# Multiple operating systems on one processor complex

by T. L. Borden
J. P. Hennessy
J. W. Rymarczyk

***As large computing systems continue to grow in capacity and to offer improved price/performance, there is an increasing requirement to consolidate systems onto one processor complex. This paper describes the reasons why users need to run multiple operating systems today, provides a brief history of IBM's partitioning products, and introduces the Processor Resource/Systems Manager™, a machine feature on the IBM 3090 Model E and ES/3090™ Model S processors that provides users with a flexible and efficient capability to run multiple operating systems on a single processor complex.***

The development of powerful, high-availability computing systems has led to a diversity of computer applications, from transaction processing to engineering design simulation, each with its own unique set of requirements. These diverse requirements have resulted in the development of many different programming languages, application programs, and even operating systems, each with its own strengths and weaknesses.

One might think that a single operating system could be designed to satisfy the requirements of all environments. In practice, the system design trade-offs that must be made, together with compatibility constraints, imply that no single design could satisfy the full range of requirements. Instead, several designs have evolved, each addressing the needs of a large segment of the marketplace. For example, on its high-end processors, IBM offers a variety of operating systems, including Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA™), Multiple Virtual Storage/Extended Architecture (MVS/XA™), MVS/370, Virtual Machine/Extended Architecture (VM/XA), Virtual Machine/370 (VM/370), and Transaction Processing Facility (TPF), each with its own customer set.

Following are some of the main reasons why multiple operating systems may need to coexist even within a single establishment:

- *Diverse workloads*—Many large establishments today must satisfy the computing requirements of several distinct groups of end users. For example, a large airline might need a very responsive reservation system, an aircraft maintenance and parts database system, and a general-purpose interactive and batch system for payroll, forecasting, planning, etc. It is not unusual to find that at least two different operating systems are required, with appropriate subsystems and software products, to provide the full set of required functions.
- *Test and development*—Many companies are critically dependent on the high availability of their computing systems. As a result, it is necessary to

allow for nondisruptive enhancements to the system software for each production system. The software changes can take many forms, such as a new release of the operating system itself, a new release of some key application, the addition of new products or functions, or simply the application of software maintenance. In any case, to minimize the possibility of an outage in the production system, it is essential that changes to critical software be tested in a system context prior to being adopted in the production environment.

Many large businesses also do a significant amount of system software development. By its very nature, system software debugging implies repeated system failures, and thus these establishments need separate production and development systems. Moreover, for businesses that require 24-hour availability of their production systems, it is impossible to perform the software development on the production system on an off-shift basis.

- *Migration*—It is sometimes necessary for a business to convert from one operating system to another. For example, a company application may need to migrate from MVS/370 to MVS/XA or MVS/ESA in order to obtain increased virtual storage capacity. Another typical migration could be a Disk Operating System/Virtual Storage Extended (DOS/VSE) customer who has outgrown an IBM 4381 system and needs to migrate an application to an IBM 3090 system running MVS.

- *Constraints within a single operating system*—As workloads grow, some applications may encounter growth constraints within a single operating system. A common example can be found with MVS applications and subsystems that were designed for the 24-bit addressing environment of MVS/370 and are being expanded to use the 31-bit addressing of MVS/ESA. Even though the MVS/ESA system provides the expanded addressing capability, it may be difficult and time-consuming to convert the applications to use more than 16 megabytes of virtual storage. The resulting virtual storage constraint may force some customers to run multiple systems, each with a separate database.

- *Backup and recovery*—For applications that require continuous availability, it may be necessary to recover from system-software, as well as hardware, failures. This availability can be had by running two copies of the system, one in production mode and the other as a stand-by backup system. In the event that the production system fails, the backup system can take over the end-user workload, providing it is properly connected to the terminal network and the system database. This switchover can be done manually, or automatically by using software such as IBM's Extended Recovery Facility (XRF).

## Evolution of partitioning

The ability to run multiple operating systems on a single processor complex has existed for more than twenty years. Traditionally, this partitioning capability has been provided in two ways: as software-based partitioning, in which system software (such as VM) creates multiple virtual machines, and as hardware-based partitioning, in which the hardware itself can be subdivided to form multiple, independent computing systems. This section briefly describes the IBM product evolution that has occurred for both types of partitioning.

**Software partitioning.** The first software product to provide a virtual machine capability was the Control Program/67 (CP/67) operating system,[1,2] which ran on the System/360 Model 67 and was first available in 1967. It gave each user a virtual machine in which the single-user Conversational Monitor System (CMS) operating system could be run to provide command processing and information management functions. Since each virtual machine was a replica of the base System/360 hardware architecture, it was also possible to run multiple copies of Operating System/360 (OS/360) in a virtual machine. In fact, even CP/67 itself was run "second-level" in a virtual machine for the purposes of debugging and testing.

CP/67 was highly successful, and in 1972 IBM announced VM/370, a successor product, for the entire System/370 processor family. VM/370 soon became one of the most popular operating systems, providing both excellent interactive computing facilities and the capability to operate "guest" operating systems in virtual machines.

VM/370 introduced the ability to run a single preferred guest in addition to numerous nonpreferred guests. The preferred guest was allocated a contiguous range of main storage beginning at absolute storage location zero, and therefore avoided the performance overhead associated with address relocation and paging. This capability is commonly referred to as Virtual=Real, or V=R, storage allocation.

VM/XA also provided a V=R preferred guest when it was introduced in 1983. In 1987 IBM announced and

delivered the VM/XA SP Multiple Preferred Guests (MPG) facility which uses the new Processor Resource/Systems Manager™ (PR/SM™) hardware feature to provide up to five Virtual=Fixed (V=F) preferred guests in addition to the V=R guest. The design of this product is described in more detail in the sections that follow.

**Hardware partitioning.** Since the introduction of the System/360 Model 67 and Model 65 multiprocessor (MP) systems in 1967, IBM has offered multiprocessor computing systems to provide increased system capacity and availability. To achieve the availability goals, all hardware components of the multiprocessor systems are duplexed so that single hardware failures are unlikely to take down the entire system. In 1973, with the introduction of the System/370 158 MP and 168 MP systems, the duplexed MP hardware design was used to provide an additional capability: physical partitioning.

In a physically partitioned MP system, the hardware facilities of the complex are divided into two sides. Each side is a separate machine that can be operated independently and even powered-off without affecting the other side. Physical partitioning has been a standard feature on all multiprocessor models on the IBM 303X, 308X, and 3090 processor families.

**PR/SM.** PR/SM is an optional feature on the IBM 3090 Model E and ES/3090™ Model S processor families that allows a single processor complex to support the concurrent execution of multiple operating systems. It consists of special hardware and microcode that can be invoked and controlled in either of two ways: directly through the machine console (hardware logical partitioning) or indirectly under software control by the VM/XA SP control program. The remainder of this paper describes these two methods of operation and provides reasons why a user might choose one method over the other.

## Logical partitioning

**Overview.** With the introduction of the PR/SM feature, the 3090E and ES/3090S processor families now offer a hardware partitioning capability that is significantly more flexible than hardware physical partitioning. Logical partitioning (LPAR), a new mode of machine operation, offers users the following advantages over physical partitioning:

1. LPAR is available on all models of the 3090E and ES/3090S processors. Physical partitioning is only available on some of the multiprocessor models (e.g., it is available on the 280, 400, 500, and 600 models and is not available on the 120, 150, 180, 200, and 300 models).
2. LPAR provides up to six partitions on the ES/3090S (four on the 3090E), whereas physical partitioning allows only two partitions.
3. Physical partitioning splits the processor complex into two equal (except for processors on the model 500) partitions. LPAR gives the user the ability to define the granularity of the partitions.

A logical partition is a collection of processor complex resources that, when combined, are capable of running an operating system. The resources which comprise a logical partition include processors, main storage, expanded storage, channel paths, vector facilities, subchannels, and logical control units. A logical partition can be System/370 or Enterprise Systems Architecture/370™ (ESA/370™) mode (System/370 Extended Architecture, or 370-XA, mode for those models that do not provide ESA/370), independent of the mode of any other partition. Partitions operate independently and are isolated from one another as if they were loosely coupled processor complexes; i.e., the only interaction between partitions is via I/O operations (shared direct-access storage device [DASD], channel-to-channel connection, or telecommunications control units). This isolation is accomplished by dedicating a portion of storage (main and expanded) and I/O elements (channel paths, subchannels, and logical control units) to a single logical partition. Computational elements (processors and vector facilities) can be either dedicated to a single partition or shared among multiple partitions. Figure 1 shows a conceptual view of a physical processor complex (a Model 600) with four

**A logical partition is a collection of processor complex resources that can run an operating system.**

logical partitions (MVSIMS, MVSPROD, MVSTEST, and VMHPO) defined. Two of the physical processors (0 and 1) are dedicated to the MVSIMS partition; the remaining four physical processors are shared among the other three partitions. A partition is defined as having one or more logical processors. Since the total number of logical processors for all of the partitions can exceed the number of physical processors, LPAR has a dispatcher for assigning a logical processor to use a physical processor at any point in time.

Logical partitioning is a new mode (LPAR) for the IBM 3090E and ES/3090S processor families that is selected at power-on reset (POR) of the processor complex. With the PR/SM feature installed, 3090E and ES/3090S processors have three basic modes—System/370, 370-XA or ESA/370, and a new LPAR mode.

In LPAR mode, main storage and expanded storage are subdivided into contiguous areas with 1-megabyte granularity and allocated to each of the partitions such that each partition appears to have a 0-origin for its storage. All storage addresses used in the instructions or channel program addresses of a partition are relocated by the processors and channels and checked to ensure that they are in the range of physical storage allocated to the partition. Figure 1 shows an example of the allocation of main storage among four partitions. In this example, all storage addresses used by partition MVSPROD must be in the range of 0 to 40 megabytes, and the central processors and channels will automatically relocate the storage accesses to use physical storage locations in the range of 56 to 96 megabytes.

Isolation of I/O activity is achieved by giving each partition its own logical I/O subsystem. Channel paths, subchannels, and logical control units are dedicated to a partition; in Figure 1, each of the four partitions has its own channel path, CHP, (e.g., CHP 14 for partition MVSPROD) to device 3C0, and each partition has a unique subchannel and logical control for device 3C0. Therefore, input/output to device 3C0 behaves as if the four partitions were loosely coupled physical processor complexes. For example, if partition MVSPROD issues a RESERVE instruction to device 3C0, the other partitions will get a "busy" if they attempt to issue any I/O instructions to 3C0. Any I/O instruction issued from MVSPROD for device 3C0 can only use channel path 14, since its subchannel only has channel path 14 available; similarly, partition MVSIMS can access device 3C0 only via channel path 10.

Each partition consists of one or more logical processors which the LPAR dispatcher assigns to physical processors at different points in time; the total number of logical processors defined can exceed the number of physical processors installed. However, for any individual partition the number of logical processors defined for the partition may not exceed the number of available physical processors. Partitions may be either dedicated or shared. Dedicated partitions have exclusive use of physical processors assigned to the partition; shared partitions share use of physical processors assigned to shared partitions under the control of the LPAR dispatcher. For shared partitions, the LPAR dispatcher maintains general-purpose registers, control registers, vector registers, and program status words (PSWs) for each of the logical partitions. In Figure 1, partition MVSIMS is dedicated and has been assigned physical processors 0 and 1 for its use. The other partitions share use of physical processors 2, 3, 4, and 5. It is a user's choice whether a partition is dedicated or shared. A partition that exhibits a steady demand for processing resources and that requires an integral number of processors can achieve the highest throughput when assigned dedicated physical processors. In most situations, however, the processing demands of a partition fluctuate from moment to moment, and greater system throughput can be achieved through the sharing of physical processors.

In summary, an LPAR is a logical machine consisting of a subset of the resources of the physical processor complex and is isolated from all other partitions by the PR/SM hardware and microcode. The only communication available between partitions is via I/O connectivity.

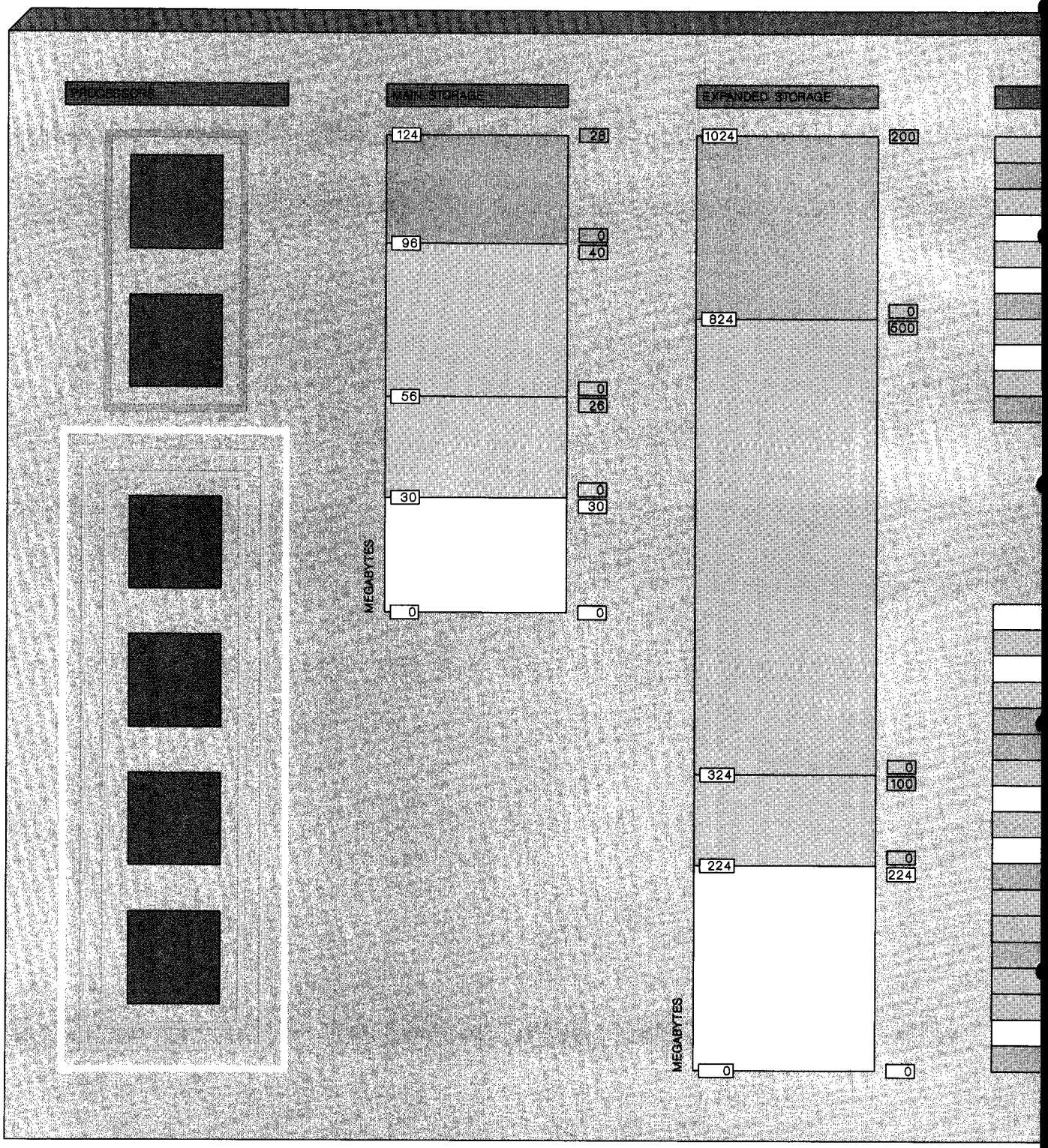
**Partition definition.** In the previous section a logical partition was defined as a user-specified collection of processor complex resources that, when combined, are capable of running an operating system. Creating and using a partition involves two steps:

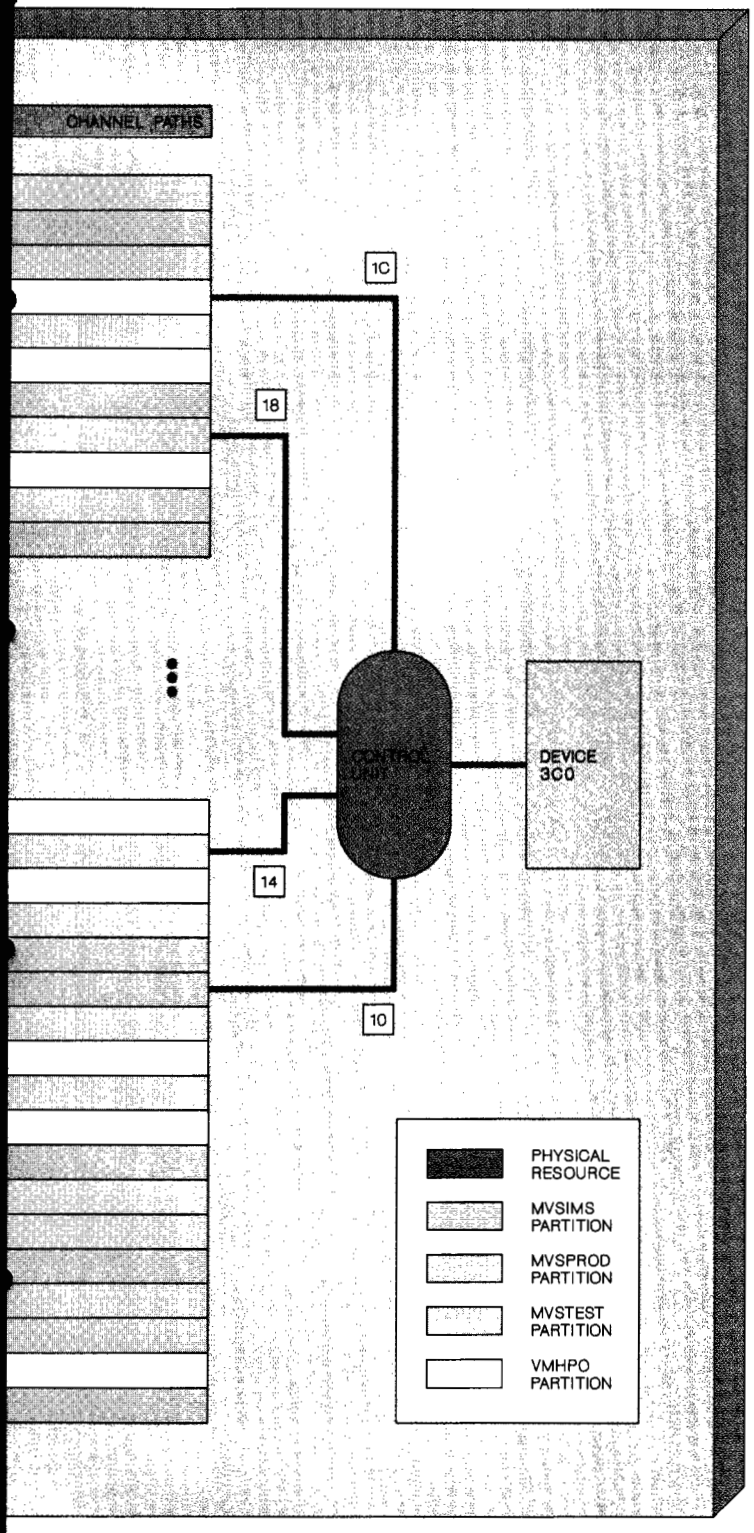
1. Defining the resource requirements of the partition
2. Allocating and initializing the resources of the partition (activating the partition)

In order to define the resources of a partition, a user must specify the following items:

- The names of the partitions to be used
- The I/O configuration
- The storage configuration
- The processor configuration

**Figure 1 Overview of logical partitioning**



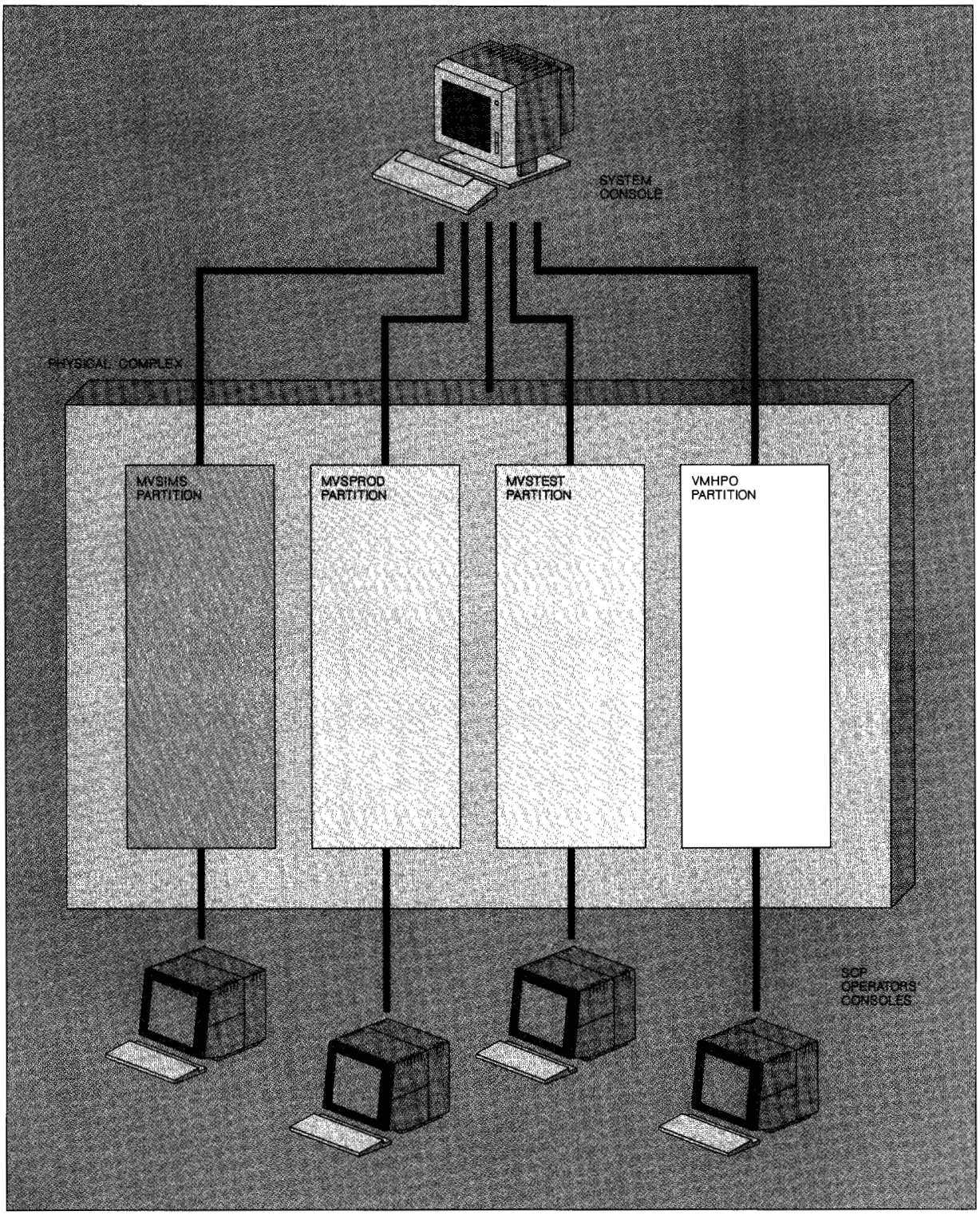



These specifications are done in two stages. The names of the partition and most of the I/O configuration are defined as input to the Input/Output Configuration Program (IOCP) and stored as part of the Input/Output Configuration Data Set (IOCDS) on the Service Processor of the 3090E or the ES/3090S processor complex. The user provides the eight-character names of each of the partitions to be used with this IOCDS in the IOCP input. These names will be used later by the system operator to operate each of the partitions. The physical I/O configuration is allocated to each of the partitions on a channel path basis since channel paths are dedicated to a partition. On the channel path identification (CHPID) macroinstruction input to IOCP, the user indicates which partition owns the channel path. The IOCP generates subchannels and logical control units for the partition for the I/O equipment attached to the channel path. Channel paths can be reconfigured from one partition to another via commands from the system console or the system control program (SCP) operator console.

The remaining resources for the partition (the storage configuration, the processor configuration, and the remaining I/O configuration) are specified with panels on the system console of the processor complex. The storage configuration consists of the amount of main storage and expanded storage in 1-megabyte increments that is required for the partition. The processor configuration for the partition consists of the number of logical processors required, the number of logical vector facilities required, and the mode of the partition, and takes into account whether dedicated or shared use of these elements is required. Shared partitions share one or more physical processors; therefore, each partition is given a weight (a relative priority) which is used by the LPAR dispatcher to allocate and control the access of the partition to the physical processors. Each partition may operate in one of the following modes: System/370, 370-XA, or ESA/370. If the partition is System/370, the remaining I/O configuration consists of associating System/370 channel numbers with specific channel paths. For LPAR this is done with a system console panel rather than using IOCP.

After a partition is defined, no resources are allocated to it, except for the I/O configuration, until the partition is activated. At activation the resource requirements of the partition are compared with the available physical resources to determine if the activation will be allowed. Allocation of the storage configuration for the partition does not occur until activation

**Figure 2 Partitioning operational overview**



of the partition; therefore, more storage can be defined than is installed, but the amount of storage in use (activated) is limited to the amount installed. Both main and expanded storage are allocated in contiguous 1-megabyte blocks. Dedicated processors and vector facilities are allocated at activation of the partition; the allocation can be completed only if the required number of processors and/or vector facilities are available for dedication. Partitions with shared processors can only be activated if the required number is less than or equal to the number of physically installed processors minus the number of dedicated processors. The same test is used for partitions requesting shared vector facilities. If resources are available for the partition at activation, they are allocated to the partition and left in power-on reset state; main and expanded storage are cleared, the channel paths are reset, and the logical processors are reset. The partition is ready for the IPL (initial program load) of an operating system. Activation is the logical power-on reset of the resources of the partition.

**Operational controls.** Operation of the LPAR environment involves the following activities:

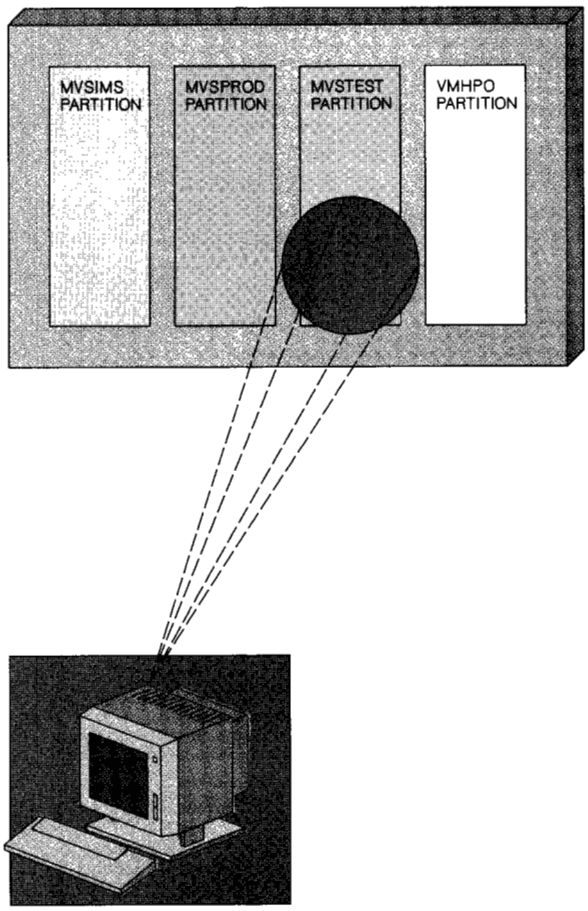
- Controlling the physical processor complex
- Managing the LPAR controls
- Controlling the logical processor complex of the partition
- Controlling the operating system running in each partition

The first three of these operations are performed at the system console; the last item is done at the operator's console for each operating system. Figure 2 shows an overview of the operational controls of LPAR.

The physical processor complex is operated from the system console. In general, the functions used to control the physical processor complex for LPAR are the same as the functions used when the processor complex is operating in one of the basic modes (System/370, 370-XA, or ESA/370). These functions include:

- Releasing the configuration
- Initiating power-on reset
- Selecting and controlling the IOCDS
- Handling problem reporting with the Remote Service Facility or the Problem Analysis Facility
- Defining and activating the System Activity Display (SAD)

**Figure 3 Console control of partitions**




- Defining the storage and processors of the physical configuration

Control of the LPAR environment is done from the system console. Activities involved in control of LPAR include defining resources for the partitions, activation and deactivation of the partitions, establishing partition dispatching weights, identifying dedicated and shared partitions, establishing a correspondence between System/370 channel numbers and channel paths, and displaying storage maps for main and expanded storage.

The logical processor complex (the partition) is also operated from the system console. Since the 3090E

and ES/3090S processors have a single system console, the system console is shared among all of the active partitions. Sharing is done by a windowing technique where the system console is controlling one partition at a time. Figure 3 shows that the system console can be used to control the four partitions by presenting console frames for one of the partitions. The partition that is currently being controlled by the system console is the target partition designated by the system operator. The target partition can be changed dynamically. Figure 3 also shows that the current target partition is MVSTEST; therefore, any controlling functions entered from the system console will act on partition MVSTEST. The functions available for controlling a partition include IPL, all forms of reset, alter/display of storage, and stop/start.

Each of the operating systems running in a partition is controlled by an operator's console that is attached to one of the channel paths for that partition.

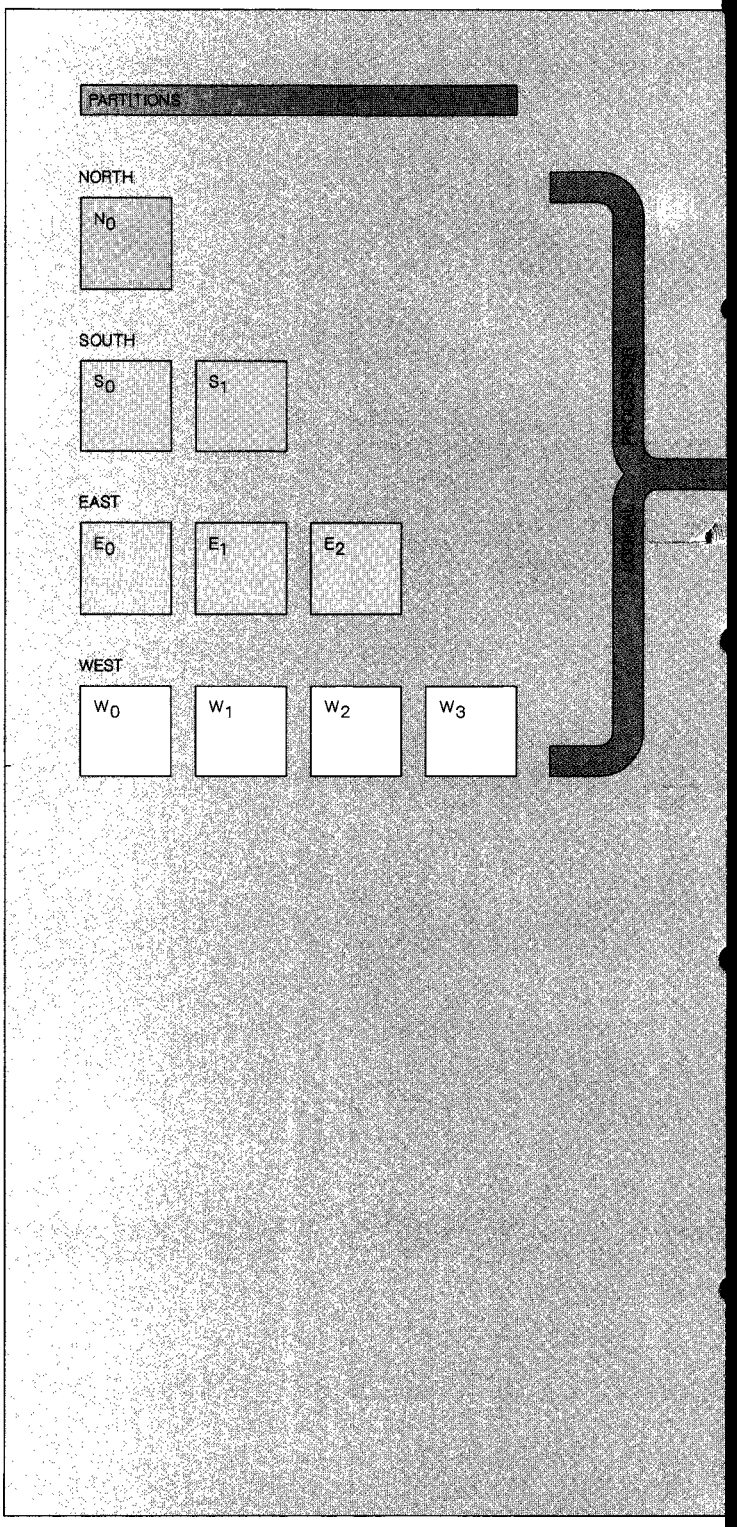
**Workload management.** Since all of the resources for a dedicated partition are used exclusively by that partition, there is no dynamic workload management for LPAR to perform for dedicated partitions. For this reason this section on workload management will focus exclusively on shared partitions.

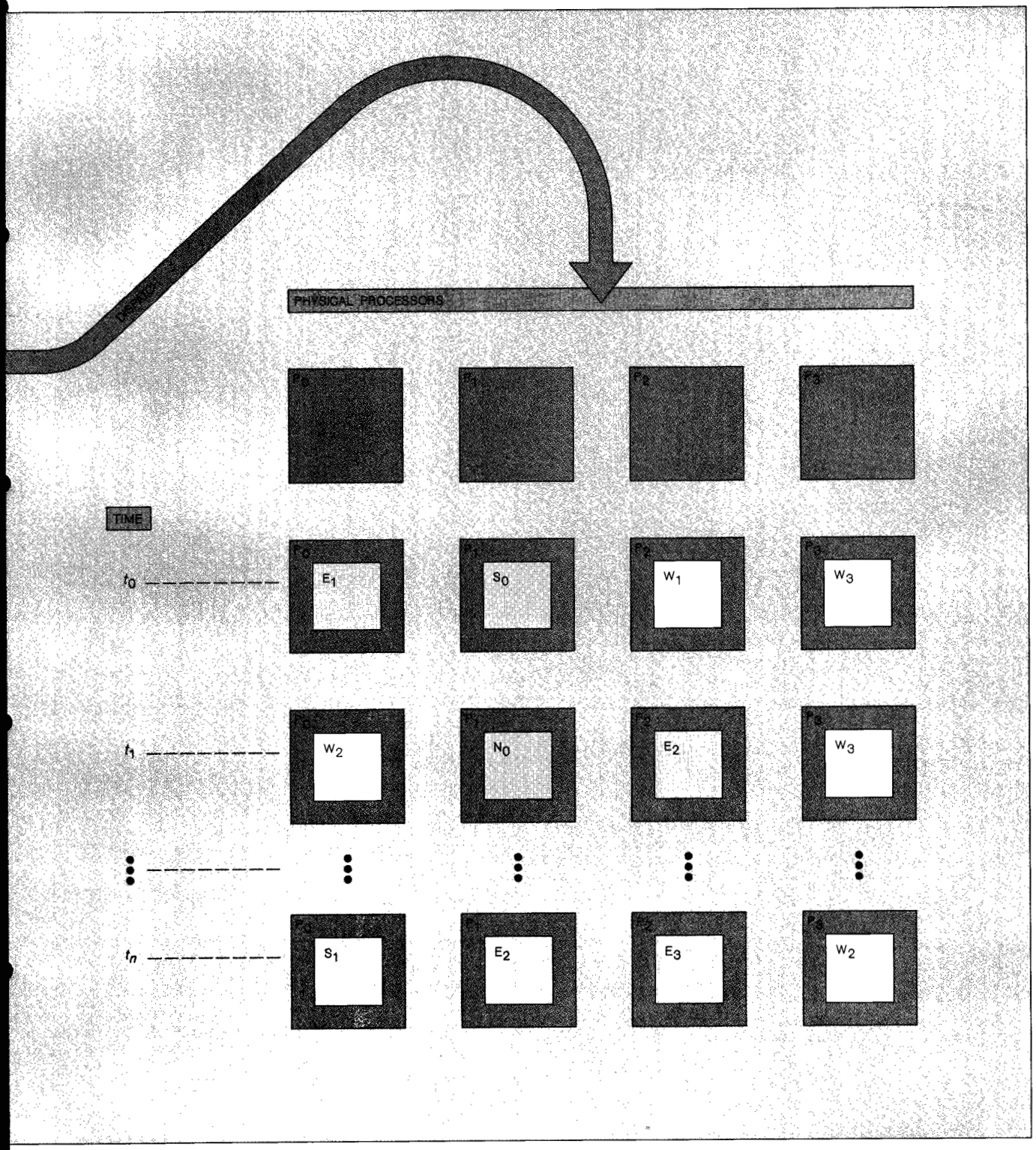
Workload management consists of allocating the logical processors of each partition to the available physical processors in a way that provides good I/O response while maximizing the use of the physical processors in accordance with a user-defined policy. This concept is illustrated in Figure 4 where four partitions (North, South, East, West) are being run on a processor with four physical CPs (processors). North, South, East, and West have one, two, three, and four logical processors respectively.

The LPAR workload manager is an event-driven dispatcher which dispatches logical processors (e.g., N0, S1, E0, W2, etc. in Figure 4) to physical processors (CP0, . . . , CP3 in Figure 4).

The LPAR dispatcher was designed with the objective of allocating all available physical processor cycles to logical processors that are ready to execute instructions while maintaining good I/O response. In order to achieve this objective, LPAR makes each individual logical processor of every nondedicated partition a separately dispatchable unit of work. This means that partitions do not have to have the same number of logical processors as the available physical proces-

**Figure 4 LPAR dispatching**




DISPATCH
PHYSICAL PROCESSORS
TIME
$t_0$
$t_1$
$t_n$
$E_1$
$S_0$
$W_1$
$W_3$
$W_2$
$N_0$
$E_2$
$W_3$
$S_1$
$E_2$
$E_3$
$W_2$

sors and that logical processors from several different partitions may be active concurrently. Further, it is not necessary for all of the logical processors of a single partition to be active concurrently. This is shown in Figure 5, which represents a snapshot of the execution of logical processors on the physical processors at an arbitrary point in time.

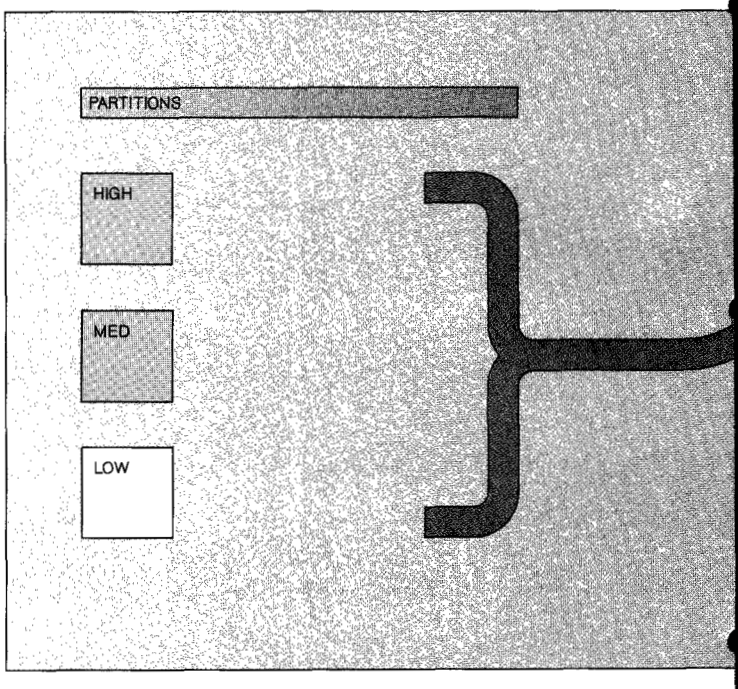
The LPAR dispatcher utilizes a number of classical dispatching/scheduling techniques to achieve its objective:

1. *Weights:* Each partition has a user-defined *weight* (priority) which is used by the dispatcher to determine scheduling priority.
2. *Wait Detection:* When a logical processor enters a wait, the dispatcher will detect the wait and select another logical processor to run.
3. *I/O Preemption:* When an I/O interruption is pending for a logical processor of a partition that is of higher priority than the currently executing logical processor, the dispatcher will preempt the lower-priority logical processor and dispatch the higher-priority logical processor.
4. *Dispatch Interval:* The LPAR dispatcher maintains a maximum time interval in which a logical processor may run for any single dispatch of the logical processor. If a logical processor is still active at the end of the dispatch interval, the dispatcher will preempt the logical processor and dispatch the highest-priority logical processor that is ready.
5. *SCP Indicated End:* An SCP (operating system) may recognize that it is doing work which may be productive when it is the only SCP using the processor complex, but which is unproductive in the LPAR environment (e.g., VM is in active wait state looking for ready work, and MVS is spinning for locks). LPAR provides an interface for an SCP that is recognizing these situations so it can voluntarily give up its dispatch interval and permit the dispatcher to dispatch the highest-priority logical processor that is ready.

As is true with many dispatchers, the amount of overhead of the dispatcher varies inversely with the utilization of the physical processors. When the partitions require all of the processor cycles, the dispatcher overhead is extremely low; as the utilization falls, the dispatcher becomes more active in looking for work.

Figure 5 illustrates the behavior of the LPAR dispatcher. In this example, there are three partitions

**Figure 5 Event-driven dispatching**

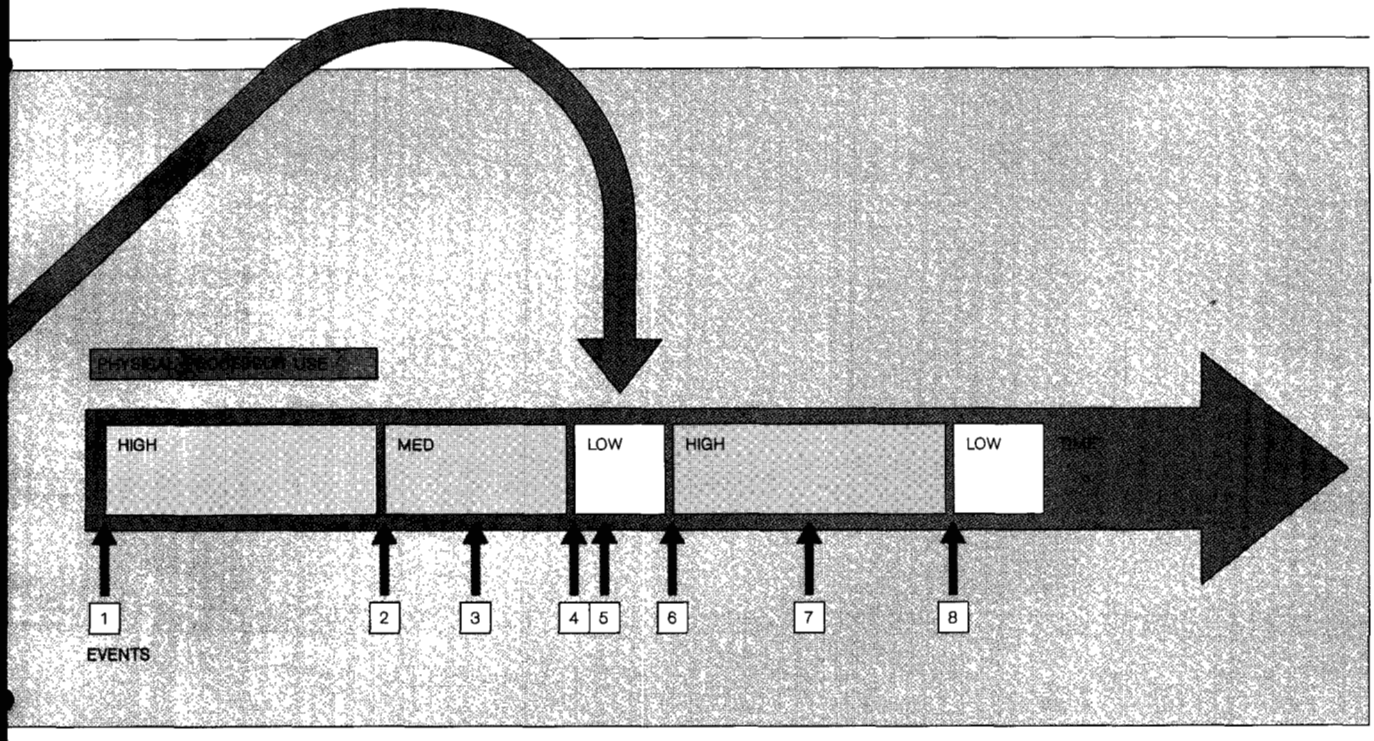



(High, Med, Low—the names imply their priorities) each with one logical processor running on a processor complex with a single control program (CP). The numbers refer to the following events:

1. Initial dispatch of High
2. High enters enabled wait for I/O; Med is dispatched
3. I/O interrupt for Low is pending; no change in dispatch
4. Med voluntarily gives up dispatch interval (active wait); Low is dispatched
5. Pending I/O interrupt for Low presented when Low dispatched
6. I/O interrupt for High; Low is preempted and High dispatched
7. I/O interrupt for High with no effect on dispatcher
8. The time interval of High ends; Low dispatched because it has become highest-priority logical processor ready to run

Two external controls can be used to affect the amount of processor resource and I/O responsiveness that a partition receives:

- Partition weights
- The number of active logical processors in the partition




LPAR provides a system console panel for the operator to set the weight of each logical partition. The weight can be a number in the range of 1 to 999. The LPAR dispatcher converts the partition weight (PW) into a dispatching weight (the CPW) for each logical processor in the partition by dividing PW by the number of logical processors in the partition. The CPWs are used by the LPAR dispatcher to determine which logical processor gets dispatched when there is more than one eligible logical processor ready for dispatching. When total processor utilization is less than 100 percent, the CPW will not limit the amount of access any partition has to the physical processors. In 100 percent utilization periods, the CPW is used to determine a target amount of access that any logical partition is to get. This is shown in Examples 1 and 2 of Table 1.

The CPWs are used to calculate a target amount of processor utilization for each logical processor and for each partition. This calculation will represent the distribution of processor resources if each partition is actively competing for the resources; however, if any partition is not able to use its target share of the physical processor resources, its unused allotment will be divided among the remaining competing partitions. Figure 6 shows the distribution of physical processor resources on a two-processor system with two partitions, each having two logical processors, where VMPROFS had a weight of 400 and MVSBATCH had a weight of 100.

The other control over processor resource consumption is the number of logical processors in the partition. A logical processor cannot consume more resources than the worth of a single physical processor. By "varying" a logical processor "off line" or "on line" to the operating system running in the partition, the user can change the ability of a partition to compete for processor resources.

**Reliability, availability, and serviceability.** A key design philosophy of LPAR is that all hardware or software failures associated with a specific logical partition should not affect other partitions. Hardware failures that are localized to a functional unit (e.g., a processor) are passed to the logical partition that was in control when the failure occurred. The appropriate error information is presented to the logical partition, and the running operating system is responsible for its own recovery.

There are, of course, shared hardware elements whose failure could bring down the entire complex,

**Figure 6 Processor utilization**

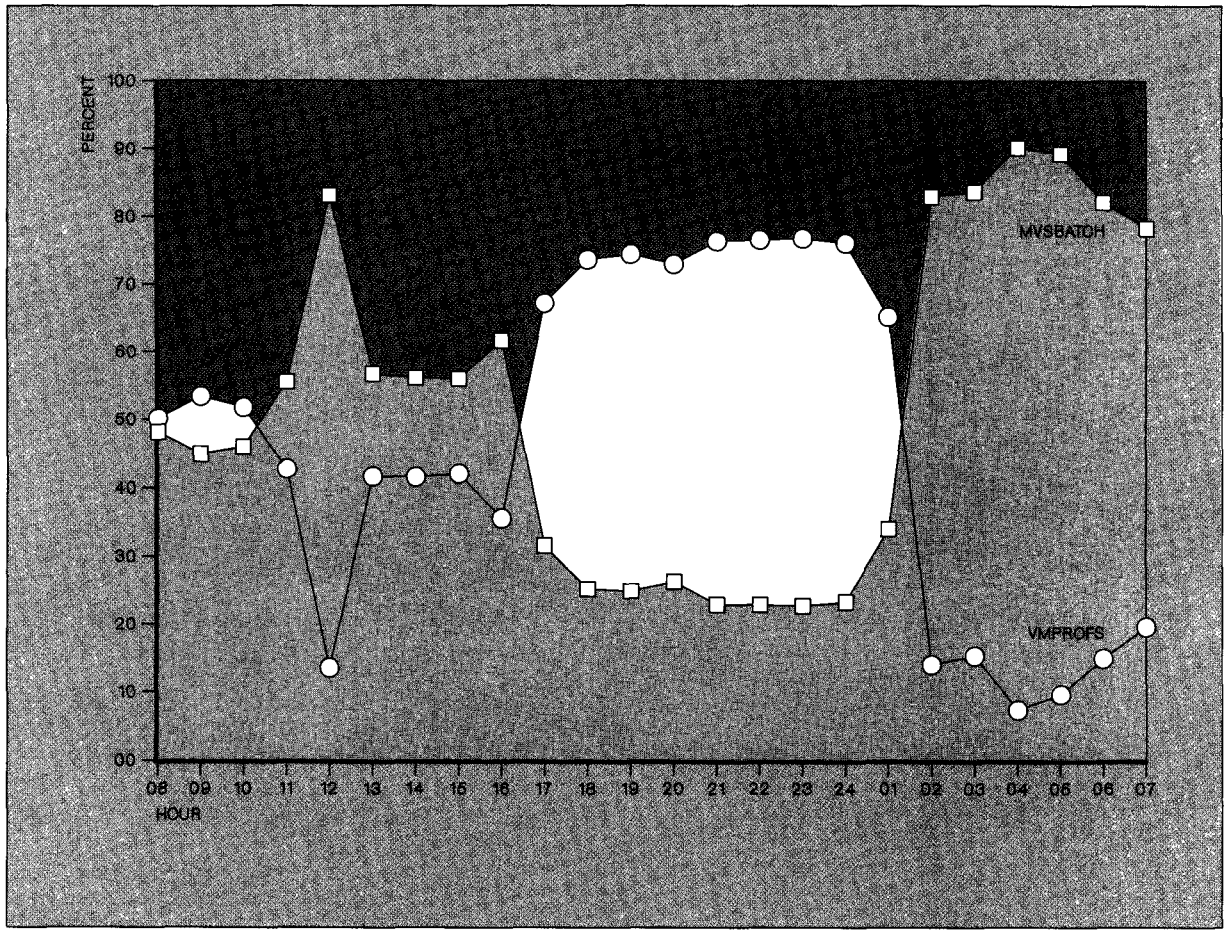



**Table 1 Partition weights on a 3090-400E with four physical CPs**

| Partition Name | Partition Weight (PW) | Number of Logical CPs | Logical CP Weight (CPW) | Logical CP Target | LP Target |
|---|---|---|---|---|---|
| **Example 1** | | | | | |
| DICK | 300 | 3 | 100 | 0.40 | 1.20 |
| FRANCK | 100 | 1 | 100 | 0.40 | 0.40 |
| MXA | 400 | 4 | 100 | 0.40 | 1.60 |
| ROAR | 200 | 2 | 100 | 0.40 | 0.80 |
| **Example 2** | | | | | |
| DICK | 200 | 1 | 200 | 0.32 | 0.32 |
| FRANCK | 600 | 2 | 300 | 0.48 | 0.96 |
| MXA | 900 | 3 | 300 | 0.48 | 1.44 |
| ROAR | 800 | 4 | 200 | 0.32 | 1.28 |

including all partitions. In this respect physical partitioning continues to provide an availability advantage due to the complete duplexing of hardware elements, and it may be combined with logical partitioning to satisfy user-specific configuration requirements.

Storage failures within the storage area assigned to a logical partition are localized to that partition and handled normally by the operating system of the partition. Similarly, since channels are dedicated to partitions, most I/O errors (including hot I/O interrupts) are localized to a partition. Software failures within a partition (e.g., loops or abnormal endings) do not affect other partitions.

A high-availability system configuration requires at least two paths from each logical partition to any critical device. To address this requirement, PR/SM permits each logical partition to have up to four paths to a device.

**Instrumentation.** When operated in LPAR mode, a 3090E or ES/3090S system has two new performance instrumentation capabilities associated with logical partitioning:

1. MVS/XA and MVS/ESA Resource Measurement Facility (RMF) reports on CPU usage by partition
2. System Activity Display by partition

The MVS/XA RMF product now produces an optional Partition Data Report as part of its Monitor I output on the basis of performance data collected through the PR/SM feature. This report indicates CPU usage per logical partition and is intended for capacity planning. The generation of the RMF Monitor II and III reports has also been updated to reflect the possibility that a logical partition may not have dedicated CPU resources.

The 3090E and ES/3090S System Activity Display (SAD) has also been extended with PR/SM LPAR mode to reflect logical partitions. For each logical partition, the user may display the current level of supervisor, problem, or total busy time, with the display normalized to show 100 percent busy time when the full share of assigned CPU resource is used. The display of channel path usage is also annotated by logical partition name. All standard SAD capabilities continue to be provided on a physical processor basis. Figure 7 shows an example of an LPAR mode SAD frame.

## VM/XA MPG

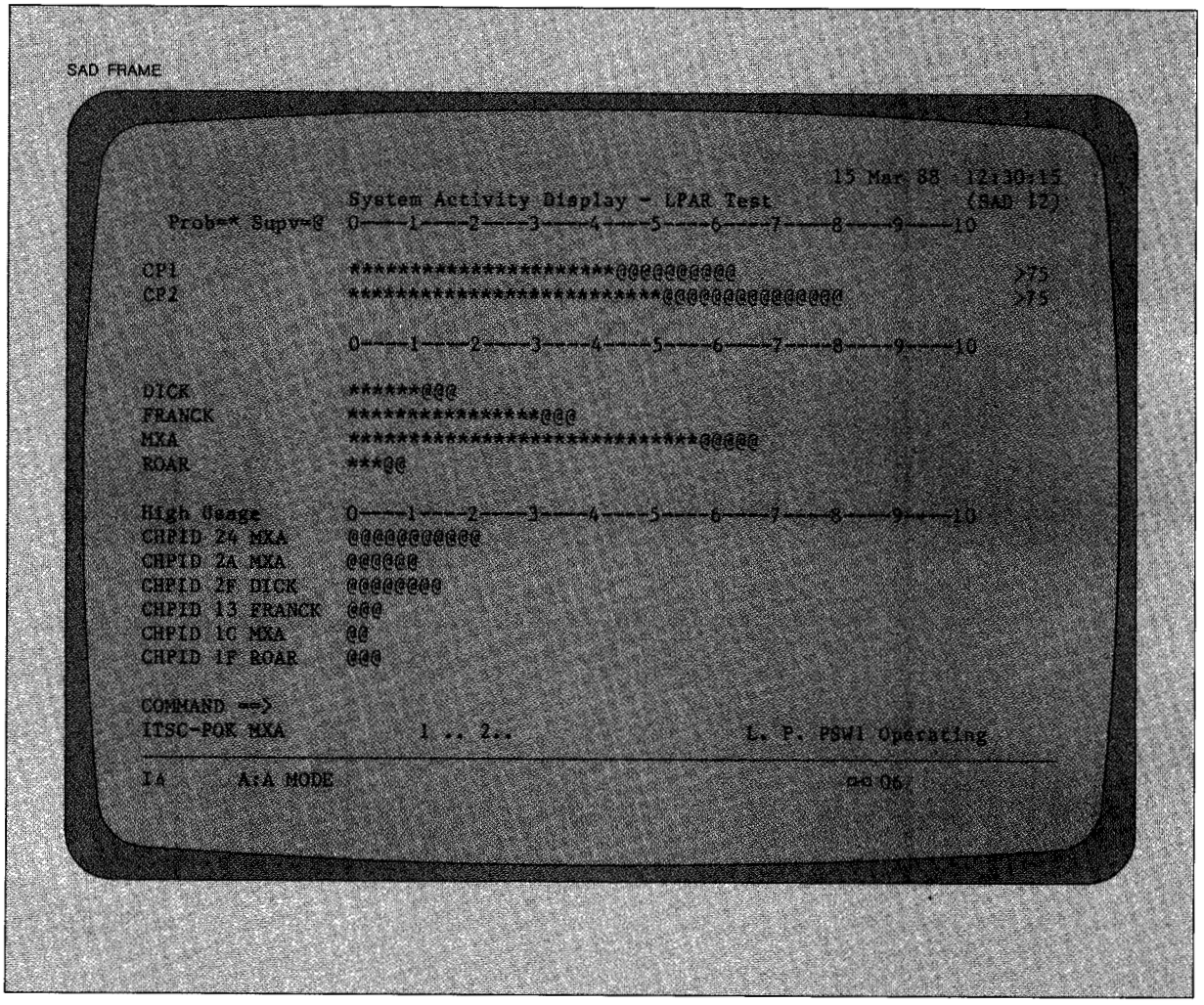
In 1987 IBM announced an enhancement to VM/XA that allows it to logically partition a 3090E or ES/3090S for use by several production operating systems. When the machine is in basic (non-LPAR) mode, the VM/XA System Product with Multiple-Preferred-Guests support (hereafter called VM/XA MPG) can make use of the PR/SM machine feature to support multiple *preferred* guests, as well as many nonpreferred guests. Along with the existing support for a *Virtual=Real* (V=R) guest, VM/XA MPG now supports up to five *Virtual=Fixed* (V=F) guests, for a total of six preferred guests.

**Overview.** As described earlier, the original purpose of VM was to create "virtual machines." That is, using a single real machine, VM can create the illusion of multiple machines. Using VM/XA, these virtual machines do not even have to be the same architectural mode as the real machine: Some could be using the System/370 architecture, and others 370-XA. If VM/XA is running on an Enterprise Systems Architecture/370 (ESA/370) machine, guests may also exploit the powerful new addressing capabilities that this architecture provides.

**Storage management.** Virtual machines also have their own storage, which need not be the same size as what exists on the real machine. To make more efficient use of the real machine, they often have less, but it is possible to define virtual machines that have even more storage than the real machine has. Specifically, VM/XA MPG manages storage in one of three ways for a guest:

1. A predefined amount of contiguous real storage, starting at absolute address zero, is set aside for exclusive use by that guest. Since every guest-absolute address maps directly to the same host-absolute address, this type of guest is classified as a V=R guest. Since only one area of real storage can start at absolute address zero, only one V=R guest can ever be logged on at a time.
2. A predefined amount of contiguous real storage, but not starting at absolute address zero, is set aside for exclusive use by that guest. Since every guest-absolute address is at a fixed displacement from the corresponding host-absolute address, this type of guest is called a V=F guest.
3. No specific amount of real storage is set aside; real storage is allocated as needed to hold recently referenced guest pages. Other guest pages are

**Figure 7 Example of LPAR mode System Activity Display (SAD)**




saved in ancillary storage, such as expanded storage or disk space reserved specifically for this purpose. This technique of managing real storage is called *paging* and allows an operating system to create a vastly greater number of *virtual* pages than there are *real* storage frames to contain them.

When a guest's storage is managed in this manner, there is no direct relationship between a guest address and the host address of the frame that contains the page. VM/XA uses dynamic address translation (DAT) tables to map guest storage. Such a guest is referred to as a *Virtual=Virtual* (V=V) guest.

Since storage pages owned by the first two categories of guests are always available immediately when needed, and since it is easier for the machine (and VM/XA) to translate the storage addresses, V=R and V=F guests are considered to be *preferred* guests. Because V=R and V=F guests generally execute faster and more efficiently than V=V guests, they are the usual vehicle by which to run production operating systems under VM/XA. The flexibility available from V=V guests is useful, however, when testing and debugging and for nonproduction guest environments.

**CPU management.** Just as a real machine might have multiple CPUs sharing storage, so VM/XA permits a

guest to define multiple *virtual* CPUs which can each execute instructions on behalf of a guest. In a production guest environment, multiple virtual CPUs provide a means by which to exploit the processing power of the real machine. Each virtual CPU represents a single guest instruction stream, and can thus only be dispatched on one real CPU at a time. If only virtual uniprocessors were supported, even a virtual CPU dispatched 100 percent of the time on one of the real CPUs would only be capable of consuming the processing power of a single real CPU. In contrast, a virtual multiprocessing guest can consume more processing power because its virtual CPUs may be concurrently dispatched by VM/XA on multiple real CPUs. However, since the processing power available from the system is limited by the number of real CPUs available, it does not make sense for a production guest to define more virtual CPUs than there are real CPUs. Also, since there is some VM/XA overhead in managing more virtual CPUs, defining excess virtual CPUs actually reduces the amount of processing power available to the guest. A production guest, therefore, should define only as many virtual CPUs as needed to consume the amount of processing power required by the guest.

**Processor power is managed by VM/XA through dedication and scheduler shares.**

**Processor resource allocation.** Processor power is managed by VM/XA through two mechanisms: *dedication* and *scheduler shares*. Through dedication, a real CPU is reserved for exclusive use by a specified virtual CPU. That real CPU will select only that virtual CPU for dispatching, and no others. Dedication allows the virtual CPU to run a bit faster for the following reasons:

- Fewer processor cache misses and cross-interrogates: Since that virtual CPU is the only one running on the real CPU, the cache of that CPU contains only lines referenced by the virtual CPU (and a small amount due to CP overhead).
- Fewer translation-lookaside buffer (TLB) misses: The translation-lookaside buffer of the processor is used by the dynamic address translation process to "remember" the results of previous translations. Since it has a limited capacity, giving the TLB of the processor to this guest exclusively improves the chances of finding the desired data in the TLB.
- Fewer exits from and entries to interpretive-execution mode:[3] A dedicated real CPU is not enabled for the same set of I/O interruption subclasses (ISCs) as a nondedicated CPU and, therefore, is interrupted to handle I/O interruptions less frequently. It also receives fewer external interruptions due to various timers. Clock comparator requests to service nondedicated guests get queued on nondedicated CPUs, and CPU timer interruptions occur less often because a dedicated guest gets a much larger minor time-slice than a nondedicated guest.
- More efficient instruction simulation: There is a software fast path for simulation of Diagnose x'44' instructions (the "voluntary time-slice end" function) when issued by a guest running on a dedicated CPU. Diagnose x'44' instructions are a frequent cause of exits from interpretive-execution mode for a multiprocessing MVS guest.
- Less time spent waiting for a CPU to become available: When a nondedicated virtual CPU leaves enabled-wait state (or otherwise becomes "ready"), it must compete with other guests in the system for a turn on a CPU. The length of time it must wait is based on system load and other scheduling considerations, but there are some situations in which VM/XA will not preempt a currently running guest in order to run a higher-priority guest. The virtual CPU is therefore forced to wait until the currently running guest gives up control of the CPU (or encounters time-slice end). This algorithm gives better total system throughput, but it reduces the responsiveness of the production guest.
- VM/XA MPG will enable the Wait-State Interpretation Capability on behalf of a dedicated virtual CPU. This machine feature prevents an exit from interpretive-execution mode when the guest enters enabled-wait state. Since most "waits" of this nature are of a very short duration for a production guest, it is more efficient to leave the machine in interpretive-execution mode than to go through the host overhead of processing the change in dispatching status.

The primary disadvantage of dedication is that when the real CPU is not needed by the virtual CPU to which it is dedicated, it sits idle, accomplishing absolutely no useful work. Overall throughput of the

system is therefore reduced if other work could be processed. Unless the virtual CPU can consume very close to what the processing power of a real CPU is worth, this disadvantage usually overshadows the advantages, so an installation should choose to dedicate a real CPU only with caution.

When a virtual CPU does not have a real CPU dedicated to it, it is managed by the scheduler on the basis of the "share" of the resources of the system assigned to it. A scheduler share is essentially a resource-consumption goal designating how much of the resources of the system are to be apportioned to that particular guest. Processing power, as a system resource, is thus allocated by the scheduler primarily on the basis of the share values assigned to each guest. A share value may be specified as either an *absolute* share or a *relative* share. An absolute share specifies a percentage of the resource to allot to that guest. On a real processor with four CPUs, therefore, an absolute share of 50 percent would denote that processing power equivalent to that of two CPUs should be reserved for that guest. A relative share is a number from 1 to 10 000 which is compared against the relative shares of other guests to determine their relative importance.[4] For example, if two guests have relative shares of 100 and 200, the scheduler will devote twice as much CPU power to the second guest as to the first (because the second number is twice as big as the first number). If these are the only two guests that are active, the second guest should be given two thirds of the system, and the first guest should be given one third of the system.

**A virtual device may or may not have a corresponding real device.**

**I/O device management.** When executing under VM/XA, a guest has virtual devices which are managed by VM/XA such that they appear real to the guest. A virtual device may or may not have a corresponding real device, and if it does, that real device may be dedicated or nondedicated. Much like a dedicated CPU, a dedicated device is one reserved exclusively for the support of a particular virtual device. A nondedicated device may be used to support several virtual devices, which may even be shared by different guests. A virtual device that does not have a corresponding real device is called a simulated device. A simulated device may be used, for example, to provide printer, card reader, or card-punch capabilities without needing a real printer, reader, or punch. Output from simulated printers and punches is kept in "spool files" and may be dynamically assigned to real printers and punches as desired. Another type of device that can be completely simulated by VM/XA is a channel-to-channel adapter. Such a device can allow communication between two guest operating systems.

When devices are dedicated, the basic allocation unit is a device; VM/XA manages channel paths. With the introduction of Start Interpretive Execution (SIE) Assist in 1985, devices dedicated to the V=R guest benefitted from special treatment by the channel subsystem. For a guest running MVS/370 or MVS/XA, the channel subsystem began to take advantage of the guest's fixed-storage layout by "interpreting" most guest I/O instructions without host intervention, thus no longer requiring the machine to exit interpretive-execution mode. Similarly, many arriving I/O interruptions belonging to the V=R guest could be "interpreted" by the machine without host intervention and without leaving interpretive-execution mode. Since the handling of every exit from interpretive-execution mode requires host processing, significantly improved performance was realized from the reduced frequency and from avoiding the VM/XA overhead of simulating the I/O instructions and interruptions.

VM/XA MPG uses the PR/SM machine feature to make this dramatic performance improvement available to V=F guests as well as V=R guests. The resulting performance is nearly that of a V=R guest, thus making V=F guests a practical mechanism to run production operating systems. On the 3090E and the ES/3090S, VM/XA supports five V=F guests and a V=R guest for a total of six preferred guests that may exploit the PR/SM machine feature.

Though nondedicated devices cannot achieve the high level of performance that dedicated devices can, they do permit the *sharing* of real devices by multiple guests. For example, a real disk may be divided into several *minidisks*. As viewed by a guest, a minidisk generally has the same characteristics as a real disk, except for its size. Minidisks allow different areas of

a real disk to be allocated to different guests and also allow the sharing of the *same* disk by different guests. VM/XA allows a real disk to be mapped by a single "full pack" minidisk, and this minidisk may then be "linked" to multiple virtual machines. The virtual machines control such sharing, if necessary, through the I/O protocols of Reserve and Release.

**Defining a virtual machine.** Each virtual machine that is permitted to log on to a VM/XA system is described in a file called the *user directory*. The directory describes the attributes of each virtual machine, including the identification (userid), privilege class, storage size, CPU configuration, device configuration, scheduler share allocation, and additional information of the virtual machine that is used to define or limit its capabilities. This information provides a starting point for the virtual machine definition. Later changes may be made through VM/XA commands issued dynamically from authorized userids. Commands are available to add or delete I/O devices, create new virtual CPUs, change scheduler shares, etc. Many such changes to the virtual configuration can even be done without disturbing an executing production guest, which may allow an operator to reconfigure a guest dynamically in response to changing resource requirements.

### Comparison of LPAR and VM/XA MPG

With PR/SM, IBM offers users a choice between two methods for the logical partitioning of a 3090E or ES/3090S: LPAR-mode operation and VM/XA with MPG support. LPAR and VM/XA MPG are complementary offerings, each satisfying a unique set of user requirements. This section is a comparison of LPAR and VM/XA MPG, evaluating such areas as user experience, flexibility, performance, and reliability.

**User experience.** Users with no VM experience may find that LPAR mode is the easier way to partition their systems. In LPAR mode, the familiar hardware console screens of the base machine are extended to support multiple partitions, as is the IOCP process for defining the system I/O configuration.

In order to use VM/XA MPG, one must configure, generate, and build a VM/XA system. Like any large operating system, VM/XA requires a certain amount of expertise to set up. Non-VM users who wish to logically partition their machines generally do not have system programmers with VM skills, and do not want to incur that staffing expense, nor the software license fee for VM. For these users, therefore, LPAR is a more attractive alternative.

In contrast, VM users may find that VM/XA MPG provides a more natural and flexible partitioning alternative. These users are generally using VM for reasons beyond its guest-machine capability, as described below, and they already have the required VM skills. These users would consequently base their decision on some of the remaining differences.

**External interfaces.** As a machine feature, LPAR provides its external interface through the system console. That is, console menus are provided to activate and control partitions, and the operator for a partition uses a panel that is very similar to the operator panel on the base, unpartitioned machine. It is therefore an easier migration in terms of human factors. However, in most installations, the system console is placed in a secure area and may therefore be inconvenient to access, although the IBM NetView™/ISCF (Inter-System Control Facility) product may be used to control the console from a remote terminal.

The VM/XA operational interface is via existing VM/XA commands, which are well known to experienced VM operators. These commands can be entered from any authorized terminal (any number are permitted), without requiring NetView.

**Types of guests.** LPAR supports up to six high-performance partitions (on the ES/3090S). VM/XA MPG supports up to six preferred guests *and* a large number of nonpreferred (V=V) guests. These additional nonpreferred guests can be very useful for test, development, and CMS-intensive applications that do not require much processing resource.

**Debugging tools.** While LPAR provides rudimentary debugging tools appropriate to a machine feature, VM/XA offers a robust set of debugging tools appropriate to an operating system. These tools include, for example, commands to trace the execution of instructions in the guest operating system. This facility is very flexible and allows quite a rich set of debugging "traps" to be created. A facility such as this one can be invaluable in tracking down subtle bugs.

**Monitoring tools.** VM/XA offers standard monitoring facilities, such as would be found in any operating system, that can be used to measure and tune the system to produce optimum overall performance. LPAR provides processor and channel utilization information which must be recorded and reduced by the operating systems running in the partitions.

**Channel path configuration.** With LPAR, channel paths must be dedicated to partitions. This dedication ensures that all I/O operations on a channel path are associated with a single partition, avoiding any need for the machine to be involved in I/O authority-checking or error recovery, both of which may be device-dependent. However, if multiple partitions need to share an I/O device, each must have at least one dedicated channel path to the device.

In contrast, VM/XA allows its guests to share channel paths and devices by assigning resources at the device level. That is, an installation may specify what devices are to be dedicated to the preferred guest without consideration for the rest of the devices on the channel path. VM/XA is thus more flexible as it permits an installation to give a preferred guest access to some devices on a control unit without giving the guest access to all devices on the control unit.

**Virtual devices.** LPAR, a machine feature, does not have the ability to "simulate" devices as does VM/XA. VM/XA uses software to create "virtual" devices, devices that appear like real ones to a guest but which do not in fact have a real equivalent. For instance, by using VM/XA one can define a virtual printer. Output sent to that virtual printer will be collected ("spooled") by VM/XA and can subsequently be directed to a variety of destinations using appropriate VM/XA commands. It can be printed on a real printer, put on tape, or even treated as input for a virtual "card-reader" device.

Another kind of virtual device that is often useful in a production guest environment is a virtual channel-to-channel adapter. This virtual device can be used to connect two guest operating systems, thus allowing them to communicate. If LPAR is used, physical hardware is required to perform the equivalent function.

**Performance.** Since LPAR is a machine feature, it offers high performance, comparable to that of an unpartitioned machine. VM/XA MPG, through the use of the PR/SM hardware, can also provide comparable performance if dedicated I/O resources are allocated to the guest machine. If the VM guest exploits the VM capability to share devices, it will incur some additional overhead necessary to implement the controlled sharing of the physical resources. For similar dedicated I/O configurations, with no shared devices, the performance of VM/XA is typically within 1 or 2 percent of that of the LPAR mode.

**Reliability and availability.** The implementation of LPAR mode, in hardware and microcode, is much smaller and simpler than that of VM/XA, and the reliability of LPAR mode is comparable to that of an unpartitioned machine. VM/XA provides a software recovery capability, called Preferred-Guest Recovery, through which VM/XA can usually sustain the V=R guest machine despite a failure in VM itself.

## Conclusion

The Processor Resource/Systems Manager provides an efficient and flexible capability to run multiple operating systems on a single IBM 3090E or ES/3090S processor complex. Through special hardware and microcode, it virtualizes the base machine to create multiple partitions, each of which is a logical machine with its own set of resources.

PR/SM may be used in either of two ways: directly from the system console (LPAR mode) or indirectly through the VM/XA SP operating system. Via LPAR mode, the PR/SM feature expands the base machine functions to include logical machine definition and control and physical resource management (e.g., partition dispatching).

Each PR/SM partition is an image of the underlying 3090E or ES/3090S machine, but each may be configured differently and operated independently. Thus, today, one partition might be a System/370-mode logical machine with no expanded storage, whereas another could be an ESA-mode machine with expanded storage and hiperspaces. As further architectural enhancements are made, PR/SM can readily serve as a means of migration and/or coexistence.



## Cited references and notes

1. R. A. Meyer and L. H. Seawright, "A virtual machine time-sharing system," *IBM Systems Journal* **9**, No. 3, 199–218 (1970).
2. J. P. Buzen and U. O. Gagliardi, "The evolution of virtual machine architecture," *Proceedings of AFIPS NCC* **42**, 291–299 (June 1973).
3. The machine is put in interpretive-execution mode when VM/XA dispatches guests. The machine enters interpretive-execution mode when a Start Interpretive Execution (SIE) instruction is issued and exits when host intervention is required for proper guest simulation, or when a host interruption occurs.
4. In this regard, relative shares work much the same as the LPAR "partition weights" described earlier.

**Terry L. Borden** *IBM Data Systems Division, P.O. Box 390, Poughkeepsie, New York 12602.* Mr. Borden is a senior technical staff member at IBM's Meyers Corners Laboratory. He joined IBM as an associate programmer in 1970 in a programming test technology area. During his career at IBM his work has included developing support systems for program development, MVS design and performance, CPU architecture, high-availability design, and PR/SM design. He is the recipient of a Division Award for MVS Cross Memory Services design, Outstanding Technical Achievement Awards for high-availability design and for PR/SM design, and a First Level Patent Award. Mr. Borden's current responsibility is for MVS architecture and design. He received a B.S. degree in computer science from the University of Illinois.

**James P. Hennessy** *IBM Data Systems Division, P.O. Box 100, Kingston, New York 12401.* Mr. Hennessy is a staff programmer in VM/XA development at the IBM Kingston Programming Laboratory. He received his B.S. degree in computer science from Rensselaer Polytechnic Institute in 1982, and joined IBM and VM/XA development the same year. Since then, he has been involved in many enhancements to VM/XA in the areas of real and virtual CPU management. Most recently, Mr. Hennessy designed and implemented a portion of Multiple Preferred Guests support available in VM/XA SP1.

**James W. Rymarczyk** *IBM Data Systems Division, P.O. Box 390, Poughkeepsie, New York 12602.* Mr. Rymarczyk joined the IBM Boston Programming Center in 1968 as a programmer working on the design of an experimental time-sharing system. In 1972 he transferred to the IBM Poughkeepsie Laboratory where he worked on hardware and microcode design projects for future systems. He was a principal logic designer for the IBM 3033 processor beginning in 1975 and became manager of Performance Analysis and Measurement in 1978. In 1983 Mr. Rymarczyk became manager of Processor Architecture and System Structure, and in 1984 he was promoted to program manager of Product Design and Verification. He is currently a Senior Technical Staff Member in the area of Systems Architecture and Performance. Mr. Rymarczyk received a BSEE from the Massachusetts Institute of Technology and has served as Adjunct Professor of Computer Science with Union College.

Reprint Order No. G321-5350.