VM/370, Attached Processor, and multiprocessor performance study

by W. H. Tetzlaff
W. M. Buco

This paper discusses performance studies of Attached Processors, multiprocessors, and VM/370. A methodology for evaluating performance is discussed. Performance improvements are explained and evaluated. These studies played a role in a new option to the VM/System Product control program that is called the High Performance Option (HPO).

In September of 1980, a system at the IBM Thomas J. Watson Research Center that originally had run on a System/370 Model 168 Attached Processor was upgraded to run on a System/370 Model 3033 multiprocessor system. It was immediately clear that the multiprocessor support had changed the behavior of the system, even on an Attached Processor configuration. At the same time, an analysis and improvement project for Release 6 with the VM/370 System Product (hereafter VM/SP) system was started. See Reference 1 for more information on the computing environment at the Research Center.

This paper discusses the Attached Processor support, what was learned about its operational characteristics, the changes made to the software, and the performance improvements realized. The multiprocessor support and its operational characteristics are also discussed. Changes in software are evaluated.

Following the performance studies reported in this paper, a new option to the VM/SP control program, called the High Performance Option (HPO), was announced. This option has a number of features that affect the performance on the Attached Processor, multiprocessor, and diadic processors. The HPO system incorporates improvements to the paging rate feedback, changes to page migration, and an increase in the number of free storage subpools. The need to keep two copies of shared pages and to examine them for change has been eliminated in HPO Release 1 by making use of a segment protect feature. The free storage algorithms have been further refined[2] in HPO Release 2. HPO Release 3.4 contains algorithm changes to improve cache performance in general, and cache performance on diadic processors in particular.

Measurement tools

At the Research Center a program called VM/Monitor[3] collects data during the first shift for two VM/370 systems. VM/Monitor has long been the primary VM/370 data collection tool here.[4] VM/Monitor collects system performance and resource utilization data by means of sampling and trace techniques, and writes the data collected on tape or (under VM/370 Release 5) to a spool file. Some of the captured events—terminal input, for example—are caused by

activities of system users. Other events correspond to the use of such resources as individual Direct Access Storage Device (DASD) accesses. Still other events, such as moving a user from one scheduling queue to another, are caused when scheduling decisions are made. Sampling is initiated by the expiration of an interval of time. The time-driven events

A special data collector was created to capture response-time data.

are used to cause information about each individual user to be recorded, as well as to cause information about DASD and tape device utilization to be written periodically.

VM/Monitor data are reduced by a special data reduction program written at the Research Center and known as the Generalized Reduction of Information program (GRIN).[4] GRIN is a program generator that has been designed to increase the productivity and effectiveness of performance analysts by enabling them to spend more time on analysis and less time on either defining or writing data reduction programs. During the investigation of the performance of VM/370, GRIN was invaluable because of its ability to create new reports very quickly. Many throwaway reports were created, while others evolved into standard Attached Processor performance reports.

A special data collector was created to capture response-time data. Basic data are created within the operating system. The collector obtained the data and recorded them in a data base during all times that the system was running. This monitor allowed us to collect response-time data without running the VM/Monitor SCHEDULE class of data.

VM/370 accounting data were also used in order to measure total accountable CPU time before and after the changes.

VM/370 Attached Processor support

VM/370 Release 4 introduced support for Attached Processors for the System/370 Models 158 and 168. In this type of system, two processors reference the same main memory and execute a shared operating system. This system differs from a more classical multiprocessor in that the Input/Output (I/O) devices are accessed only from the main processor. Reference 5 contains a more detailed explanation of the Attached Processor support.

Lock management. Prior to the introduction of Attached Processor support in VM/370, it was not necessary to have any locks because the structure of the control program forced all necessary serialization. The Attached Processor support introduced several locks to protect the integrity of system tables at times when both processors were executing system code.

Two types of lock usage were introduced, named *spin* and *suspend*. If it is possible to delay a particular process until the lock becomes available, the state vector is saved and the process is suspended. If suspension is not possible, the processor is placed in a loop, testing for the availability of the lock. This is the spin condition; it takes place in a lock manager that also counts and times the spin conditions for each lock. Lock-spin time is shown in Figure 1 as a percentage of elapsed time waiting for a lock for various ranges of active system users.

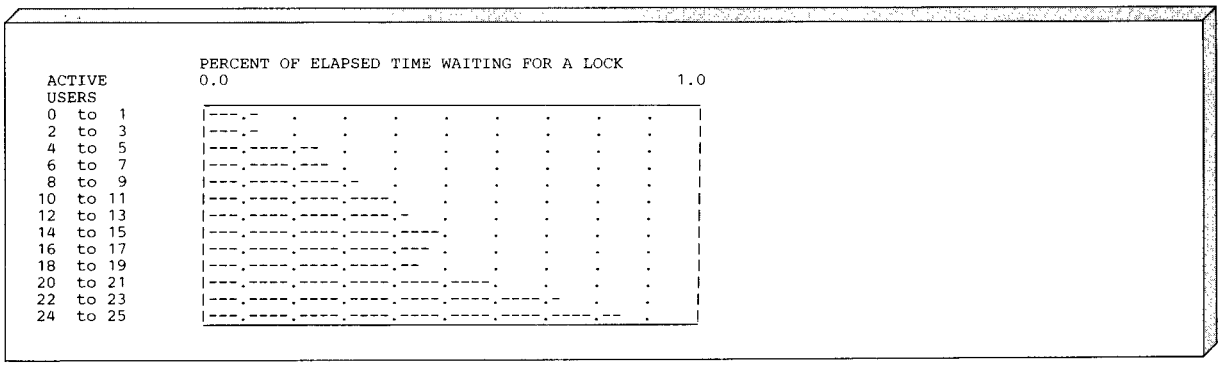
Lock spin time was typically found to be about 0.5 percent of elapsed time on VM/370 Release 5. The dominant lock on the system was the SYSTEM lock. The SYSTEM lock is really a general lock for activity that is not covered by more specific locks. The linear relationship with the multiprogramming level can be observed in Figure 1.

If a lock is needed but unavailable, it may be possible to defer the processes that require the lock. The action of deferring processes happens from 200 to 400 times per second during normal load, indicating that this is an often-used feature. Like the lock-spin time, the defer rate increased linearly with multiprogramming level.

I/O and paging. In an Attached Processor configuration the attached CPU does not have access to any I/O channels. Thus, I/O operations requested by a user on the Attached Processor must be executed on the main processor. Similarly, paging, like I/O, must be initiated on the main processor.

In VM/370, shared pages may be modified by a user. When this happens, the modified page is given to the user, and a new unmodified page is made avail-

Figure 1 VM/370 Release 5 lock-spin time



able to other users. This feature is supported by a subroutine that is called by the dispatcher and is called when the dispatcher detects that it is about to dispatch a different user. The subroutine inspects all shared pages to determine whether the hardware had turned on the change page flag to indicate that the page had been changed by the last user. The shared-page architecture requires that all shared pages be inspected each time the dispatcher switches to a different user.

The performance improvement study determined, through a combination of static instruction counting and dynamic path counting, that substantial CPU time was spent examining the page tables looking for shared pages that had been changed. One way to control this would be to try to limit the number of switches to a different user.

Shared pages (in CMS, for example) cause an additional complexity on Attached Processor systems. It is not possible to have one set of shared pages because, when a shared page is modified, there is no record of which processor modified it. Thus VM/370 maintains two sets of shared pages, one for each processor. On a uniprocessor, it is not unusual to have several hundred shared pages in main memory simultaneously. The necessity of keeping two sets of shared pages doubles the number of pages in main memory at the same time. We found that there was almost always an exact match between the particular pages that each processor had in main memory. The doubling of the pages caused about 400 page frames to be used for this purpose. The loss of 200 pages can be significant in a system that is storage-constrained.

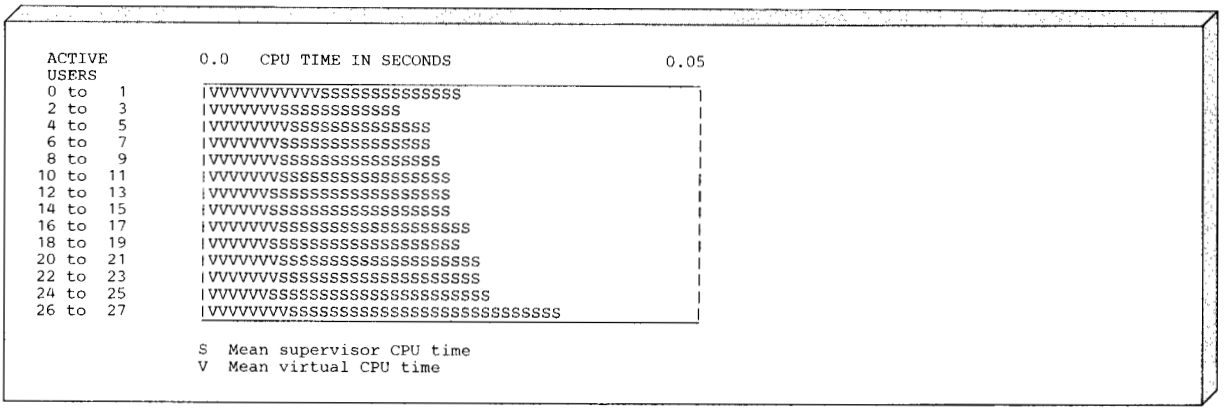
When a processor requires a shared page, it initiates a page-in. However, the user might later be dispatched on the processor that does not have the page. That does not cause a double page-in because the system detects that the first processor now has the page in memory and, therefore, does a memory-to-memory copy.

Dispatching. The Attached Processor dispatcher works in basically the same way that it did before Attached Processor support was added. Changes deal primarily with the resumption of suspended processes and with the ability to force processing onto a particular CPU. The dispatcher uses the same priority list for finding a dispatchable user for either processor. Suspended processes are dispatched ahead of the normal priority dispatching order.

The study data showed the dispatch rate to be somewhat higher than we had previously found for the uniprocessor system. The rate appeared to be quite high, with 1000 to 1600 trips through the dispatcher per second, but somewhat lower in the context of the I/O and paging rates. A page-in or I/O operation normally involves one dispatch when the requesting machine begins to wait for completion of the operation. Later, when the operation has completed, a second dispatch is required to begin processing again. The dispatch rate is about two times the page-in rate plus the I/O rate.

On a uniprocessor system, VM/370 uses preemptive dispatching, which means that the highest-priority user always runs. This is accomplished by examining the dispatching list each time a user becomes ready to run as a result of an I/O completion. When two

Figure 2 VM/370 Release 5 interactive CPU usage



processors are used, preemption does not take effect on the Attached Processor. The I/O completion causes an interrupt on the system with the device. The user who is then ready to run is selected by the dispatcher if he is the highest-priority ready user. If he is not the highest, he will not be run. The user being run on the Attached Processor may be of lower priority, but he is not preempted on the Attached Processor. Thus, Attached Processor dispatching does not conform to strict preemptive dispatching. This nonpreemptive dispatching causes some biases in the treatment of users, so that low-priority, computation-bound users may obtain a great deal of CPU time.

One of the measures of the differences between work run on the main CPU and the Attached Processing Unit (APU) is the ratio of *virtual time* (i.e., time running in problem program state) to page-in operations. We found that, on Attached Processor systems, the Attached Processor used 2.5 times as much virtual time between page-ins. Thus, the Attached Processor benefits primarily long-running users. These users avoid preemption by not voluntarily giving up control to the dispatcher.

The data showed that system CPU time was considerably higher when an Attached Processor system was being run. Data from scheduler traces showed that the greatest increase in supervisor time was associated with transactions because these transactions make greater use of supervisor services and do the bulk of the paging on the system. The supervisor time per transaction doubled on an Attached Processor system as compared with a uniprocessor system. This was apparently due to longer paths associated with paging and other services. The data used to produce Figure 2 also showed that supervisor time per interactive transaction increased linearly with the number of active users, whereas virtual time remained constant. Problem state time per transaction stayed essentially the same.

Multiprogramming level as a measure of load. Callaway[6] demonstrated the usefulness of using multiprogramming level as a measure of contention on a VM/370 system. He showed the relationship between virtual time and total CPU time under increasing load. Our data are summarized in Figure 3. Using the same type of graphs, we have shown that the multiprocessor and Attached Processor systems studied become I/O-saturated before they become CPU-saturated. By this we mean that increasing the multiprogramming level does not increase the number of I/O operations. This has not been the case for our uniprocessor systems because they become CPU-saturated first.

Responses are normally categorized as *interactive* (or Q1 transactions, because they complete while in scheduling queue number 1) or *long-running* (called Q2 transactions because they complete while in scheduling queue number 2). Transactions are initially placed in Q1. If they have not completed after using a specific quantity of CPU time, they are moved to Q2. The graph in Figure 4 shows the Q1 response time relationship to multiprogramming level, which is useful in evaluating the ability of the system to maintain service during overload. The events displayed by the graphs have proved to be repeatable and have proved to be characteristic of a given system. It is possible to observe changes in the shape

Figure 3 Virtual (V) and total (T) CPU time as a function of active users

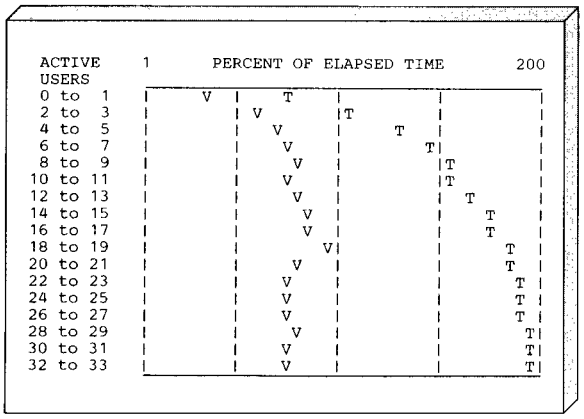


Figure 4 Interactive response time

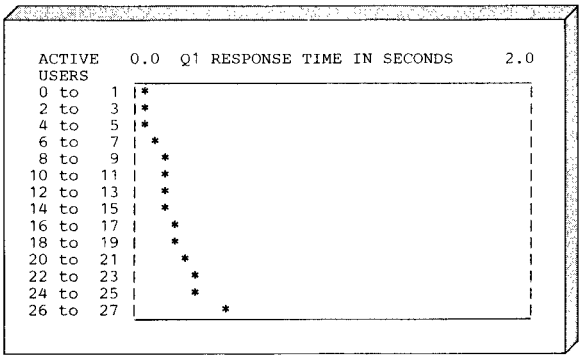
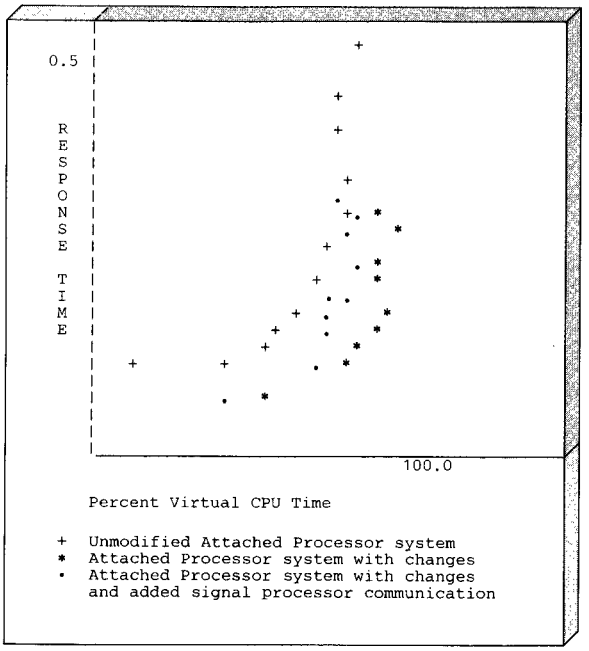


Figure 5 Response time throughput tradeoff



of the response-time-to-multiprogramming-level relationship curve as the system improvements are made.

The graph that was found most useful was a combination of the percent virtual CPU time data (from Figure 3) and the Q1 response time data (from Figure 4). The plot of these measurements in Figure 5 shows the tradeoff between response time and throughput. Each point on the graph in Figure 5 represents the average response time and the average virtual CPU utilization at a particular multiprogramming level. In general, at low load the response time is very good, and the CPU is relatively underloaded. At heavy load, the response time is higher, and the virtual CPU time is as high as it can be. At overload, the response time is very high, and the virtual CPU time is lower due to the increased need for supervisor CPU time to manage the system. Performance improvements are easily seen by shifts in the curve toward lower response times and higher virtual CPU time.

Changes in Attached Processor support

Our observations of Attached Processor performance indicate that the additional processor gives good improvement in *expansion factors* for noninteractive work, particularly that which is CPU-bound. The expansion factor is a measure of increased elapsed time to complete a transaction due to contention for resources. The expansion factor is the actual elapsed time divided by the elapsed time that would have been required without contention between users. (Reference 4 contains a discussion of the calculation of expansion factors.) We felt there was potential for greater throughput relative to a uniprocessor as measured by virtual CPU time. Thus our goals in modifying the system have been to reduce Q1 response time and increase virtual CPU time.

Paging rate. We observed that the INDICATE command did not display paging rates that were consistent with the rates derived from VM/Monitor data. (The INDICATE command may be used to display the current paging rate of the system at a terminal.) We found that for Attached Processor systems the INDI-

CATE command displayed the total paging rate divided by two. Thus it displayed the average rate per processor. The same calculation was also used in the systems control algorithms, which prevented the system from recognizing that the actual paging rate was too high and taking corrective action. Division by two was removed to allow proper feedback.

Dispatcher. Another change to the system relates to a preemption effect that gave Q2 users increased preference on Attached Processor systems. The dispatcher was modified to select the highest-priority Q1 user, even though a higher-priority Q2 user might be ready to run. This change might be more appropriately made by changing the scheduler so that when the system is an Attached Processor, Q1 priorities are made higher than Q2 priorities.

For several reasons, the dispatcher has been modified to redispatch an interrupted user if he was in Q1. This modification gives Q1 users added preference and reduces the number of dispatcher switches so that shared pages need not be inspected. This reduces supervisor CPU time and increases the probability of the processor's finding needed data in the cache (i.e., high-speed storage), thereby allowing instructions to be executed faster.

Time quantum. If a CPU-bound user (who may be in a loop as a result of a bug) is dispatched on the Attached Processor, he may run a long time because of the lack of preemption. The mechanism for stopping such a user is the time quantum. The end of a time quantum creates an interrupt, which gives the dispatcher the opportunity to discover whether a higher-priority user has become ready to run. When a user runs to the end of a time quantum he is marked as *CPU-bound* if he did not do any I/O during the time quantum. Based on this definition of CPU-bound, that user's next quantum will be four times as long as the previous one. In practice, even a low-priority user in a loop may be dispatched often enough to use as much as half of the power of the Attached Processor. In order to lessen the impact of such a user, we eliminated the multiplication of the first time quantum by four for CPU-bound users.

Some experiments were done with code that used the System/370 Signal Processor (SIGP) instructions to cause more communication between the two processors. This instruction allows one processor to interrupt the other. The code was intended to improve the throughput of low-multiprogramming-level Attached Processor systems. The systems signal one another when they have made work available that the other processor should do now. This creates more preemption.

Evaluation after changes

Evaluation of individual changes was done by comparing the response time-throughput tradeoff data as shown in Figure 5. There are several reasons why it

It was necessary to improve service quickly and without stopping the flow of improvements.

was not possible to do complete evaluations of each change individually, all of which relate to the fact that the changes were being made on a system at the same time it was providing service to a large number of users. Thus, it was necessary to improve service quickly and without stopping the flow of improvements. At the same time, data would have to have been collected over a period of many days in order to fully evaluate each change. It would also have been necessary to remove some of the previous changes in order to evaluate their interactions. Otherwise, there might have been a regression in service that would not be acceptable to our users or our system management philosophy.

Figure 5 shows the response time versus throughput profiles for three typical usage days. The shape of the curve moved down and to the right as a result of system modifications. The day with the added signal communications falls between the other two curves. Additional signal communication was not a useful change for the system. (We ran with that change for three days to confirm the reduced performance and then removed it from the system.)

The signal communication modifications were characterized by high supervisor CPU time, high lock-spin time (five times higher than without the modifications), and poor interactive performance. It seems that the Attached Processor architecture has a

Table 1 Mean Q1 response time analysis for two months

Time	November Mean Response	February Mean Response	Improvement	Percent Improvement
9 AM	0.428	0.343	0.085	20
10 AM	0.481	0.346	0.135	28
11 AM	0.357	0.277	0.080	22
12 PM	0.221	0.217	0.004	2
1 PM	0.480	0.303	0.177	37
2 PM	0.474	0.357	0.117	25
3 PM	0.605	0.390	0.215	36
4 PM	0.617	0.405	0.212	34
mean	0.458	0.330	0.128	28

natural advantage in that it tends to place the need for locks on the main processor, thus reducing the contention for locks. The attempt at balancing supervisor CPU time between the two processors was ill advised because it also maximized lock contention.

Table 1 gives an overall evaluation of the changes by comparing mean Q1 response time for each hour of the day for each of two months. November of 1979 was the last month before the scheduling changes were tried. February of 1980 was the first month after the scheduling changes had been completed. Thus Table 1 contrasts the hourly response times for these months. The fact that each of the eight hours has improved response time shows the statistical validity of the conclusion that there was an improvement. A 28 percent mean reduction in response time resulted, despite an increasing workload, and it was accompanied by an increased throughput.

VM/System Product experiences

The multiprocessor support provided by VM/System Product (VM/SP) allows both processors to access data channels. If a device is to be accessible from both processors it must have the same address on each processor. It is not required that all addresses be accessible from both processors. Processing for a user's I/O operation begins on the processor that is executing the user's program. When processing has reached the point at which a Start I/O operation (SIO) is to be done, it may then be determined that a path to the requested device is not actually available. If so, the operation is queued for the other processor.

In VM/SP two more locks were added to the system: one for I/O and the other for real memory. This was done in order to reduce the use of the system lock.

During the first shift of the first day of operation the system experienced severe performance problems. Paging bottlenecks were the most noticeable difference from the VM/370 Release 5 system that had been running. Measurements indicated that the selection of old pages for drum-to-disk migration was not as effective as VM/370. The characteristics of the user virtual memory requirements were confirmed to be the same, and the size of the drum paging space was the same. The decreased effectiveness of the drum-to-disk migration was probably due to changed page reference patterns in CMS. Previously, drum migration had been able to ensure that 70 to 80 percent of the pages of the active users were on drum. Under VM/SP the percentage dropped below fifty. This drop in page availability had the predicted increase in page-in time accompanied by increased Q1 response time.[3]

By having a multiprocessing system, the conditions for preemption change but equity among programs is not resolved. Our system was not fully symmetric. It was deliberately configured to provide paths to all devices from the main CPU, and redundant DASD paths were the only ones available from the second processor. This ensured the continued ability to run VM/370 Release 5 on the hardware. The virtual time per page-in was observed as 1.5 times higher on the processor with fewer I/O devices than on the other processor. The multiprocessor architecture is not a solution to the preemption problem because both CPUs suffer from some nonpreemptive dispatching.

Several days of running with a completely asymmetric configuration were done to determine whether the additional paths were a help or a hindrance. The result was that it was not possible to measure any difference in response time or throughput.

Page migration. Page migration is a facility that moves unreferenced pages from high-speed paging drums to larger but lower-speed paging disks. Page migration changes were implemented to cause the drum-to-disk migration to operate more nearly continuously, and to improve the approximation to the Least-Recently-Used (LRU) algorithm. This was accomplished primarily by invoking the page migration routine more often (once per minute instead of every ten minutes), and compensating by moving fewer pages.

The changes to the migration algorithms had a very positive effect on the use of drums for paging.

The changes to the migration algorithms had a very positive effect on the use of drums for paging. The drums became used typically for about 80 percent of the pages of the active users. The shift of paging toward the drums caused the average page-in time to be halved. Because page-in time is the largest component of the Q1 response time, the shift of paging toward drums reduced the response time by about one third.

Free storage manager. Even after the page migration changes had been installed in the system, there were still two striking differences between the VM/370 Release 5 and the VM/SP system. First was lock-spin time, which was about five times higher than that for VM/370. Second was supervisor CPU time, which was also higher. The particular lock that was associated with most of the spin time was the free storage management lock (DMKFREE). This lock had very little spin activity on VM/370 Release 5, but on VM/SP it was the dominant lock on the system. Our previous experience had shown that lock-spin time is a very sensitive measure of lock-hold time. The lock-spin probability is proportional to the product of the lock-hold probabilities of the two processors. Thus, a small change in lock-spin time indicates a much larger increase in lock-hold time. This information pointed to the free storage manager.

The free storage manager is used to manage available storage and make blocks available for control blocks and buffers. Storage is initially accounted for by chaining blocks together, in what is called the *main chain.* The main chain is ordered by storage address so that adjacent free blocks can be recognized and merged into one block. VM/370 makes high use of data areas that require less than 30 double words of storage. In order to make small blocks available quickly (without searching the main chain), a system of subpools was created. There are ten subpools that contain blocks of 3, 6, 9, etc. double words of storage. A request for storage may be filled very quickly by rounding up to the next multiple of three double words and taking an available block of that size. Available storage is placed on the subpools, as a result of freeing a previously used block. Storage blocks in a particular subpool are chained together and managed in Last-In-First-Out (LIFO) order.

Dynamic and static measurements determined that requests calling for storage from the main chain required three orders of magnitude more CPU time than the subpools to obtain a storage block. Thus, a small increase in the use of the main chain could have a dramatic effect on the time spent in the free storage manager.

The most obvious change in the pattern of storage use in VM/SP was the display terminal input area increase. The input area buffer for display terminals was increased from 28 to 31 double words. Thus, these requests were moved out of the subpool management.

An additional subpool was created so that there would be eleven pools of up to 33 double words. This brought the input buffer back into subpool management, which provided a considerable improvement. Storage management time was reduced from 10–15 percent of elapsed time to 5–10 percent of elapsed time.

The use of eleven subpools has made some reduction in the lock-spin time, thereby allowing the system to run at less than two percent lock spin most of the time, as shown in Table 1. This is still four times the lock-spin time of VM/370 Release 5, and the difference is almost fully accounted for by the DMKFREE lock, which is the dominant lock on the system.

Table 2 shows response times for the various systems studied. With VM/SP, we observed a 34 percent increase in Q1 response time. The page migration

changes and eleven subpool codes have produced a system that is about equivalent to the previous VM/370 Release 5 system.

Epilogue

The purpose of this epilogue is to describe some of the additions and changes made to the performance algorithms by the VM/SP High Performance Option (HPO) product. This discussion is not meant as a guide or tutorial on the VM/SP High Performance Option, but rather as an overview of the significant differences between current products and the previous discussion in this paper.

Shared pages. When run on certain processors (e.g., IBM 3081, 3083, 4381, and partitioned 3084), HPO uses the Segment Protect feature to protect the pages of shared segments. This reduces the time spent scanning the pages to inspect the change page flag, because the hardware now prevents any change. On an IBM 308X run as a Dyadic Processor, there is an additional benefit in storage, since only one copy of a page is needed.

Because the hardware is preventing any changes to the pages, including any changes to the protect keys, the Set Storage Key (SSK) instruction can now be assisted by Virtual Machine Assist (VMA). This results in a faster execution of these instructions by CMS on processors with VMA and Segment Protect hardware (e.g., IBM 308X and 4341 Processors).

Dispatching. The dispatching algorithms have been changed to improve performance. On AP, MP, or dyadic processors generated for MP or AP modes of operation, two dispatch lists are now maintained. When a virtual machine begins a new transaction, it tends to run on the same processor for the entire transaction. This *soft affinity* has been provided to allow better reuse of the data in each processor's real storage cache. This is especially helpful for redispatching a virtual machine after a short delay, such as a page fault.

Signals between processors. HPO has reduced its need for Signal Processor (SIGP) instructions, especially the SIGP WAKEUP, compared to VM/SP. These SIGP instructions had been introduced to make the system more responsive to work when one processor is busy doing something, the other processor is idle, and an interrupt occurs that creates work to be done. Signal Processor instructions were introduced so that the busy processor could be sure that the idle processor started to work on the newly eligible work. This created a problem in perception, since a lightly loaded system appeared to have less remaining capacity than in fact it had. *Low Utilization Effects* are characterized as extra overhead spent when the system is not fully loaded. As the system becomes more heavily loaded, the probability of having one processor busy and one idle is decreased, so that the overhead of interprocessor communications is reduced.

Table 2 Evaluation of VM/SP changes

System	Days Measured	Q1 Response Time
VM/370 Release 5 with AP changes	10	0.216
VM/SP with AP changes	9	0.289
VM/SP with AP changes and page migration changes	10	0.263
VM/SP with AP changes, page migration changes, and 11 subpools	6	0.198

In HPO, the system now has a new way of handling this condition. The former way was to have the busy processor do the following: Receive the interrupt, determine that the other processor is idle, queue the work, signal via SIGP for the other processor to WAKEUP, and then go back to its useful work.

HPO now has the idle processor scan its own queue for available work. If the processor has no work, it looks to see whether the other (busy) processor has two or more units of work. If the busy processor has a queue, the idle processor steals the second entry on the queue. This reduces the time spent processing the interrupt, yet allows the "two processors and one unit of work" condition to be handled efficiently. This change is called *Active Wait State.*

Page migration. The page migration changes described in this paper were integrated into HPO.

Free storage management changes. As a result of the research described in Reference 2, a split set of subpools was created. There are now subpools of 2, 4, . . . , 30, 32 double words and 64, 96, 128, . . . , 992, 1024 double words. This allows any request up to two 4096-byte pages to be satisfied from a subpool. The latest release of HPO has extended these storage management changes in several additional ways:

- Each processor has its own set of subpools. This reduces interference between the caches of the two processors in a dyadic complex.

- Each subpool has a separate lock. Since only one processor normally goes after storage in each subpool, the locking procedure rarely finds the lock held by the other processor. There is still a lock for the main chain of storage blocks that are not on a subpool, but the contention for it has been greatly reduced.
- Each processor also has available another subpool of 128-byte cache-aligned entries, called *Prime Storage*. This area is requested (with a special request) for control blocks that are characterized by short life and high activity. Having the prime storage areas cache-aligned ensures that no control block will span cache lines, and no cache line will have two control blocks, with one in use by each processor.
- There is another area for larger blocks (128–768 double words) that is also cache-aligned for control blocks that are characterized by high activity and long life. The only control block currently placed in this area is the VMBLOK, but the interface is more general and may be used for additional blocks in future releases.

Concluding remarks

Attached Processor and multiprocessor systems often have characteristics that differ from those of uniprocessor systems from which they are derived. For example, a priority dispatcher that has preemption on a uniprocessor may lose some of its control of CPU allocation through a lack of preemption across processors in the complex. A lack of preemption in dispatching is better compensated for in scheduling and dispatching than by creating additional communication among processors.

The relationship between lock-spin time and lock-hold time provides an interesting way to measure lock-hold time by measuring lock-spin time. Attached Processor systems have a useful separation of function that unevenly distributes lock usage, thus lowering lock-spin time. An even distribution of lock-hold maximizes lock-spin time.

The graphs that relate throughput in virtual CPU time to interactive response time are very useful. They show very clearly the tradeoff that the system makes between throughput and response time. The shapes of the graphs were shown to be highly repeatable, despite daily fluctuations in load. The graphs provide a way to evaluate changes to the system, despite day-to-day load changes.

Three significant performance improvements designed to reduce response time and to increase throughput have been discussed. The changes we made resulted from our study of multiprocessor and Attached Processor systems, but benefits were also created for uniprocessor systems. The dispatcher changes reduce task-switch overhead on any system using shared pages, improve Q1 response, possibly improve cache hit ratios, and improve the Q2 response on Attached Processor/multiprocessor systems. The page migration changes should improve migration on any system that uses drum-to-disk migration. The storage management changes reduce supervisor CPU time for any VM/SP system.

Acknowledgments

The study of Attached Processor performance was initiated when C. Stephenson and G. Waldbaum proved that Q1 response time was longer on the Attached Processor system at the Thomas J. Watson Research Center than it was on the uniprocessor. L. Junker and N. Brenner provided the continuous monitoring of interactive response time on the systems and also the data reduction of historical response-time data. W. Doherty, P. Capek, L. Climenhaga, M. Linehan, and A. Greenberg participated in the process of understanding the system and proposing changes. W. J. Doherty, H. Serenson, and R. P. Kelisky provided management support. The authors thank D. R. Patterson for his critical review of the manuscript.

Cited references

1. W. J. Doherty and R. P. Kelisky, "Managing VM/CMS systems for user effectiveness," *IBM Systems Journal* **18**, No. 1, 143–163 (1979).
2. G. Bozman, W. Buco, T. P. Daly, and W. H. Tetzlaff, "Analysis of free-storage algorithms—revisited," *IBM Systems Journal* **23**, No. 1, 44–64 (1984).
3. *VM/370 System Programmer's Guide*, GC20-1807; available through IBM branch offices.
4. W. Tetzlaff, "State sampling of interactive VM/370 users," *IBM Systems Journal* **18**, No. 1, 164–180 (1979).
5. L. H. Holley, R. P. Parmelee, C. A. Salisbury, and D. N. Saul, "VM/370 asymmetric multiprocessing," *IBM Systems Journal* **18**, No. 1, 47–70 (1979).
6. P. Callaway, "VM/370 performance tools," *IBM Systems Journal* **14**, No. 2, 134–160 (1975).

Reprint Order No. G321-5231.

William H. Tetzlaff *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Mr. Tetzlaff joined the Service Bureau Corporation in

1966. He joined the Research Division of IBM in 1969 and has done research in the areas of information retrieval and system performance. He published several papers on that research, and received an IBM Outstanding Contribution Award for his work on system performance. He recently completed a temporary assignment as a member of the Technical Planning Staff of the Research Division. Mr. Tetzlaff studied engineering sciences at Northwestern University and is a graduate of the IBM Systems Research Institute. He is currently manager of VM Analysis and Restructure in the Computer Sciences Department of the Research Division.

William M. Buco *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Mr. Buco is currently Technical Assistant to the Director of Computing Systems at the Research Center. From 1970 to 1974 he worked at the IBM Cambridge Scientific Center on prototype versions of Discontiguous Shared Segments and schedular extensions for VM/370. He worked at the Research Center from 1974 to 1977 as a systems programmer improving the performance and reliability of VM/370, and from 1977 to 1983 he managed the VM/370 systems programming project. In 1974 Mr. Buco received a B.A. in mathematics from Northeastern University and in 1977 an M.A. in computer science from Columbia University.