage, of paging statistics, privileged operation usage statistics, and various interruption and call statistics.

The SCHEDULE class provides trace data to monitor the flow of work through the scheduler. It shows how working set sizes are estimated (by displaying the values of variables from which they were computed) and thus how the level of multiprogramming is established. By revealing user execution characteristics in queue and the resulting scheduler priorities, the trace shows how the discrimination facilities of the scheduler affect the service rates of different kinds of users under storage-bound conditions. Anomalous conditions of low CPU and main storage utilizations with high contention for main storage have yielded to analysis using the SCHEDULE class of data, resulting in a number of improvements to the system. (See Appendix B.)

The RESPONSE class provides trace data relating to all terminal transactions with the system. With this class of data, the important relationship between user activity and system scheduling activity and resulting system performance may be accurately established. These data have been most usually required when an unexpected situation in the control program scheduler queues has required a knowledge of exactly what the users were doing for full understanding and problem diagnosis. The data may also be used together with SCHEDULE data to obtain user think times, system response times and expansion factors, and for full analysis of command language usage.

The DASTAP class provides sampled data on the activity counts of all DASDs and tape devices on-line at the time the monitor is started. The associated data reduction provides time-stamped I/O activity summaries which may be correlated with the PERFORM class reports on system utilization and performance. In fact, one reduction program in the library combines summaries from both PERFORM and DASTAP classes.

The SEEKS class of data collection traces every start I/O request for a DASD device. Together with data reduction, this class provides the ability to study disk arm contention problems and possible bottlenecks in the paths to the devices.

The USER class of data provides sample information about the amount of resources each virtual machine on the system was using. This information may play a supporting role to the PERFORM class when it is felt that one or more users are dominating the system or getting an unfair share of resources. This class has most often been used when a system has had so much high-speed storage that all users were resident, the scheduler discrimination facilities were not utilized, and the dispatcher alone attempted to distribute the CPU utilization.

The INSTSIM class of data collection traces every privileged instruction simulation. When a programming system is running in the virtual machine environment, the single most significant source of overhead may be privileged instruction simulation. The INSTSIM class provides the ability to derive information on the frequency of use and virtual storage location for each privileged instruction type encountered. This information is useful in optimizing the performance of a programming system in a virtual machine environment, particularly when an operating system is involved that usually runs on its own real hardware.

The SYSPROF class of data collection is a separately controlled extension to the SCHEDULE class. It provides more detailed information on the overall performance characteristics of a system, and is intended for future study of installed systems.

Measurement support for the study of the execution characteristics of virtual machines is provided by the SCHEDULE and INSTSIM traces. The scheduler drop queue records yield information concerning number of page frames read, stolen, resident, and referenced while in queue and the resulting projected working set size, also the amount of simulation and problem program the CPU received during that time. The number of I/O operations to both DASDs and spooling devices are also recorded, the latter in terms of lines printed and cards read and punched. Typical user profile type of data may be gathered during user benchmark sessions, thus displaying his activity in that special environment.

The major requirement for a general purpose data reduction facility is met by the adaptation of the Statistics Generation Package (SGP) for use under CMS with VM/Monitor data. This version, known as VM/SGP, is available as an Installed User Program (IUP).[11] (SGP is a Field Developed Program for use with OS and OS/VS SMF data.) This adaptation makes available the high-level selection and report generation language that is a key feature of SGP.[2] Thus, by knowing the symbolic names of the data items available from VM/Monitor and by being familiar with the syntax of the SGP language (very similar to PL/1), a system analyst is able to generate fairly sophisticated data reduction programs, tailored precisely to suit his own requirements. The production of such programs typically takes as little as half an hour, depending on the load on the VM/370 system being used. Three phases are involved in the production: the creation of the selection and report generation requests, the translation of the source into PL/I source, and the compilation of the source using the PL/I Optimizing Compiler. All aspects of reduction program production and use are supported with CMS EXEC files to smooth and facilitate the work. Working with these facilities, a library of programs has been built up and used to answer a large range of questions relating to the performance and tuning of the VM/370 system.

Some examples of the use of VM/Monitor, both independently and in conjunction with the INDICATE command, are given in Appendix C.

installation management data reduction

Reduction of the data obtained with the collection tool described previously may be performed to provide information to management. The main function is to produce a one-line summary report of each day's activity. This report may be used to construct weekly and monthly reports of the utilization and contention for the major resources of the system. The daily report includes means for the following variables: CPU utilization, problem time, idle time, page wait time and I/O wait time, number of users logged on, users in eligible lists, users in dispatch queues (level of multiprogramming), and main storage utilization. Finally, the

number of cylinders allocated on the primary paging device is given in order to indicate when that device is filled and when overflow to slower paging devices may start to degrade overall performance. Thus, management may have available numbers concerning the utilization and contention for their critical resources and may plan accordingly. In addition, knowing flexibility of the measurement tools and the ability of the system analysts to exploit them, management may have the confidence of expecting almost any question to be answered within reasonable periods of time should it not have been possible to answer them with existing programs.

Summary

The data collection, display, and analysis tools and techniques found desirable to support the smooth running of a VM/370 installation appear to be diverse, yet their use must be well coordinated. In summary, a user's productivity may be enhanced by linking the nature of his terminal activity to the performance of the system by making available to him measures of utilization and contention for major resources. Operators and system analysts may monitor the performance of the system in real time and attempt to diagnose and correct undesirable load situations. Full analysis of severe or complex performance problems may be conducted off-line when necessary and special user measurement requirements may be satisfied. Basic trends in resource utilization and system performance are available to management in summary form.

Although these facilities have been developed and put to use subsequent to initial installation of the system, they appear to provide the kind of services that should be an integral part of control program functions, since they help users, maintenance programmers and installation management to strive toward making optimal use of their computing system.

Appendix A: Overload

The term overload has, up until now, been used rather loosely, but it is possible to define it empirically in terms of values of the response to the INDICATE LOAD command by examining performance characteristics of VM/370 running as a CMS time-sharing system under a wide range of load conditions. Such a study has been made of the performance of the two-megabyte System/370 Model 158 running a 2.1 version of VM/370 at the Thomas J. Watson Research Center. The primary objective of this work was to study user job execution expansion factors under heavy load in order to be able to define the reasonable upper limit that a user could be expected to tolerate. The utilization and conten-

tion for the CPU and main storage at this upper limit would then provide overload criteria in terms of the response to the INDICATE LOAD command.

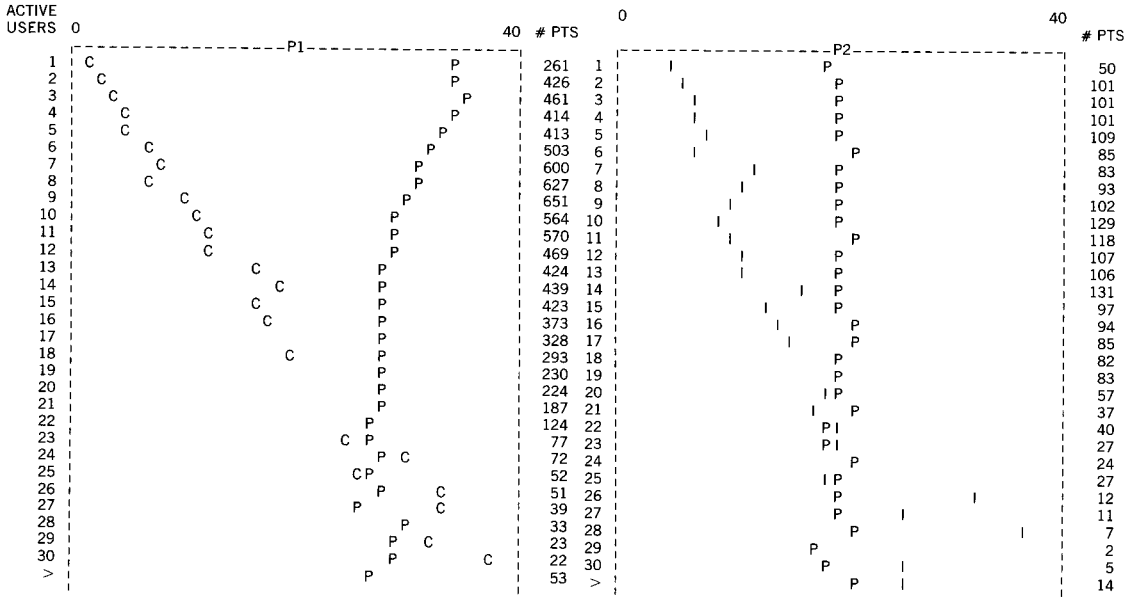
The expansion factor of a job is defined as the ratio between the elapsed time of the job when the system resources are being shared by many such users, and the elapsed time of the same job when no other users are competing for resources. The wide range of jobs performed on the subject system made the direct measurement of their expansion factors a difficult task indeed. Therefore, it was thought that a good estimate of job expansion factors could be obtained by measuring the average time slice expansion factor in terms of the elapsed time between consecutive time slices and the total CPU time received during a time slice. Since it takes proportionately longer to accrue a time slice's worth of the CPU as the load on a system increases, there should be a direct relationship between that effect and the total job expansion factor.

The expansion factor curves displayed in Figure 2 are obtained from the VM/Monitor trace of scheduler activity. They are plots of the expansion factors in execution times for empirically segregated I/O-bound and compute-bound Q2 users against the total number of active users (where the latter are those users contending for main storage at each point in time). Each point on the curve represents the mean expansion factor for all observations at that level of active users. An observation is made when a user is involuntarily dropped from Q2 because he is "time-slice ended" (i.e., he has not finished his current command and wishes to continue immediately). At that point, the ratio is calculated between the time that has elapsed since he last became eligible for Q2 and the total amount of the CPU he received in that time.

The sequence of events in such a cycle is: the user is added to the eligible list for Q2 waiting to continue execution where he left off at the end of the last time slice. While resident in an eligible list, the user is receiving absolutely no resources, just accumulating the elapsed time factor in the ratio. When there is enough main storage space available, the user is added to Q2 and will begin to receive utilization of the CPU in a quantum by quantum manner determined by the dispatcher or the occurrence of page exceptions or I/O waits. At this point in the cycle, the user is accumulating amounts of the CPU, but elapsed time is advancing faster because he is idle while other members of the multiprogramming set execute and because he falls into the wait state for paging or an I/O operation. Finally, when he has accumulated a certain maximum amount of the CPU while in Q2, he is "time-slice ended" and the scheduler drops him from the queue. The ratio is therefore a time-slice-based expansion fac-

Figure 2 Expansion factor—active user correlation

P1—"C" = average time slice expansion factor for compute bound Q2 user. (In queue virtual I/O count <100 defines compute bound.)
P1—"P" = mean fraction of time slice CPU in problem state × 40 for compute bound user.
P2—"I" = average time slice expansion factor for I/O bound Q2 user. (In queue virtual I/O count >100 defines I/O bound.)
P2—"P" = mean fraction of time slice CPU in problem state × 40 for I/O bound user.



Not available with VM/SGP.

tor, hopefully representative of job-based expansion factors. Note that the number of data items that contribute to the mean value at each point on the graph is displayed.

Supplementary information is, in fact, needed to relate time-slice expansion factors to job expansion factors, since at the higher levels of active users, more overhead would be experienced because of increased paging overhead. Thus the problem state proportion of the CPU within each time slice might be expected to decrease with a corresponding increase in job-elapsed time. This increase would be in addition to the increases in job time due to the sharing of resources and contention for main storage (eligible list loading). Superimposed on top of the expansion factor plots are the mean problem state fraction of each CPU time slice at each level of active users.

These examples of expansion factor curves are taken from six hours of prime shift operation of the machine just described with as many as 88 users logged on under a wide range of load conditions. A number of interesting observations may be made about the two expansion factor curves.

The origin of the "C" plot confirms that when one compute-bound user is executing alone in the system he fully utilizes the CPU and his expansion factor is one.

The origin of the "I" plot indicates that one I/O-bound user running alone had an expansion factor of four since his I/O wait time is included; he thus utilized about 25 percent of the CPU.

The slope of the "I" plot is lower than that of the "C" plot, confirming that the Dispatcher favors I/O-bound users by giving them higher priority than the more compute-bound users.

Data not shown indicates that the two megabytes of main storage yielded an average of 350 pageable pages and a mean level of multiprogramming (Q1 + Q2) of 10. Thus, for all plot positions above about 10, the eligible lists become progressively more loaded (INDICATE LOAD RATIO value greater then 1.0). Points below this value occur with no contention for storage, points above occur with progressively more contention.

The "P" plot for compute-bound users indicates the expected degradation in the problem state because of increased paging overhead. However, by the time storage is fully utilized and the eligible lists have first come into use, the maximum degradation has occurred. Thus, the following deductions may be made about paging overheads above and below the point where storage is just filled (when the sum of the working set sizes of the active users equals the number of pageable pages on the system, and RATIO equals 1.0). When RATIO is greater than 1.0, each nontrivial user, as he receives consecutive time slices, cycles through the eligible list and has to completely restore his working set to main storage before continuing normal speed of execution. In fact, it appears that he is experiencing the maximum degradation in problem time ratio to total time, due to paging. No matter how much higher the level of active users moves beyond this point, the paging overhead for the user will not increase further; we effectively have a swapping system. At levels of activity below the loading of the eligible lists, the paging overhead required to maintain each user's working set gradually increases until storage is filled.

The deductions presented above are supported by studying the changes in system problem state and paging rates as the number of active users increases. The degradation in system problem

time and rise in system paging rate as the number of active users increases is displayed in the sample plots of Figure 3. Recall that the problem time ratios in the previous plots were accumulated from user drop queue records. The problem time displayed below is a total system value; it is accumulated by the control program and sampled every 60 seconds by VM/Monitor. Note that the maximum fall in problem time and the greatest slope in the paging rate curve have occurred at the level of 14 active users. Beyond this level, we have a well-behaved swapping system where further increases in expansion factors are due to more and more time spent in the scheduler eligible lists.

Returning to the previous plots, observe that for the I/O-bound user, the "P" plot is level. This indicates that I/O simulation is the dominant portion of overhead (average 55 percent CPU per time slice) and that any changes in paging overhead are insignificant.

Finally, returning to the question of criteria for overload, there are no knees or discontinuities on the previously discussed plots that could provide us with suitable cut-off points since it would be impractical to run below the point of maximum paging overhead. Thus, we are left with an arbitrary decision to be made in terms of what the maximum expansion factors are to which we should expose the users. Users themselves have indicated that they are prepared to force the system to indefinitely high expansion factors just to feel that they are getting something done rather than nothing at all. This could simply be a matter of ignorance, but in any case, the system should be protected from such misuse (and maybe users should be protected from themselves).

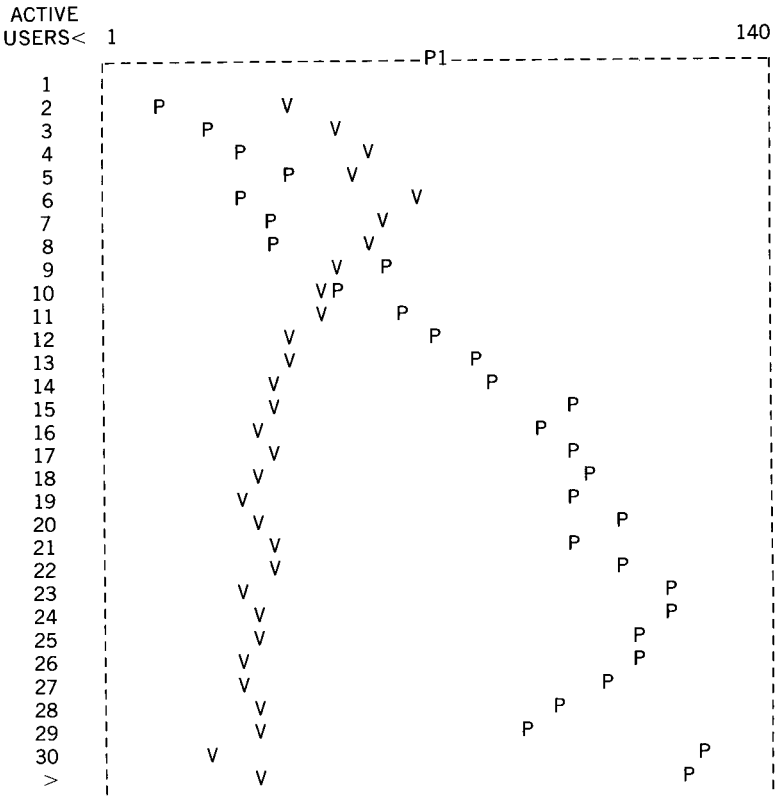
Which controls are available to prevent, for example, expansion factors from exceeding 20? From the expansion factor plots this corresponds to an active user count of about 20. Since the mean value of Q1 + Q2 has been found to be about 10, the loading of the eligible lists (E1 + E2) is about 10, and RATIO = (10 + 10)/10 = 2.0. Therefore, if we define overload as occurring when expansion factors exceed 20 for Q2 users, then the corresponding INDICATE LOAD response for the operator or system analyst to look out for is the following:

CPU − 100% Q1 + Q2 − 10 STORAGE − 100% RATIO − 2.0

The active-user/logged-user plot in Figure 4 provides us with one simple but not fully satisfactory choice for control. The plot indicates that on the average a user population count of 65 will yield an active user count average of 20 and, hence, an average expansion factor of 20. Thus, by restricting access to the system to about 65 users, we might hold the expansion factors within the desired bound. Three problems can be immediately associ-

Figure 3 Active user—problem time and paging rate correlation

P1 – "V" = Mean percentage of CPU time spent in problem state at each level of active user.
P1 – "P" = Mean paging rate at each level of active users.



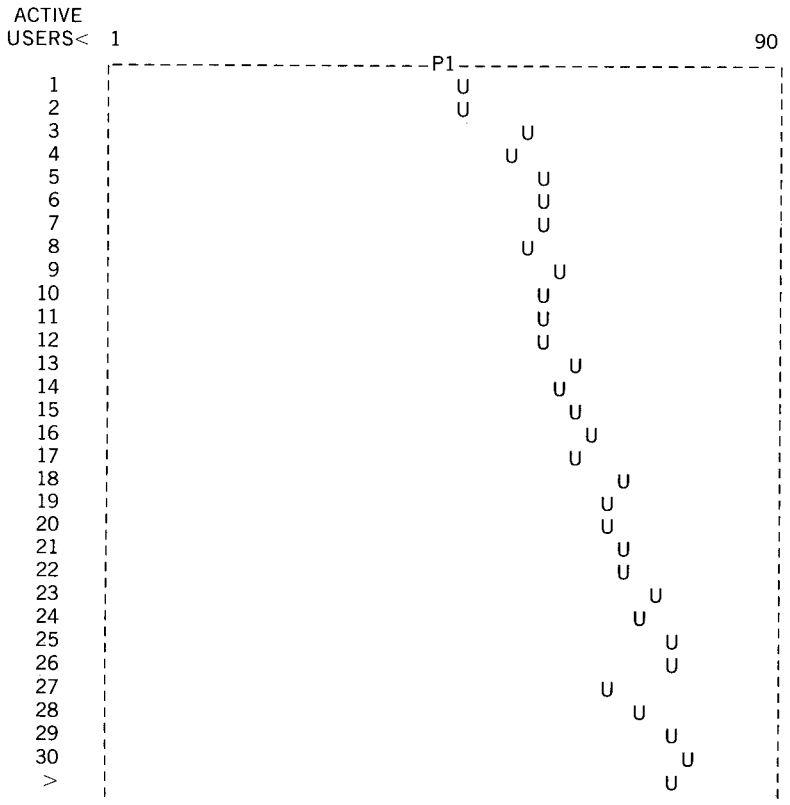
Not available with VM/SGP.

ated with such a restriction. First, a very wide distribution of values contributes to the mean value for logged-on users in the active-user/logged-user plot. Clearly the restriction could equally well lead to underloading or overloading of the system, the consolation being that on the average the system should be running well, and that the scheme would work much better for a large system than for a small one. Further, we are open to the criticism that we are denying access to the system to users wishing to perform trivial but essential tasks. In conjunction with this point, when access lines are a scarce resource, inactive users will tend to remain logged on, and so under-utilization is more likely to be the case.

In conclusion, therefore, it appears necessary that the system itself should be capable of limiting the demands placed upon it and thereby distributing its finite resources in an equitable way among a selected subset of users such that they all execute at a

Figure 4 Active user—logged user correlation

P1—"U" = mean users logged on to the system



Not available with VM/SGP.

reasonable speed. In effect, such a system would deny foreground execution to nontrivial jobs, once certain load conditions were exceeded, and would offer the choice of automatic shipment of the job to a batch facility, indefinite background execution with a locked keyboard, or termination of the current request and return to trivial activity.

Appendix B: An example of problem analysis

This appendix describes how an anomalous load and utilization situation on a System/370 Model 145, running VM/370, was analyzed with the aid of the data collection and analysis tools.[12]

The system was running with the data collection tool activated with classes RESPONSE and PERFORM. During the afternoon, the operator received complaints from users concerning an apparent complete lack of response to other than trivial commands. The

INDICATE command showed a CPU utilization of 10 percent, a main storage utilization of 15 percent, and a Scheduler Contention RATIO of over 3.0 indicating that there were twice as many users in the eligible lists as there were in the multiprogramming set. After 10 minutes or so the problem disappeared and normal service continued. The question was: how could the utilization be so low under such high-contention conditions? Later analysis of the measurement data selected 15 minutes on either side of the time of the reported problem and revealed the following situation. For a period of ten minutes, the CPU and main storage utilizations were indeed low with one user in Q2 the whole time and various users in Q1, one at a time, as they continued trivial-type interactions with the system. The number of users eligible for Q2 ranged from four to six. The bottleneck "freed" up when the one Q2 user dropped from queue. He had been occupying only four page frames of high-speed storage. The top priority eligible user for Q2 was immediately added to Q2, his storage requirements being equal to the number of available pageable page frames on the system. Since he was the top priority eligible user, he could not be skipped over even though users below him required less storage, and he was effectively blocking the scheduler. The question to be answered therefore was how could one user occupy a small part of storage for so long while apparently doing nothing.

The scheduler trace revealed that he had entered Q2 11 minutes ago, had only received four seconds of CPU time, and was in the I/O wait state for most of the time. Finally, the transaction trace from the RESPONSE class was consulted. These data reveal every input to the system and output from the system. The data is stamped with the time and user identification for easy matching with the scheduler trace. It was found that the user who occupied storage for 11 minutes was executing a small EXEC procedure which read data from tape. The tape drive was attached to the virtual machine but no reel was mounted; consequently, the read diagnose command issued by the CMS virtual machine resulted in an intervention-required message being sent to the operator and the user being set in the I/O wait state but not dropped from queue. The operator was unfortunately busy with other tasks and didn't get the reel mounted for some time, and it wasn't until the time that the reel was mounted that the blockage in the scheduler queues was relieved. As a consequence of this diagnosis, the usual procedures for problem identification and resolution were expedited.

Appendix C: More examples

This appendix documents in detail some of the ways in which the INDICATE command together with VM/Monitor and VM/SGP

have been used to improve the running and use of the VM/370 systems.[13]

INDICATE has frequently been used to check out load situations when 100 percent CPU utilization figures have been noticed for extended periods of time. On the occasions where one user can be shown to be receiving the major share of the CPU, corrective action may be taken. For example, a CMS user was found to be looping in a problem program bug, using up all spare CPU cycles and causing poor response for nontrivial users. When the user failed to respond to messages and telephone calls, he was removed from the system with a FORCE command and system performance and responsiveness immediately improved. When there are several users contributing to persistent compute-bound conditions, more detailed analysis has been found necessary using VM/Monitor. In fact, there is continued monitoring of exceptional use of the system, since it may be considerably easier to use a time-sharing system rather than a batch system, even when the latter service is provided on a machine well-suited to heavy compute-bound jobs. In cases where a time-sharing user executes very long compute-bound jobs, consultants may advise on use of the alternative systems or the use of the shipping facilities between the systems.

VM/Monitor has been used to study the characteristics of certain programming systems running in the virtual machine environment, with the intention of providing data to justify the use of the environment on the given real machine configuration. Paging characteristics were of particular interest in studying the Scratchpad system,[5] since it causes very little privileged instruction simulation under VM/370 and suffers its major source of overhead in paging. In fact, it was found in running a 768K version of Scratchpad on a 512K real VM/370 system, that paging overhead accounted for more than 50 percent use of the CPU. Since Scratchpad problem solving is usually a compute-bound operation, this loss of CPU cycles to paging operations on a small system almost dictates the use of a large, main storage system.

INDICATE and VM/Monitor have been used together to debug communications types of virtual machines. An early version of an IBM 2780 communications virtual machine was found to loop endlessly with high priority in the trivial interactive portion of the dispatchable list while waiting to receive data from a remote location. The virtual machine was repeating a start I/O, looking for the condition signifying the arrival of data; the designer did not realize that such activity in connection with a 270x device would be considered high priority by the scheduler and could utilize 40 percent of the CPU of a Model 145 effectively doing no useful work. Two such machines working on separate lines were

found to dominate a system and severely impact the response of the system for terminal users.

A System/7 virtual machine was found to contain both user and supervisory program bugs that could severely affect the response of the host Model 145 VM/370 system. Specifically, when normal data transmission between the VM/370 virtual machine and the remote System/7 had been proceeding smoothly, the System/7 was suddenly powered down, and the user program in the virtual machine continued to request further data transfers. The resulting execution loop containing a start I/O to the high-speed transmission line caused domination of Q1 activity (as in the previous example) and consequently intolerable response conditions for other users of the VM/370 system. Later, the high-speed line was erroneously detached from the virtual machine while the program was still running, and subsequent start I/Os were issued against a nonexistent virtual device. The supervisory program was not set up to handle the situation and repeated the operation in a tight loop. This repetition caused absorbtion of all unused CPU cycles and unacceptable loading of the system. Both problems, once diagnosed, were easily fixed with minor program changes.

ACKNOWLEDGMENTS

The author wishes to acknowledge the invaluable guidance provided by Walter J. Doherty throughout the development period of the tools and techniques described in this paper. Acknowledgments are also due to C. Richard Attanasio for CP Tracer, a forerunner to VM/Monitor, to William H. Tetzlaff, for assistance in the conversion of SGP for use under VM/370 and developement of the reduction program library, and to the VM/370 Technical Support Group for improvements made to the control program components of the measurement package as available under VM/370.

CITED REFERENCES AND FOOTNOTES

1. P. H. Callaway, J. P. Considine, and C. H. Thompson, "Uses of virtual storage systems in a scientific environment," *IBM Systems Journal* **11,** No. 3, 200–218 (1972).
2. W. H. Tetzlaff, J. A. Cooperman, and H. W. Lynch, "SGP: An effective use of performance and usage data," *Computer* **5,** No. 5 (1972).
3. Apparent increase in execution time due to sharing of system resources with other users via the mechanisms of scheduling and time slicing.
4. Private correspondence with W. J. Doherty, Thomas J. Watson Research Center, Yorktown Heights, New York.
5. J. H. Griesmer and R. D. Jenks, *The Scratchpad System*, Research Report RC 3925, IBM Corporation, Thomas J. Watson Research Center, Yorktown Heights, New York.
6. *VM/370 Command Language Guide for General Users*, Form No. GC20-1804-3 for Release 2 PLC 13, IBM Corporation, Data Processing Division, White Plains, New York.
7. *VM/370 Operator's Guide*, Form No. GC20-1806-4 for Release 2 PLC 13, IBM Corporation, Data Processing Division, White Plains, New York.

8. Y. Bard, *Performance Analysis of VM/370 Systems*, Cambridge Scientific Center Report G320-2102, IBM Corporation, Cambridge, Massachusetts (October 1974).
9. *VM/370 System Programmer's Guide*, Form No. GC20-1807-3 for Release 2 PLC 13, IBM Corporation, Data Processing Division, White Plains, New York.
10. *IBM System/370 Principles of Operation*, Form No. GA22-7000, IBM Corporation, Data Processing Division, White Plains, New York.
11. *VM/SGP Program Description and Operations Manual*, Form No. SH20-1550, IBM Corporation, Data Processing Division, White Plains, New York. Plotting facility is not available with the VM/SGP IUP.
12. S. J. Boies, *User Behavior on an Interactive Computer System*, Research Report RC 4169, IBM Corporation, Thomas J. Watson Research Center, Yorktown Heights, New York (August 23, 1972).
13. W. S. Hobgood, "Evaluation of an interactive-batch system network," *IBM Systems Journal* **11**, No. 1, 2–15 (1972).

Reprint Form No. G321-5008