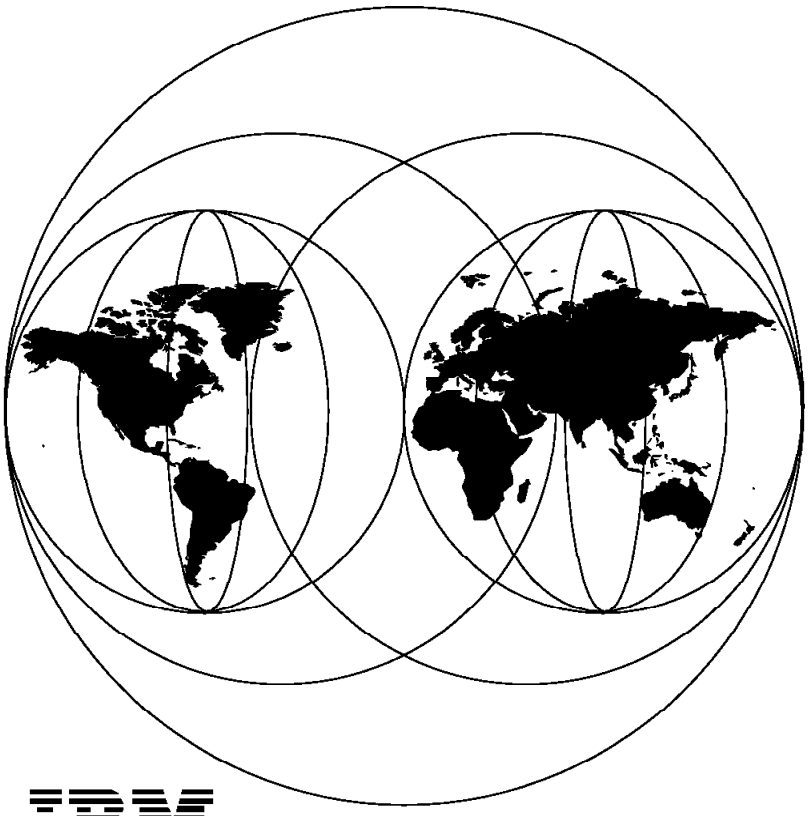
SG24-2583-01

RS/6000 SMP Enterprise Servers Architecture and Implementation

December 1996



IBM

International Technical Support Organization

Austin Center

International Technical Support Organization

SG24-2583-01

RS/6000 SMP Enterprise Servers Architecture and Implementation

December 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special Notices” on page 351.

Second Edition (December 1996)

This edition applies to IBM RS/6000 SMP servers models G40, J40 and R40 and IBM AIX Version 4.1.4 and 4.2.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This redbook describes the family of RS/6000 Symmetric Multiprocessor (SMP) servers that run AIX V4. It covers multiprocessing concepts, AIX V4 threads implementation, the SMP servers hardware architecture with a specific focus on the memory subsystem. It describes the SMP servers models G40, J40 and R40, the SystemGuard service processor as well as Service Director for RS/6000. It highlights the Cluster Power Controller (CPC) feature. The redbook also describes how to install an SMP system with AIX V4 and gives hints and tips on how to upgrade a UP system to an SMP. Finally, it covers some of the AIX V4 performance tools that are useful for using and tuning an SMP.

This document is intended for customers, system engineers and anyone who needs to understand the IBM RS/6000 SMP servers in terms of design and for those who plan to set up an SMP system.

A basic knowledge of AIX V4 is assumed.

How This Redbook Is Organized

The document is organized as follows:

- Chapter 1, “Multiprocessing Concepts” on page 1

  This chapter provides an overview of the concepts related to designing a Symmetric Multiprocessor (SMP) system. It helps the reader to understand what is different from a uniprocessor system.

- Chapter 2, “Introduction to AIX V4 Threads” on page 31

  This chapter describes how threads are implemented in AIX V4 in order to support SMP systems.

- Chapter 3, “SMP Servers Architecture” on page 49

  This chapter describes the design rationale of the RS/6000 SMP servers for a commercial environment. It highlights the hardware architecture of the memory subsystem including the Data Crossbar switch.

- Chapter 4, “SMP Servers Hardware Features” on page 71

  This chapter describes the three RS/6000 SMP servers (Models G40, J40 and R40) based on the PowerPC 604 processor, and their related hardware features. It also highlights some hardware specifics to these models.

- Chapter 5, “SystemGuard” on page 105

  This chapter describes the SystemGuard service processor that is available with the RS/6000 SMP servers. It gives details on how to use some of its features.

- Chapter 6, “Service Director for RS/6000” on page 151

  This chapter provides an overview of the Service Director offering from IBM, and give details of how to implement this service on RS/6000 SMP Servers.

- Chapter 7, “Cluster Power Controller” on page 175

  This chapter describes the Cluster Power Controller which can be used to control one or multiple SMP servers and associated peripherals.

- Chapter 8, “Installing an SMP Server” on page 203

  This chapter provides an overview of installing an SMP server with AIX V4 and some of the features that are useful in an SMP environment.

- Chapter 9, “UP to SMP Upgrade” on page 263

  This chapter describes the process for upgrading a UP server to an SMP server.

- Chapter 10, “SMP Performance Tools” on page 293

  This chapter shows how to use some of the performance tools to observe characteristics of an SMP server running AIX V4.

- Appendix A, “SystemGuard Remote Operation Configuration” on page 325

  This appendix provides samples of modem files used for SystemGuard console mirroring.

- Appendix B, “Sample Programs” on page 333

  This appendix provides multithreaded sample programs that can be used on an SMP for testing or demonstration purposes.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Austin Center.

The project was designed and managed by:

Yves Bex
International Technical Support Organization, Austin Center

The authors of this document are:

Jon Gittoes
IBM United Kingdom

Ron Barker
IBM Dallas

Dominique Lacan
IBM France

Thomas Sinou
IBM France

Chai Cher Kion
IBM Singapore

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Jim Nicholson
IBM Austin

Bret Olszewski
IBM Austin

Luc Smolders
IBM Austin

Ken Pearson
IBM Austin

Austin Newton
IBM Dallas

Julie Craft
IBM Austin

Special thanks to the editors of this publication. Their hard work, dedication and perseverance greatly enhanced the readability and visual presentation of our technical information.

Marcus Brewer, Editor
IBM ITSO, Austin Center

Stephen Riggs, Editor
IBM ITSO, Austin Center

Rebeca Rodriguez, Editor
IBM ITSO, Austin Center

The authors of the first edition of this document were:

Yves Bex
IBM ITSO, Austin Center

Bob Minns
IBM ITSO, Austin Center

Rob Hendry
IBM South Africa

Satish Sharma
IBM South Africa

Colin Fearnley
IBM South Africa

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Chapter 1. Multiprocessing Concepts

The purpose of this chapter is to introduce concepts, terms and abbreviations related to symmetric multiprocessing.

1.1 Multiprocessing versus Uniprocessing

A uniprocessor (UP) can accomplish parallelism inside the processor itself. For example, the fixed point unit and the floating point unit can run several instructions within the same CPU (Central Processing Unit) cycle. POWER, POWER2 and PowerPC architectures have a very high level of instruction parallelism. *However, only one task at a time can be processed.*

Uniprocessor designs have built-in bottlenecks. The address and data buses restrict data transfers to a one-at-a-time flow of traffic. The program counter forces instructions to be run in strict sequence. Even if improvements in performance are achieved by means of faster processors and more instruction parallelism, operations are still run in strict sequence. However, in a uniprocessor, an increase in processor speed is not the total answer because other factors, such as the system bus and memory, come into play.

Adding more processors seems to be a good solution to increase the overall performance of a system. Having more processors in the system increases the system throughput because the system can perform more than one task at a time. However, the increase in performance is not directly proportional to an increase in the number of processors because there are many other factors to be taken into consideration, such as resource sharing.

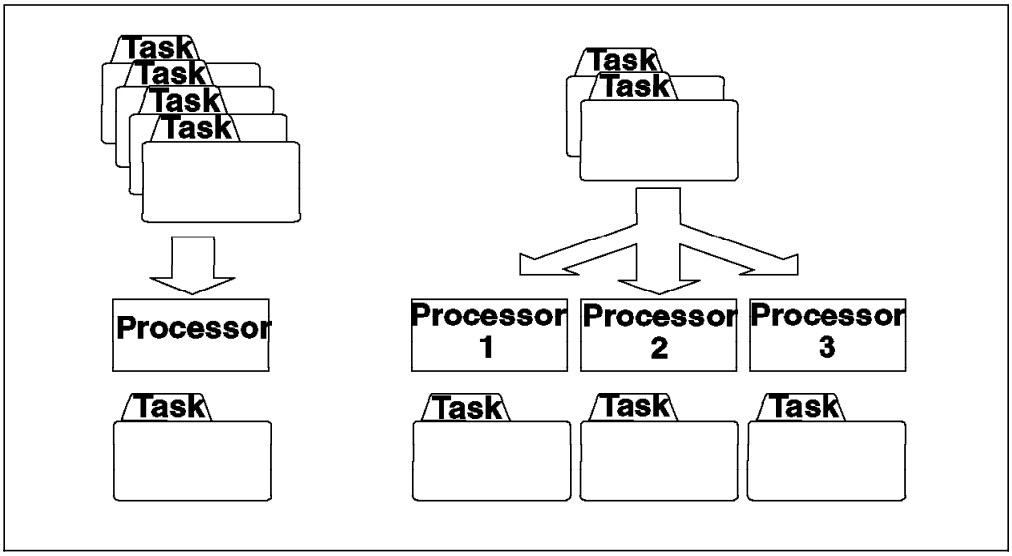


Figure 1. Uniprocessing versus Multiprocessing

1.2 Multiprocessing Issues

Multiprocessing involves using more than one CPU. Some of the more important aspects to consider when designing a multiprocessor (MP) are:

- **Are resources shared?**

  Do the processors have their own resources, or do the processors share them? Resources to consider include the operating system, the memory subsystem, the I/O subsystem and devices.

- **Are the processors equal?**

  Are all the processors equal, or are some of them specialized in specific tasks? For instance, some processors might do integer arithmetic only, and others might do floating point operations only.

- **How are the processors connected?**

  They might be loosely coupled through a Local Area Network (Ethernet, token-ring, FDDI and so forth) or tightly coupled through a switch, a crossbar, a bus, or a similar technology.

- **How easily can the system be enhanced or upgraded at a later date?**

  Will it be easy to add another processor? How much performance can we expect to get from a processor upgrade?

  Usually, the addition of a new processor will not cause system throughput to increase by the rated capacity of the new processor. This is because there is additional operating system overhead, increased contention for system resources and hardware delays in switching and routing transmissions between an increased number of components.

- **What happens if one of the processors fails?**

  If one processor fails, can the system continue operation on the remaining processors? Is it necessary to reboot after a processor failure in order to reconfigure the system?

1.2.1 Sharing Resources

When resources must be be shared, there are a number of aspects that need to be taken into account. First, the bandwidth between the processors and resources, such as memory and I/O subsystem, must be as high as possible. The inherent **latency** (delay to get data) in communication between subsystems should be as short as possible; this is important. Some latency examples are:

- For an SMP, the latency between CPU and memory is about 200 nanoseconds.
- For an SP2 with a high-speed switch, the hardware latency between CPUs is in a range of 40 to 200 microseconds.
- On an Ethernet LAN with no collisions, the latency is one millisecond, as can be shown by a ping.

Arbitration is needed when concurrent access to a resource occurs, such as a read or write. Managing concurrent access to resources may increase latency with locks, invalidation procedures or inherent contention. This limits scalability of the system.

1.2.2 Multiprocessor Types

There are basically four different types of multiprocessors, and this section introduces you to them briefly, for information only.

1.2.2.1 Shared-Nothing MP

A shared-nothing MP has some of the following characteristics: Each processor is a stand-alone machine. Processors share nothing; each one has its own caches, memory and disks. Also, each processor runs a copy of the operating system. Processors can be interconnected by a LAN if they are loosely coupled or interconnected by a switch if they are tightly coupled. Communication between processors is done via a message-passing library.

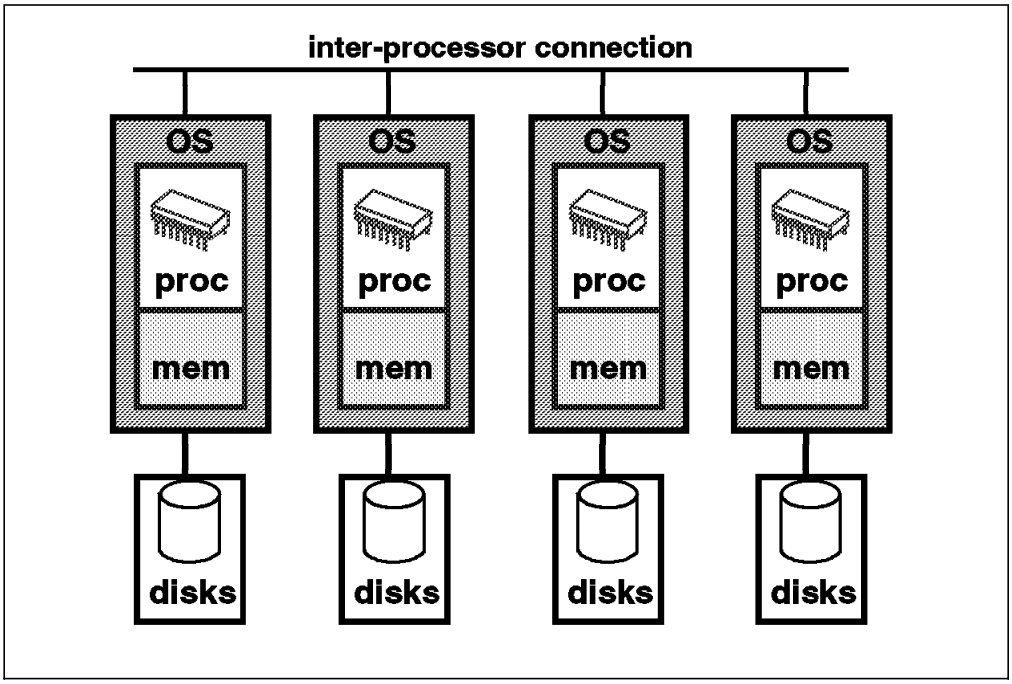


Figure 2. Shared-Nothing Multiprocessor

Examples of a shared-nothing MP are in the IBM RS/6000 SP range, as well as Tandem, Teradata and most of the *massively parallel* machines, including Thinking Machines, Intel Touchstone, Ncube and so on.

The *advantages* of a shared-nothing MP are:

- A very high scalability (up to 512 nodes on an SP2). Since processors don't share any resources, there are no contentions for accessing these resources; latency is constant, so scalability can be very high (typically several hundreds of processors). Of course, any shared resources can introduce limitations to this scalability.
- A very high availability. If one processor fails, the rest of the system can continue running without the failed processor. The application running on that processor will have to be rebooted on a different processor or a different set of processors, but the overall system will keep running.

The *disadvantages* of a shared-nothing MP are:

- It is hard to present a single system image; so system administration can be more difficult.

- If you want to take advantage of this parallel architecture, a specific programming interface needs to be used, such as a message-passing library. It is not a very familiar programming model; it requires specific skills.

1.2.2.2 Shared-Disks MP

In a shared-disks MP, each processor has its own caches and memory, but disks are shared. Also, each processor runs a copy of the operating system. Processors are interconnected through a LAN or a switch. Communication between processors may be done via message passing.

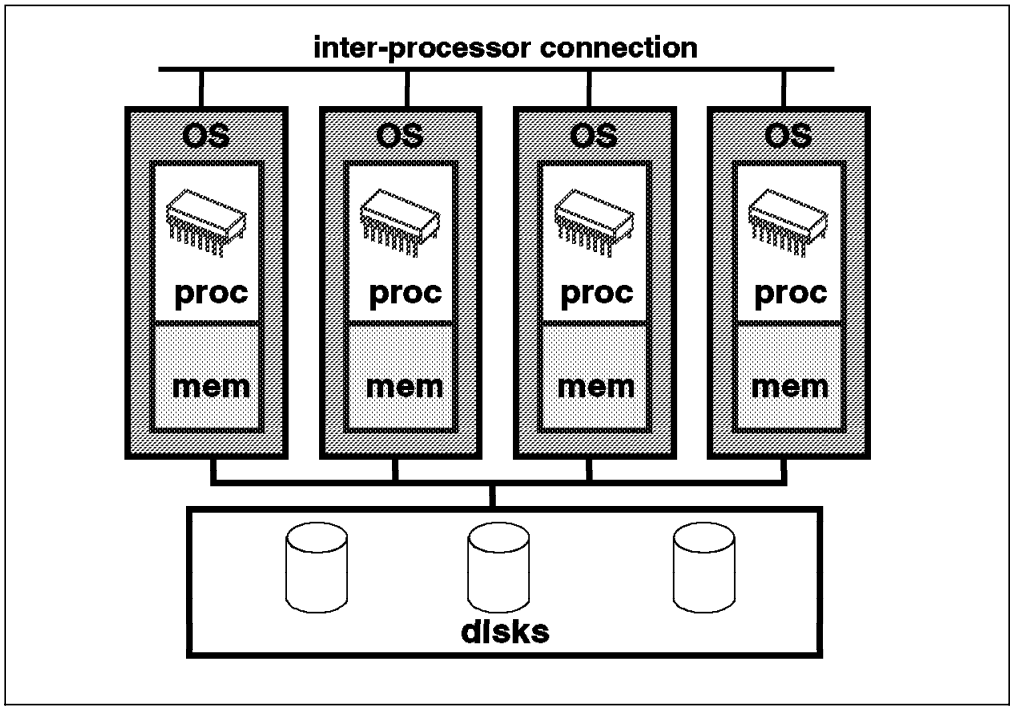


Figure 3. Shared-Disks Multiprocessor

Examples of shared-disks MPs are IBM RS/6000s with HACMP and DEC VAX-clusters.

The *advantages* of a shared-disks MP are:

- Part of a familiar programming model is preserved; data on disk is addressable and coherent, but memory is not.
- Availability is high since data on disk is still accessible by the other processors in case of a processor failure.

The *disadvantages* of a shared-disks MP are:

- Scalability is limited because of bottlenecks in the physical and logical access to shared data on disks.
- The programming model is mixed. Data is shared on disks but not on memory. Communication between the processors must be done through a specific API (Application Programming Interface). The programming model is familiar for data on disks but is unfamiliar for data on memory.

1.2.2.3 Shared-Memory Multiprocessor

In this type of MP, all of the processors are tightly coupled inside the same box with a high-speed bus or a switch between the processors, the I/O subsystem and the memory. Each processor has its own caches, but they share the same global memory, disks and I/O devices. Only one copy of the operating system runs across all of the processors. This means that the operating system itself has to be designed to exploit this type of architecture (it has to be parallelized). Note that, as this work is done for the operating system, all single-thread applications that were running on uniprocessors can, in general, run without notable changes on shared-memory MP.

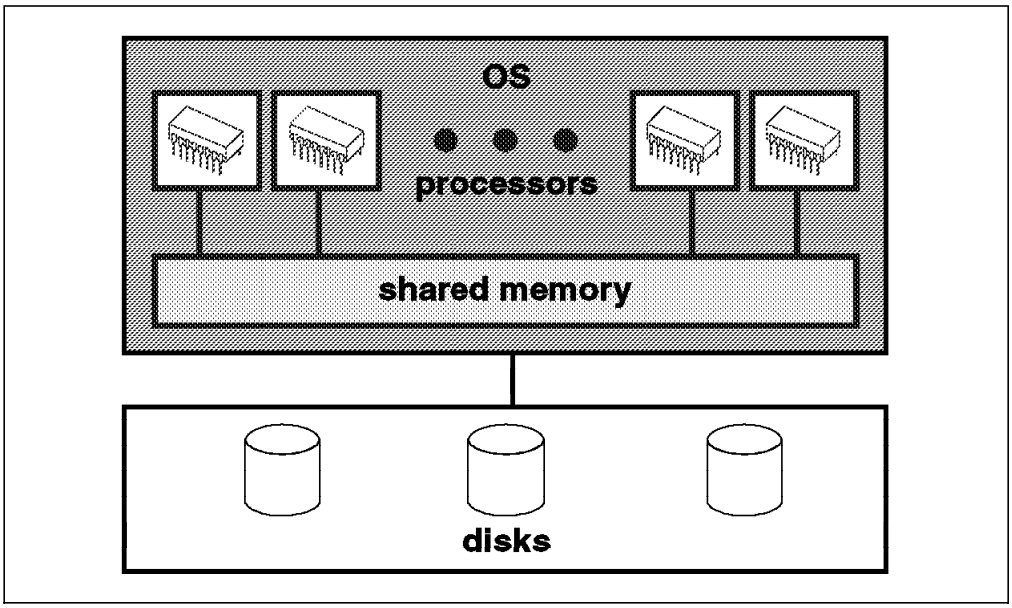


Figure 4. Shared-Memory Multiprocessor

Shared-memory MP is one of the most common multiprocessing implementations for transaction processing. Following are some examples of this model:

- IBM G40, J40, R40.
- HP T500, K 4xx series
- SUN SparcServer 1000, 2000, Ultra Enterprise 2
- DEC AlphaServer 2100, 8400
- NCR 3455 and 3525
- Sequent Systems
- SGI Power Challenge
- Compaq Proliant 4500

The *advantages* of a shared-memory MP are:

- Only one operating system is running on all of the processors. A shared-memory MP has a Single System Image (SSI); thus, system administration is easier as only one operating system must be managed.
- Programming model is familiar because UP programming model can still be used. This model can use POSIX procedures and well-known or classic

programming languages and methods. Productivity is better, and numerous skills can be found.

The *disadvantages* of a shared-memory MP are:

- It is not easy to guarantee scalability for a given application. Scalability is limited due to the sharing and contention of resources.
- In order to take full advantage of multiprocessing, parallelization is not trivial and using a thread library can be required. This programming model requires specific skills.

1.2.2.4 Non-Uniform Memory Architectures

Here are some of the characteristics of the NUMA model:

- Each group of processors (typically four) has a shared local memory.
- Each group can access all memory from remote groups as a normal address space access.
- Only one version of the operating system is needed.
- Communication between groups is done via a high-speed bus or ring.

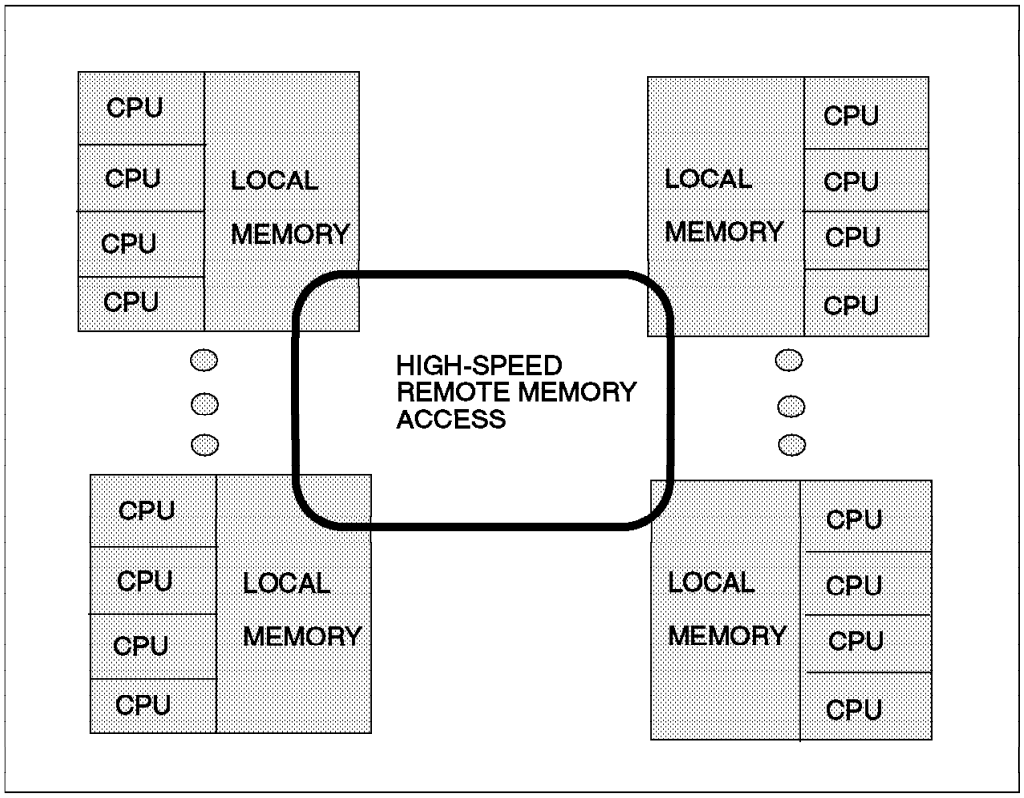


Figure 5. NUMA Architecture

This model tries to overcome scalability limitations of SMP systems. Memory is broken into a local and a remote part. When remote data is needed, cache coherency protocol loads the remote part into the local memory. This design works best when there is a minimum of data shared and being simultaneously updated. If many CPUs attempt to update the same information, many data transfers will occur, slowing overall performance.

The *advantages* of the NUMA model are:

- Better scalability than classic SMPs.
- Single operating system and standard programming model allows an easy port of applications.

The *disadvantages* of the NUMA model are:

- Optimization of data locality between nodes is critical, especially for the operating system.
- This is a new technology, untested in production environments and without significant benchmarks.

1.2.3 Symmetric versus Asymmetric Shared-Memory Multiprocessors

In an asymmetric, shared-memory MP, processors are not equal. One processor is designed as the master processor, and the others are slave processors. The master processor is a general-purpose processor which can perform I/O operations as well as computation while running the operating system. Slave processors can only perform computation. On a slave processor, all I/O operations and general kernel requests are routed to the master processor. Utilization of the slave processor might be poor if the master processor does not service slave processor requests efficiently. As an example, I/O-bound jobs may not run efficiently since only the master CPU runs the I/O operations.

Failure of the master processor is catastrophic in an asymmetric system, as the system cannot continue to run. The main reason for such an implementation is that the operating system as well as device drivers used for uniprocessor systems do not have to be rewritten and can easily be used on an MP machine. Often, this implies a limited scalability of four processors.

In a symmetric shared-memory multiprocessor (SMP), all of the processors are functionally equivalent; they all can perform kernel, I/O and computational operations. The operating system manages a pool of identical processors, any one of which may be used to control any I/O devices or refer to any storage unit. Since all processors are equal, the system can be reconfigured in the event of a processor failure and, following reboot, continue to run. Performance can be optimized, and more processors can be used.

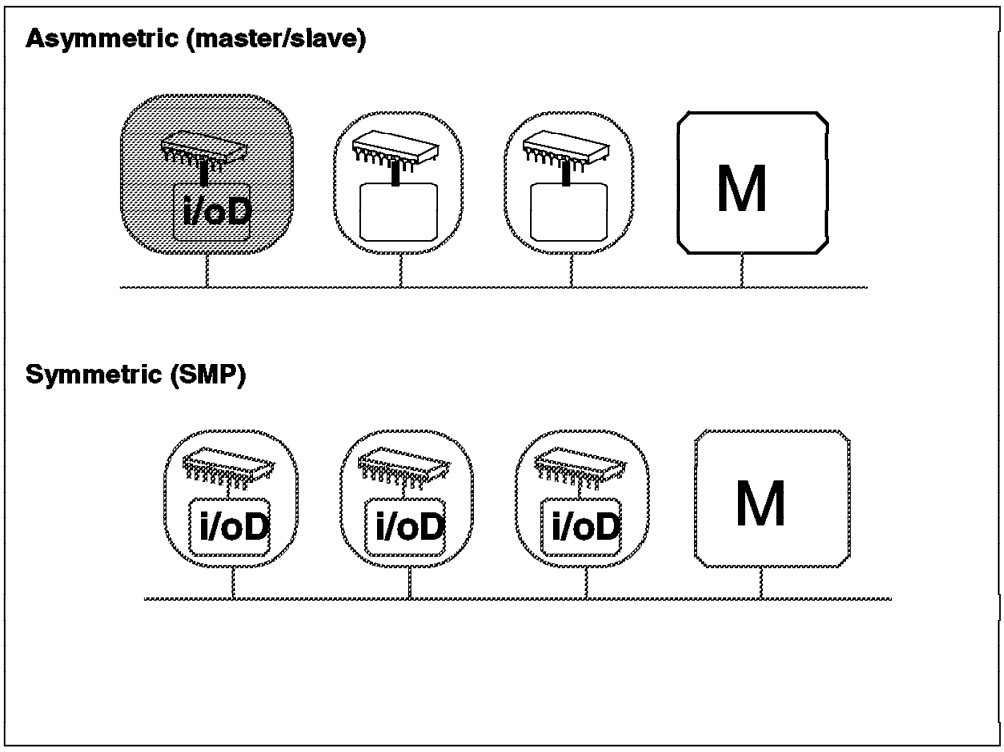


Figure 6. Symmetric versus Asymmetric Shared-Memory Multiprocessors

1.3 SMP Hardware Characteristics

Sharing resources is probably the main technical issue in the design of an SMP system. In order to support symmetric multiprocessing, specific techniques must be provided at the hardware level and at the software level. This section introduces the memory hierarchy concept and some of the techniques used to solve resource-sharing and contention issues.

1.3.1 Memory Hierarchy

In order to improve the hardware performance of a system (UP or SMP), different levels of memory are used. These different levels of memory can be ordered according to their access time and capacity.

If you look at the different types of memory available on a typical system, you will find the CPU registers at one end. They are extremely fast but very small, and they have a high cost per bit. At the other end, you will find the disks, which are very slow but have a very low cost per bit. This allows a very high disk storage capacity.

In most UP or SMP implementations, you will find between these two ends a first level of cache (L1) which is a fast memory with a small capacity. The number of CPU cycles that are needed for the processor to load data from L1 depends on L1 implementation. In the PowerPC implementation, L1 is on the CPU chip itself; so it takes only one cycle to load data from L1. When the L1 cache is outside the processor chip, several cycles are required to load data from L1. A typical L1 capacity is around 32 to 64 KB.

You might also find a second level of cache (L2) which is also a very fast memory, faster than the main memory. It takes around seven to 10 cycles to load data from L2. Its capacity is usually higher than L1 at around 1 to 4 MB.

The main memory is the third level of memory. Its access time is slow in comparison to L1 and L2 but much faster than disks. Twenty to 50 cycles are needed to load data from the memory, and the capacity can reach several gigabytes.

A cache is, in fact, a high-speed memory that holds a small subset of the main memory. Because of its short access time, it improves data locality (data is closer to the processor) and significantly increases system performance for frequently used instructions and data that are stored in the cache. When using caches, the increase in performance depends on the workload. For example, an L2 cache can be very beneficial with a commercial workload and not very beneficial with a technical workload.

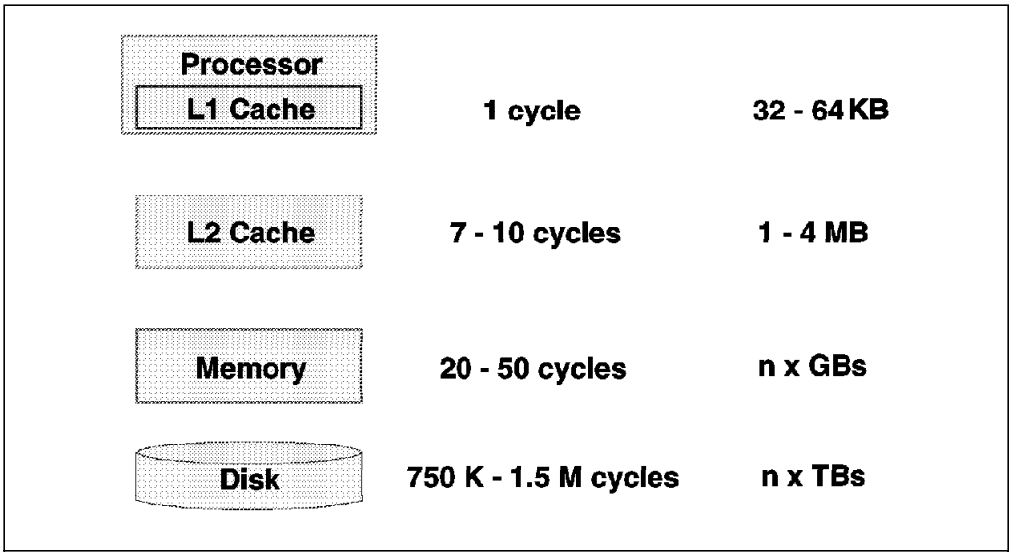


Figure 7. Memory Hierarchy

1.3.1.1 Cache Hit versus Cache Miss

When a CPU fetches a memory address, if the data is found in the cache, it is a cache hit; otherwise, it is a cache miss. If a cache miss occurs, the data is loaded from main memory to the CPU and stored in the cache to take advantage of the higher speed of the cache for a future fetch of the same memory address. The hit ratio is the percentage of cache hits. Logically, the higher the hit ratio, the better the system performance.

1.3.1.2 Cache Coherency Problem

In an SMP, all of the processors have their own cache to improve data locality. Only the main memory is shared. Since caches are not shared, it is necessary to keep all the processors' caches coherent. Cache coherency is one of the most important issues when designing an SMP system.

Consider an application that runs on two processors, processor 1 and processor 2, as shown in Figure 8 on page 10.

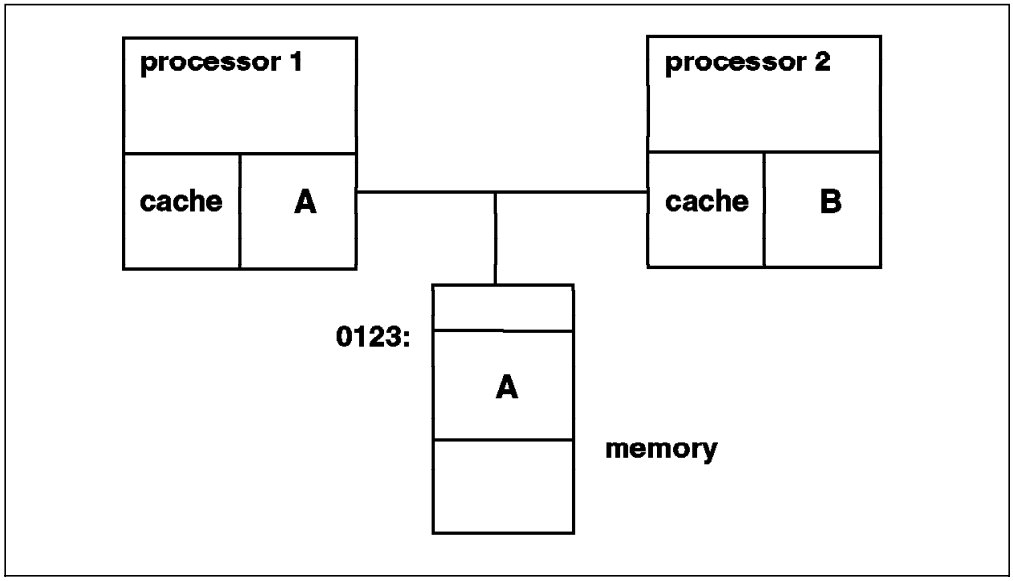


Figure 8. SMP Cache Coherency Problem

Let's assume that processor 1 loads into its cache memory address 0123, which happens to contain the character A. Then processor 2 writes B into address 0123. If processor 1 wants to again load address 0123, what will happen? In a naive implementation, processor 1 will see the value A in its cache and load that value because it does not know that processor 2 has already changed the same memory address in its cache. This is what we call the cache coherency problem.

1.3.1.3 Snooping

One solution to the cache coherency problem is snooping. Snooping is hardware logic that is added to the processor. Snooping is affiliated with normal memory reads. While a memory operation is in process, the other caches in the system are interrogated (snooped) to see if the data currently resides there. If one processor needs to write into a cache, a message is broadcast which causes that entry to be invalidated in all other caches. This is called a cross invalidate. Cross invalidate reminds the processor that the value in the cache is not valid. In this case, there is a cache miss. The processor must then look for the correct value in another cache or in the main memory.

Since cross invalidate increases cache misses and the snooping protocol adds to the bus traffic, solving the cache consistency problem reduces the performance and scalability of all SMP systems. In other words, as latency time for a request can be very variable due to the location of data and snooping activity, adding more processors is not always the better thing to do to improve response time for a given request.

All PowerPC processors except the 603 have this extra logic; the POWER and POWER2 processors do not.

Bus snooping is used to drive a MESI four-state protocol as it is described in the following section.

1.3.1.4 MESI Protocol

The unit of storage in the cache is the cache line. The size of the cache line is implementation dependent. The PowerPC has a cache line size which is 64 bytes. This cache line is divided into two 32-byte sectors.

The PowerPC maintains cache coherency on a cache sector basis by using the four-state MESI protocol. Each sector has two state bits to implement the four-state MESI protocol. The four states are:

- **M** (modified) - The addressed sector is valid in this cache only. The value in this sector has been changed in the cache, but the change is not yet reflected in memory.
- **E** (exclusive) - The addressed sector is valid in this cache only. The data is consistent with system memory.
- **S** (shared) - The addressed sector is valid in this cache and at least one other cache. It is still consistent with system memory.
- **I** (invalid) - The addressed sector is not valid in the cache.

This is best explained by means of an example. Let's assume we have two processors, processor 1 and processor 2, sharing the same memory, and let's describe different operations that can occur between these two processors.

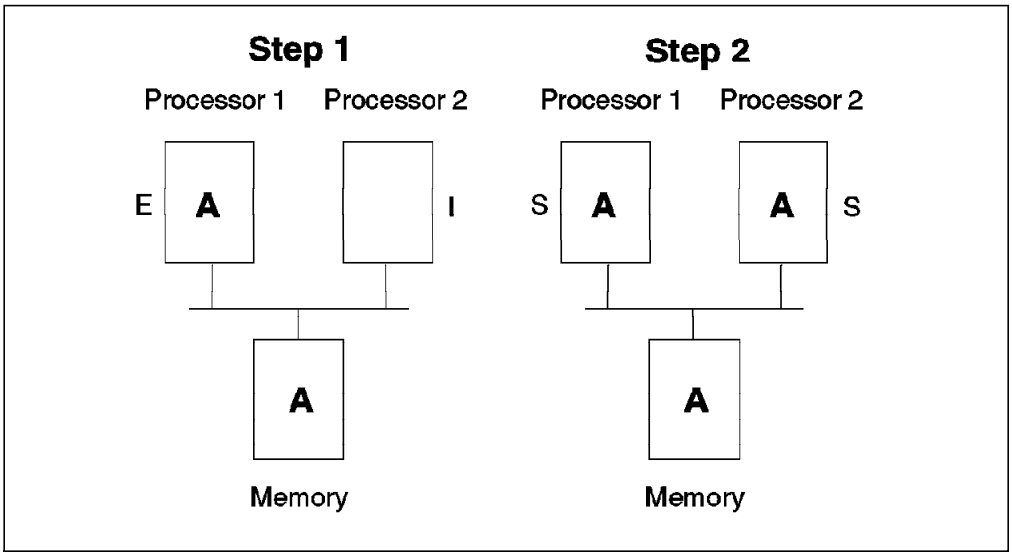


Figure 9. MESI Protocol - Steps 1 and 2

Step 1 - Processor 1 reads value A from memory and loads it in its cache. The corresponding cache sector is marked exclusive for processor 1 and invalid for processor 2.

Step 2 - Processor 2 needs the same memory address. Value A is read from memory, and the corresponding sector is marked shared in both processors.

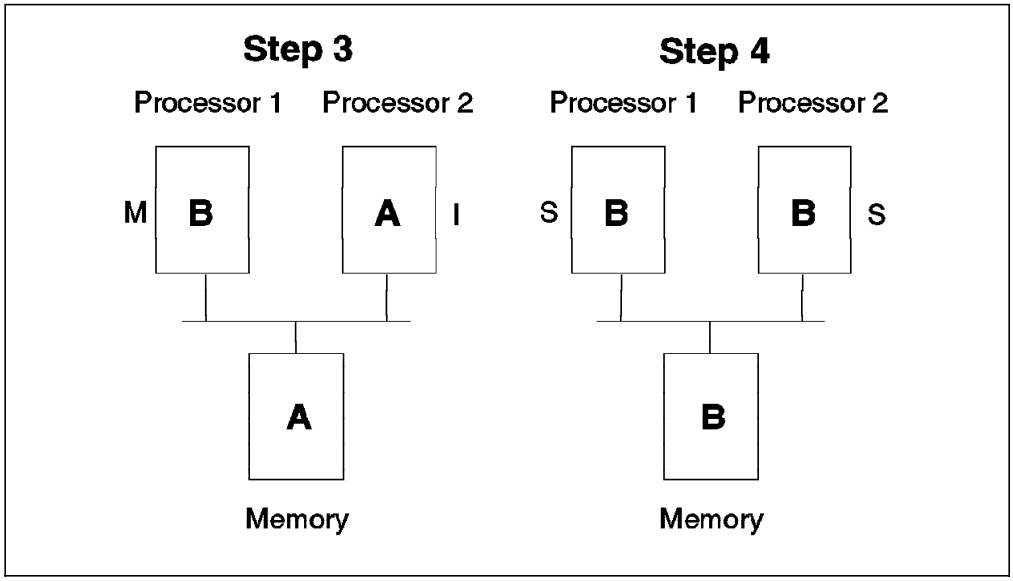


Figure 10. MESI Protocol - Steps 3 and 4

Step 3 - Processor 1 updates the address with a new value B. The snooping logic invalidates the processor 2 sector containing this address. The processor 1 cache sector is marked modified, and the processor 2 cache sector is marked invalid.

Step 4 - Processor 2 again reads the same memory address. Since processor 1 holds the modified value, processor 2 loads the value from processor 1. If the operation is a RWNITM (Read With No Intent To Modify), the memory is updated during the transaction. If the operation is RWITM (Read With Intent to Modify), the memory is not updated. It makes sense not to update the memory since the address value is going to be modified. For more information on the memory subsystem, you can refer to Chapter 3, “SMP Servers Architecture” on page 49.

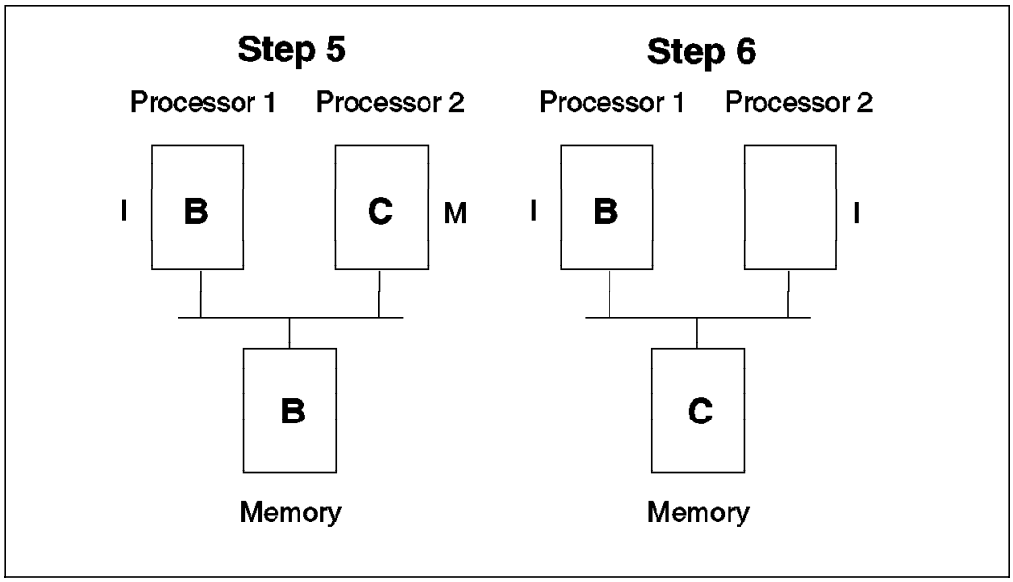


Figure 11. MESI Protocol - Steps 5 and 6

Step 5 - Processor 2 updates the same address. The snooping logic invalidates the cache sector on processor 1, and the memory address is marked invalid.

Step 6 - In this example, the memory is updated because the processor 2 cache is full. In this case, the Least Recently Used (LRU) sector is the one written back to the memory.

You can see in this memory operation example that snooping in conjunction with the MESI protocol insures cache coherency. This solves one of the main SMP design issues.

1.3.1.5 L1/L2 Caches and Store Policy

In the IBM SMP implementation, L1 has a 64-byte cache line divided into two 32-byte sectors. L2 cache has a 32-byte cache line.

The store policy defines the way L1, L2 and the memory are kept coherent or up to date. The store policy between L1, L2 and the memory is implementation dependent.

The store policy between L2 and the memory is write back. This means that when L2 is updated, the memory is not.

In fact, the memory is updated in several cases. It is updated in case of a cache migration caused by a Read With No Intent To Modify (RWNITM) from another processor. For example, when a processor loads data from another processor and does not want to change that data. Logically, if the processor wants to change that data, it is not necessary to update the memory since the fetched memory address is going to be changed.

The memory is also updated when the processor needs room in its cache. When this happens, the Least Recently Used (LRU) algorithm is used, and the LRU sector is written back to the memory. The memory is also updated in the case of a cache flush.

1.3.1.6 False Sharing

Figure 12 illustrates false sharing. In this figure, we represented a whole cache unit of storage (a cache sector in the case of the IBM SMP system).

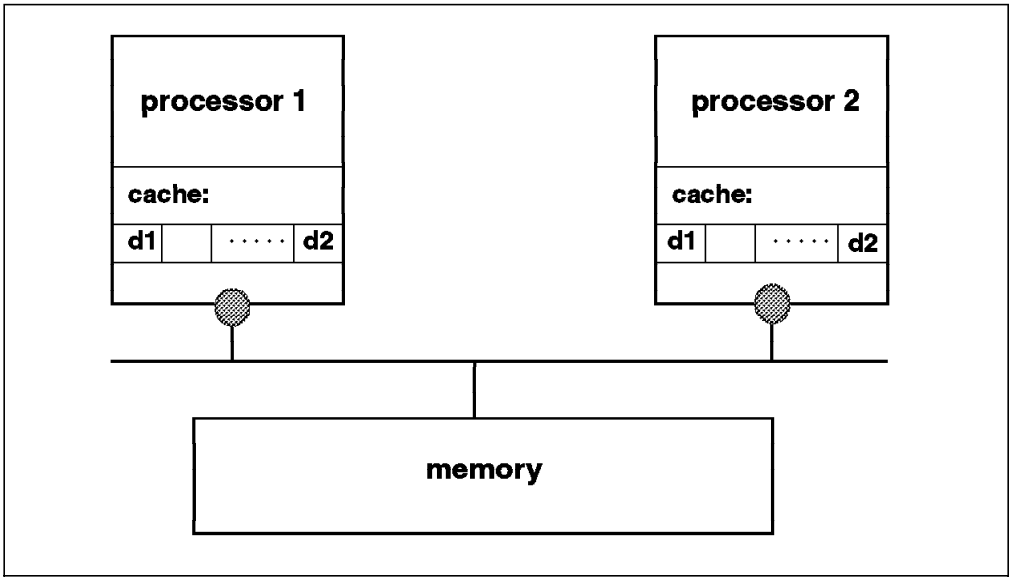


Figure 12. False Sharing

Let's suppose that processor 1 and processor 2 loaded the same memory address in their cache. If processor 1 changes only a portion of the cache sector, d1 for example, the cache consistency logic will invalidate all the sectors in the processor 2 cache. Then, if processor 2 tries to modify another portion of its cache sector, d2 for example, which is still invalid since the whole sector is invalid, a cache miss will occur. This is called false sharing.

Thus, false sharing increases cache misses and bus traffic, and this may cause the SMP throughput to be reduced. The bigger the size of the cache line, the higher the miss rate. Some implementations have a 256-byte cache line. The IBM SMP works with a 32-byte cache sector, and the coherency mechanism is based on the cache sector.

1.4 SMP Software Characteristics

Since most operating system activity is triggered by events, interrupts and system calls, all processors are able to run any part of the kernel and access any kernel data simultaneously. So, changes must be made to a UP operating system in order to support an SMP. One of the main changes is the implementation of threads. A thread is an independent flow of control within a process. All threads within a process can run concurrently on different processors. Threads are well suited for exploiting SMP architectures. Since a classical UNIX process is considered as a single-threaded process, threads will be used in this section to illustrate some concepts.

1.4.1 SMP Synchronization Issue

There is a potential synchronization problem where two processors might be trying to update the same piece of data at the same time and incorrect results can be obtained.

Consider an example where two threads are updating the same variable.

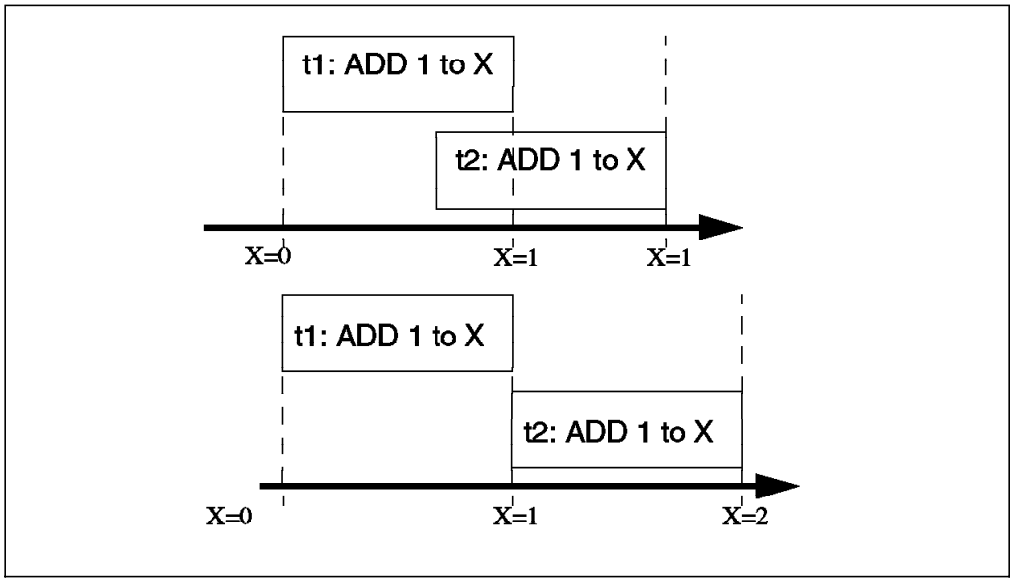


Figure 13. Synchronization Issue

In the top half of the diagram, threads t1 and t2 are both adding one to the same shared variable, X, whose value is x. The final value must be x+2, but t2 is

incrementing the variable before t1 has finished. Therefore, the final value will be x+1. In order to avoid this, both threads have to be serialized as is shown in the bottom half of the diagram.

The section of code that changes shared data and, therefore, must not be run by more than one processor at a time, is called a *critical section*. In the above example, the critical section is the code that changes the variable, X.

The problem of serializing access to shared data is generic to parallelized code. It occurs at both the user and the kernel level. This problem is resolved by locks on critical sections of code.

1.4.2 Locks

Basically, a lock is a memory location that threads use to regulate their entry into critical sections. If the location is zero, then the lock is free, if the location is non-zero, then the lock is busy. For a processor to take a lock, the simplest code sequence is:

```
test the lock
if the lock is free
then
    set it to busy
else
    wait for it to become free
end if
```

Since taking a lock requires several operations (read, test and set the lock), this operation is itself a critical section. Several threads can test the same lock simultaneously and set the same lock bit. Therefore, multiprocessor hardware must provide a way to perform this test-and-set operation atomically. The PowerPC provides special instructions that allow atomic updates of a word in memory.

- lwarx - loads a word from memory and establishes a reservation
- stwcx - stores a word if the reservation is still present

This kind of atomic operation is the basic block on which all of the locking primitives are based.

1.4.3 Lock Types

There are two types of locks we need to consider:

- Mutual Exclusion (Mutex) Locks

  This type of lock is an exclusive lock that allows one thread at a time in a critical section. Other threads have to wait (or spin), even for read-only.

- Read/Write Locks

If a piece of shared data is read mostly, it makes sense to distinguish between the many threads that only want to read the data but not change it, and a few threads that want to change/write data. This type of lock allows multiple readers to be in the critical section at once but guarantees mutual exclusion for writers. It allows one writer to hold the lock and block any others, and it is also called a Read Shared/Write Exclusive lock.

1.4.4 Waiting for Locks

When a thread wants a lock that is already owned by another thread, the thread is blocked. It has to wait until the lock becomes free, and there are two ways of waiting: spinning or sleeping.

- Spin locks - These allow the waiting thread to keep its processor by repeatedly checking the lock bit in a tight loop (spin) until the lock becomes free. Spin locks are suitable for locks that are held only for very short times.
- Sleeping locks - The thread sleeps until the lock is freed, and then it is put back into the run queue. Sleeping locks are suitable for locks that may be held for longer periods.

Waiting for locks always decreases system performance. If a spin lock is used, the processor is busy but not doing useful work. If a sleeping lock is used, the overhead of context switching and dispatching, and the consequent increase in cache misses, is incurred.

1.4.5 AIX Version 4.1 Kernel Locks

The OSF/1 1.1 locking methodology is used as a model for the AIX Version 4 multiprocessor lock functions. The OSF/1 locking model has two types of locks, simple locks and complex locks.

In AIX Version 4.1, since the system is preemptive and pageable, some characteristics have been added to the OSF/1 1.1 locking model. The AIX Version 4.1 simple locks and complex locks are preemptable. Also, a thread may sleep when it tries to acquire a busy simple lock if the owner of the lock is not currently running. In addition, a simple lock will transform when a processor has been spinning on a lock for a certain amount of time; this amount of time is a systemwide variable.

AIX Version 4.1 simple locks have the following characteristics:

- spin, exclusive, nonrecursive (any nesting must be removed), preemptable (only in process environment)
- two states: locked, unlocked

AIX Version 4.1 complex locks have the following characteristics:

- RW (Read-Shared, Write-Exclusive), sleeping, recursive on request, preemptable
- Three-states: exclusive-write, shared-read, unlocked

It is the AIX developer's responsibility to define and carry out an appropriate locking strategy to protect global data. This is also true for device driver programmers who have to write their own kernel extensions.

In order to maintain compatibility between AIX Version 3 and AIX Version 4, AIX Version 3 locks are still available. These locks, `lockl()` and `unlock()`, are exclusive, recursive and sleeping locks.

1.4.6 UP Synchronization

AIX V3.2 is preemptable. A process can be preempted by a higher-priority process or by an interrupt routine. The interrupt routine or higher-priority process may access and change data that is already being changed by the original process. Thus, in AIX Version 3.2, as shown in Figure 14, synchronization must be provided between processes themselves and between processes and interrupt handlers.

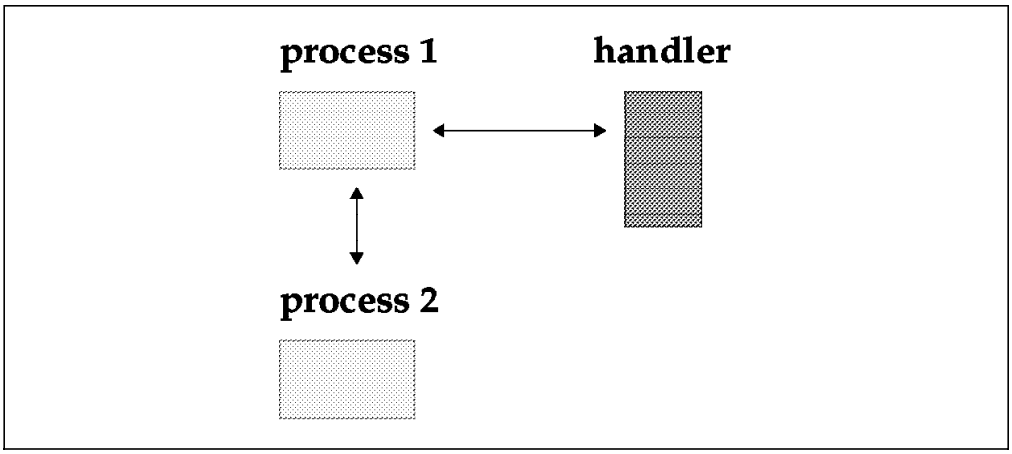


Figure 14. UP Synchronization

In order to avoid data corruption between several processes, AIX V3.2 provides a lock mechanism that allows serialized access to shared data, (`lockl()` and `unlockl()`). These locks are used at the process level only.

Interrupt handlers are not allowed to acquire locks. When a process needs to accomplish an atomic operation without being interrupted by an interrupt routine, it must disable the interrupts by using the `i_disable()` primitive. The `i_disable()` primitive is only effective on the processor on which the process is running. The `i_enable()` primitive will again enable interrupts.

1.4.7 SMP Synchronization

In an SMP, synchronization must be achieved between threads themselves, between threads and interrupts or between interrupts themselves. Figure 15 on page 18 illustrates this relationship.

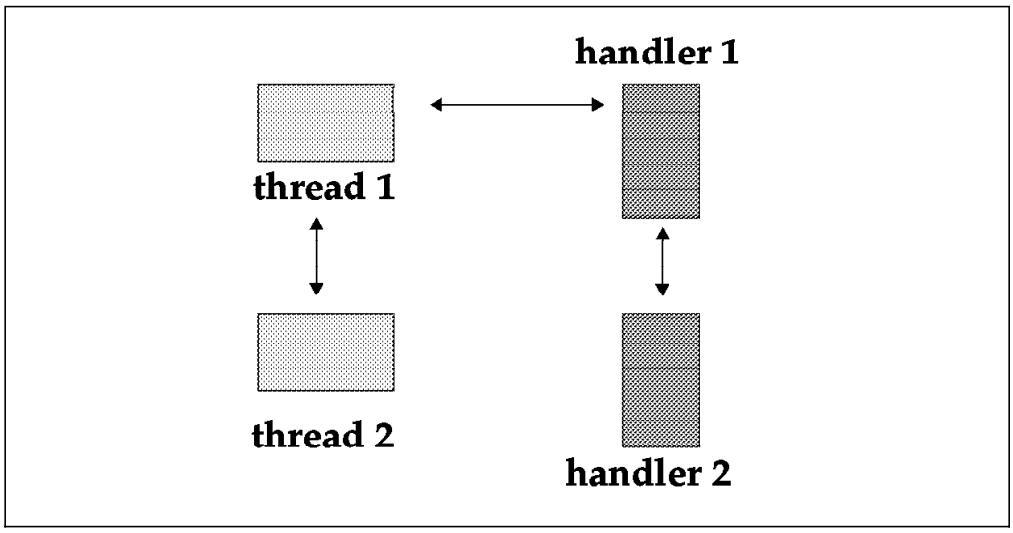


Figure 15. SMP Synchronization

Masking interrupts to serialize with an interrupt handler no longer works in an SMP environment. Since the I/O is symmetric and the interrupts can be routed to any of the processors in the system, masking interrupts will not prevent the interrupt routine from simultaneous operation on another processor. Therefore, AIX V4.1 provides new primitives to achieve serialization with an interrupt handler. Serialization requires the use of locks in addition to the traditional masking of an interrupt. New kernel services, `disable_lock()` and `unlock_enable()`, combine disablement and locking in the correct order.

In summary, there are three types of critical sections that must be protected from concurrent operation in order to serialize access to a resource in an SMP:

- thread-thread -> This critical section must be protected from concurrent operation by multiple threads by using AIX V4.1 simple or complex locks.
- thread-interrupt -> This critical section must be protected from concurrent operation by an interrupt handler and a thread by using the `disable_lock()` (to lock) and `unlock_enable()` (to unlock) kernel services. Simple locks must be used.
- interrupt-interrupt -> This critical section must be protected from concurrent operation by multiple interrupt handlers using `disable_lock()` (to lock) and `unlock_enable()` (to unlock) kernel services. Again, simple locks must be used.

In order to minimize the impact on system performance due to SMP locking, all interrupt-interrupt and thread-interrupt critical sections should never directly use the simple lock primitives. They should instead use the new kernel services, `disable_lock()` and `unlock_enable()` primitives, since, in this kind of critical section, disabling is always needed. These two primitives have been optimized.

1.4.8 AIX V4.1 Kernel Locking Interface

Following is a subset of the kernel locking interface used by AIX developers or device driver developers.

- Lock allocation services
  - lock_alloc(): allocates system memory for a simple or complex lock
  - lock_free(): frees the system memory of a simple or complex lock

  - lock_mine(): checks whether a simple or complex lock is owned by the caller

- Simple locks services
  - simple_lock_init(): initializes a simple lock
  - simple_lock(): issues a simple lock
  - simple_lock_try(): issues the lock if the lock is free; does not block and returns immediately if the lock is busy
  - simple_unlock(): unlocks a simple lock
  - disable_lock(): raises the interrupt priority and locks a simple lock
  - unlock_enable(): unlocks a simple lock and restores the interrupt priority
- Complex Locks
  - lock_init(): initializes a complex lock
  - lock_islocked(): tests whether a complex lock is locked
  - lock_read_to_write(): upgrades a complex lock from shared-read mode to exclusive write mode
  - lock_write(): issues a complex lock in exclusive-write mode
  - lock_write_to_read(): downgrades a complex lock from exclusive-write mode to shared-read mode
  - lock_set_recursive(): prepares a complex lock for recursive use
  - lock_clear_recursive(): prevents a complex lock from being acquired recursively

1.4.9 AIX V4.1 Lock Services Summary

In the previous sections, we covered the locking mechanisms that are required at the kernel level between kernel threads. Serialization to shared resources is not only an issue for kernel threads. It is also an issue for user threads or user processes.

For user threads and user processes, serialization to shared resources is also done by means of locks. Different Application Programming Interfaces (APIs) are provided by the system to lock shared resources.

Serialization between user threads can be done by using the pthread library subroutines, `pthread_mutex_lock()` and `pthread_mutex_unlock()`. For more information on threads, you can refer to Chapter 2, “Introduction to AIX V4 Threads” on page 31.

Serialization between user processes can be achieved by using the `msem_lock()` and `msem_unlock()` subroutines that are part of the Base Operating System. These subroutines allow you to lock or unlock a semaphore.

1.4.10 Lock Penalty

There is no such thing as a free lunch, and locking is no exception to this rule.

Suppose that we know from `tprof` that when running a certain application, the system spends 10 percent of its time in a kernel component. Suppose that the component is complex and touches a lot of data; so the developer decides to make the whole component one big critical section. That is, there is only one mutex lock for the whole component, and it is requested at all entry points in the component and released at all exit points. On a four-way SMP, this mutex lock will be busy 4 x 10 percent = 40 percent of the time.

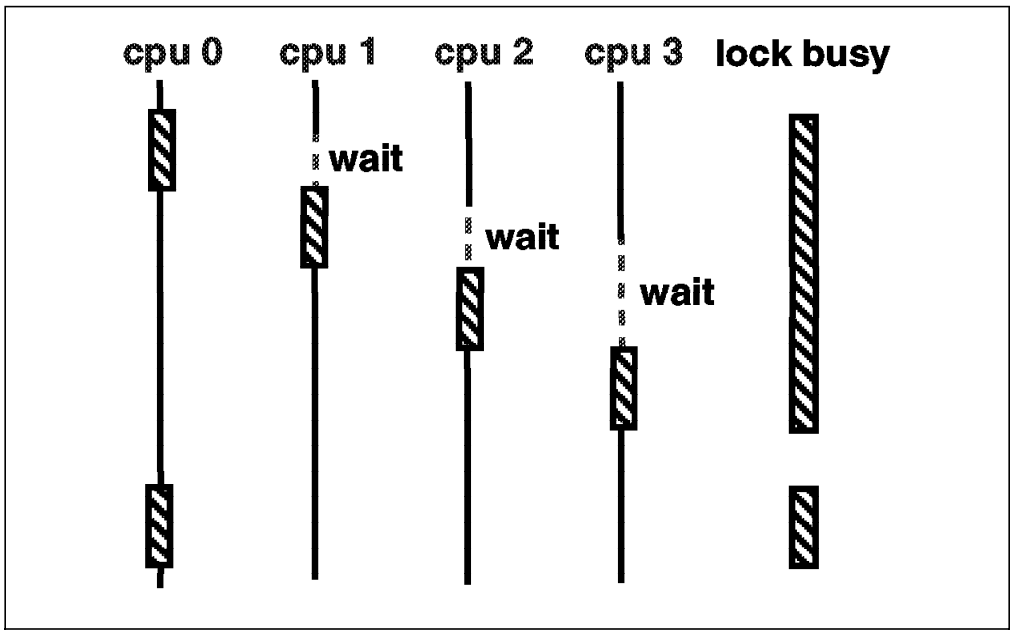


Figure 16. Lock Penalty

According to queueing theory: the busier a resource, the longer the average wait to get it, and the relationship is nonlinear. If the use of the lock is doubled, the average wait time for that lock more than doubles.

1.4.11 Lock Granularity

The most effective way to reduce wait time for a lock is to reduce the size of what the lock is protecting. In other words, reducing the lock protection time reduces the waiting time.

Figure 17 illustrates lock granularity. Instead of locking the whole code routine, it is better to lock only the portions of code within the routine that actually modify shared data.

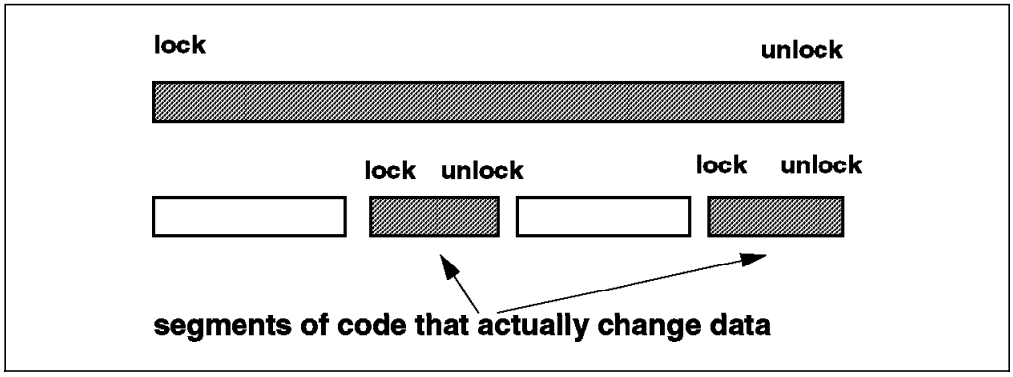


Figure 17. Lock Granularity

Here are some rules:

- The frequency with which any lock is requested should be reduced.
- Lock just the code that accesses the shared data, not all the code in a component (this will reduce the lock holding time).

- Locks should always be associated with specific data items or structures, not with routines.
- For large data structures, choose one lock for each element of the structure rather than one lock for the whole structure.

On the other hand, with granularity too fine, the frequency of lock requests and lock releases will increase. This will therefore add additional instructions. A balance must be found between a too-fine and a too-coarse granularity. This is again the developer's responsibility.

1.4.12 MP-safe versus MP-efficient

A program can be described as being MP-safe if:

- Each critical section is correctly protected with a lock.
- All acquired locks are released when done.
- Data is never changed while holding a read lock.
- Deadlocks are avoided.

The program can run on any processor without any data integrity problems, but it doesn't mean that it has been optimized for running on an SMP.

An MP-efficient program is an MP-safe program that spends the minimum time dealing with locks. That means that it has been specifically designed to run on an MP.

High-throughput device drivers, such as disks and communication drivers in AIX V4.1, are MP-efficient.

1.4.13 Processor Affinity

In AIX V4.1, the schedulable entity is the thread, and the thread with the highest priority is the one that is dispatched. This means that a thread is bounced from one processor to another. As a result, it suffers many cache misses when reloading instructions and data on the processor where the thread is dispatched.

If we try to run the thread on the processor it last ran, some of the instructions and data might still be in the processor cache. This technique may reduce the amount of cache misses and improve performance.

Affinity with a processor is the amount of data that is already in the processor cache. Processor affinity is the policy of trying to run a thread on the same processor where it last ran. AIX V4.1 has been changed to enforce affinity with the processors; so affinity is done implicitly by the operating system.

This is accomplished as follows: As shown in Figure 18 on page 22, run queues are ordered according to their priority, with 127 being the lowest and zero being the highest. When a thread is dispatched from a queue on a processor, the identity of the processor is registered in the structure of the thread.

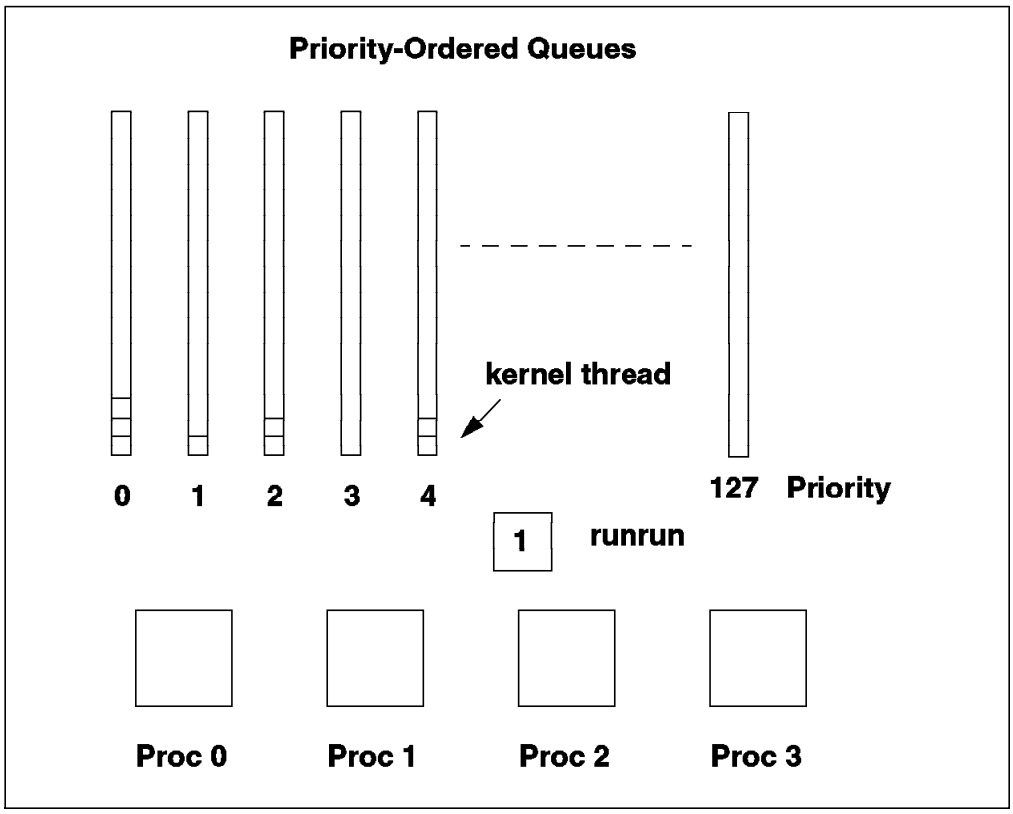


Figure 18. Threads Dispatching

In this way, each time the dispatcher selects a thread, it knows the processor number on which the thread last ran. When a processor asks to run a thread, the dispatcher chooses the thread with the highest priority from the priority-ordered run queues. It then tests to see if this thread has affinity with the processor.

If it has affinity, the thread is dispatched to the processor. If it does not, the dispatcher tries to find another thread which last ran on the processor; so it scans the queues until it finds one. This scanning is not done indefinitely; it has some limits.

1. If the priority difference between the thread with the highest priority and the thread that last ran on the processor is greater than a threshold value, the thread with the highest priority will be chosen. That threshold value is 0 by default. This means that the search for a thread with affinity with the processor is limited to the same queue.
2. Scanning is stopped when the number of scanned threads is higher than a predefined value. By default, that value is three times the number of processors (for example, 12 on a four-way SMP).
3. Scanning is also stopped when the dispatcher encounters a boosted thread and the parameter `affinity_skipboosted` is `FALSE`.

   A boosted thread can be described as follows: When a thread with a low priority holds a lock, and if a higher priority thread is waiting for the same lock, the low-priority thread gets the priority of the higher-priority thread. This means it is *boosted*. This priority inversion is automatically done by the system and is a way to reduce lock contention.

1.4.14 Binding

Binding is the strongest form of processor affinity, it may be obtained by using the `bindprocessor` command or the `bindprocessor()` system call.

The `bindprocessor` command allows a user to bind a process to a specific processor. You cannot bind a process until the process is already running, so it must exist to be able to bind it.

Once a process is bound to a specific processor, it cannot run on an idle processor and take advantage of it. Therefore, binding a process may cause some performance problems by letting some of the processors remain idle.

The `bindprocessor()` call allows a developer to bind a thread to a specific processor at the programming level.

1.4.15 Processor Numbering

If a processor has failed or is disabled, the software must be able to run on available processors. Thus, the processor number that is used by the software must not be tied to a physical processor. This is why processors on a system are identified by using either physical numbers or logical numbers. The physical numbers are in the Object Data Manager (ODM) and identify the actual processors on the system regardless of their state. Numbering starts with zero. For example, proc0, proc1 and so on.

Logical numbers identify only the enabled processors. Generally, the software uses the logical numbers. Processor states are enabled, disabled or unknown; unknown indicates a hardware failure has been detected by the system.

The following table illustrates the naming schemes for a four-processor system with different processors in different states.

ODM name	Physical number	Logical number	Processor state
proc0	0	0	Enabled
proc1	1		Disabled
proc2	2		Unknown
proc3	3	1	Enabled

Table 1. Physical and Logical Processor Numbering

Generally, all operating system commands and library subroutines use logical numbers to identify processors. The `cpu_state` command is an exception because it uses the physical processor numbers. This command can be used to list system processors and their states, it can also be used to enable or disable processors (a reboot is required for the change to take effect).

1.4.16 UP Application Compatibility

A UP application which is a single-threaded process (more than 80 percent of UP applications have been written this way) can run on an SMP if it already runs on AIX Version 4. This application won't take advantage of the multiprocessing capabilities since it will run on only one processor at a time. Also, the application will run slightly slower than on the equivalent uniprocessor because of some overhead caused by the MP kernel.

If the application is multiprocessed, the application must be ported to the SMP environment to be at least MP-safe. Serialization is natural on a UP, but running

the same application on an SMP may cause some data integrity problems since two processes running on two different processors may change the same data at the same time. Locks must be used so that the application can run safely on the SMP.

If the application is already multithreaded, porting is still necessary. The application may use a different thread library or a different level of the thread library than the one which is available on AIX Version 4. Also, all critical sections must be identified and protected by locks.

Both the multiprocessed and the multithreaded application will take advantage of the multiprocessing, but in general, a multithreaded application will perform better than a multiprocessed application.

1.4.17 UP Device Drivers Compatibility

Uniprocessor device drivers must be ported to the SMP environment. In order to run UP device drivers unchanged on an SMP, we introduced the notion of master processor and funneling. These two notions are briefly explained below.

1.4.17.1 Master Processor

The concept of master processor does not have the same meaning here as in the master/slave scheme used for an asymmetric multiprocessor. In order to run UP device drivers unchanged on the SMP, UP device drivers must run on a specific processor which is called the master processor. We say that their execution has to be *funneled* to the master processor.

The master processor is defined by the value of MP_MASTER in the /usr/include/sys/processor.h file and is 0 by default. The master processor is not tied to a physical processor; therefore, the system can be started even if a processor has failed. The MP_MASTER value is a processor logical number and is assigned at boot time.

1.4.17.2 Funneling

On a UP system, a device driver may disable interrupts using specific functions. On an SMP system, since an interrupt can run on any of the processors, the UP way of disabling interrupts is no longer sufficient. In order to run a UP device driver on an SMP system, all interrupts for that device driver must be routed to the master processor. Also, the device driver code itself must run on the master processor. It must be funneled.

Therefore, all device drivers that do not tell the kernel that they are MP-safe are funneled. For an MP-safe device driver, a specific flag (DEV_MPSAFE) must be specified by the device driver in order to be considered as MP-safe by the kernel and run on any processor.

Funneling is intended to support third-party device drivers and low-throughput device drivers, such as the diskette drive.

1.4.18 PowerPC Specifics

The PowerPC processor defines a *weakly ordered memory* model which allows optimized use of the system memory bandwidth. Load and store operations are not necessarily done in the order of the code. The hardware can reorder load and store operations. While this allows optimization of the memory bandwidth for a UP system, it might be an issue for the SMP. Because of its weakly

ordered memory model, the PowerPC can allow access to the application's critical data before obtaining the lock and also release the lock before the changed data is visible to the other processors. To keep memory consistent, load and store operations can be forced to run in strict order. Specific functions are therefore provided in AIX V4.1 in order to support the PowerPC weakly ordered memory; they are `_check_lock()` and `_clear_lock()`. These functions must be used at the user-process level instead of the `cs()` (compare and swap) function. People who are porting UP applications to the SMP environment should be aware of that difference.

Out-of-order Execution is part of the PowerPC architecture. It means that instructions are not necessarily run in the order of the code. For instance, an instruction that resides in the instruction cache can be sent to an idle execution unit for execution even though it is not the next sequential instruction. This model allows an optimized use of the instructions parallelism. The PowerPC architecture also prevents out-of-order execution of instructions that depend on the result of previous instructions.

1.5 SMP Scaling

Scaling is one of the most important metrics of MP performance, and it relates to how much the performance increases as processors are added.

In a perfect world, one would expect performance to increase linearly as processors are added. However, this is not the case because of the overhead required to maintain a consistent view of the memory and the other shared resources for each of the processors in the system. As more processors are added, each additional processor increases performance by slightly less than the previously added processor. However, adding more processors ceases to boost performance after some critical number, as the following diagram shows.

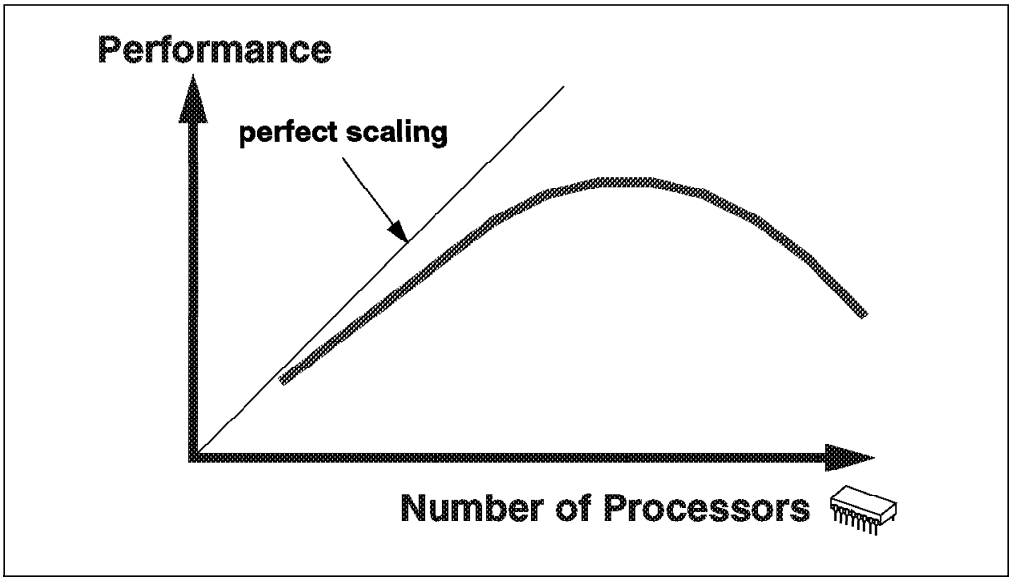


Figure 19. Scaling

There are many reasons why real workloads do not scale perfectly on an SMP system, and some of them are listed below.

- Increased bus/switch contention when the number of processors increases.

- Increased memory contention because all the memory is shared by all the processors.
- Increased cache misses because of larger operating system and application data structures.
- Cache cross-invalidates and lateral reads to maintain cache coherency.
- Increased cache misses because of higher dispatching rates.
- Increased cost of synchronization instructions.
- Increased operating system and application pathlengths for lock/unlock.
- Increased operating system and application pathlengths waiting for locks.

It can be seen from some of the above factors that scaling is workload dependent. Some workloads may scale relatively well on an SMP while others will scale poorly.

For example, the SPECrate workload scales very well on an SMP because it is made of several independent programs that do not interact with each other. In this case, contentions due to resource sharing are quite low.

On the other hand, a commercial workload, like the TPC-C, will scale worse than the SPECrate. It is difficult to scale well on such a workload. These relationships are shown in Figure 20.

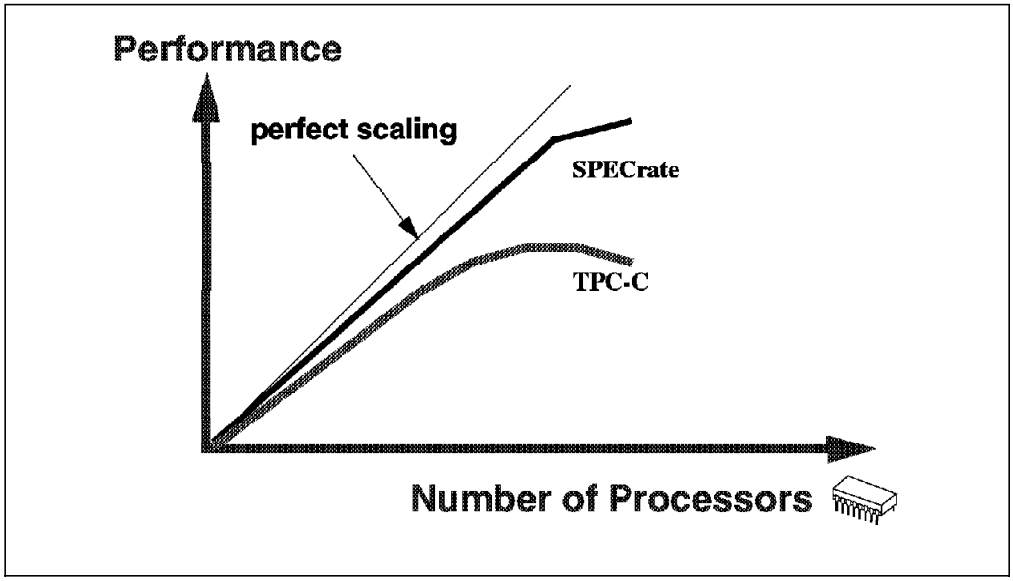


Figure 20. Scaling is Workload Dependent

1.5.1 Scaling Metrics

There is no universally accepted metric for scaling. In general, a one-way SMP will run slower (about 10 to 15 percent) than an equivalent processor running a UP version of the operating system. This occurs because of the MP overhead that is inherent in the kernel of the MP operating system. So, as a result, most vendors will show scaling starting from two processors.

The following table is a hypothetical representation of how scaling can be represented. However, getting a ratio to one processor greater than 2.5 with four processors on the TPC-C benchmark is a good result.

Number of Processors	Value of Hypothetical Performance Test	Ratio to 1 Processor	Ratio to Number of Processors
1	100		
2	170	1.7	0.85
4	300	3.0	0.75
6	430	4.3	0.72
8	560	5.6	0.70

Table 2. Hypothetical SMP Scaling Metrics

1.5.2 Two-Dimensional Scaling

Most vendors can scale in one direction only, by adding more processors.

The IBM RS/6000 SMP servers allow two-dimensional scaling by being able to utilize higher performance processors as well as by increasing the number of processors that can be added. These SMPs have been designed to allow for three generations of PowerPC chips to be included in the system (601, 604 and 620) and to support up to eight PowerPC processors. The memory subsystem has been over-designed to cater for that growth.

1.6 Using an SMP

In order to effectively use an SMP, there are a number of things that need to be considered, such as parallelizing the application, Amdahl′s Law and assessing the applicability of commercial applications against technical applications. These are discussed in more detail in the following sections.

1.6.1 Parallelizing an Application

Parallelizing an application is one opportunity to effectively use an SMP. There are two ways to achieve this:

1. The traditional way is to break the application into multiple processes. These processes communicate using Inter-Process Communication (IPC), such as pipes, semaphores or shared memory. The processes must be able to block events, such as messages, from other processes, and they must coordinate access to shared objects with something such as locks.
2. The other way is to use the POSIX threads. Threads have coordination problems similar to those in processes and similar mechanisms to deal with these problems. Thus, a single process can have any number of its threads running simultaneously on different processors. Coordinating them and serializing access to shared data is the developer′s responsibility.

Threads and processes each have advantages and disadvantages to be considered when determining which method to use for parallelizing an application. Threads may be faster than processes, and memory sharing is easier, but a process implementation will distribute more easily to multiple machines or clusters. Most RDBMS vendors use a “multithreading” that is their own implementation of threading and not true kernel threading. This allows an easier port for several platforms. But it is likely that, in the future, standard POSIX threading will be used.

1.6.2 Amdahl′s Law

Amdahl′s Law quantifies the fact that if only a part of the program speed is increased, the part that was not increased still runs as slowly as before.

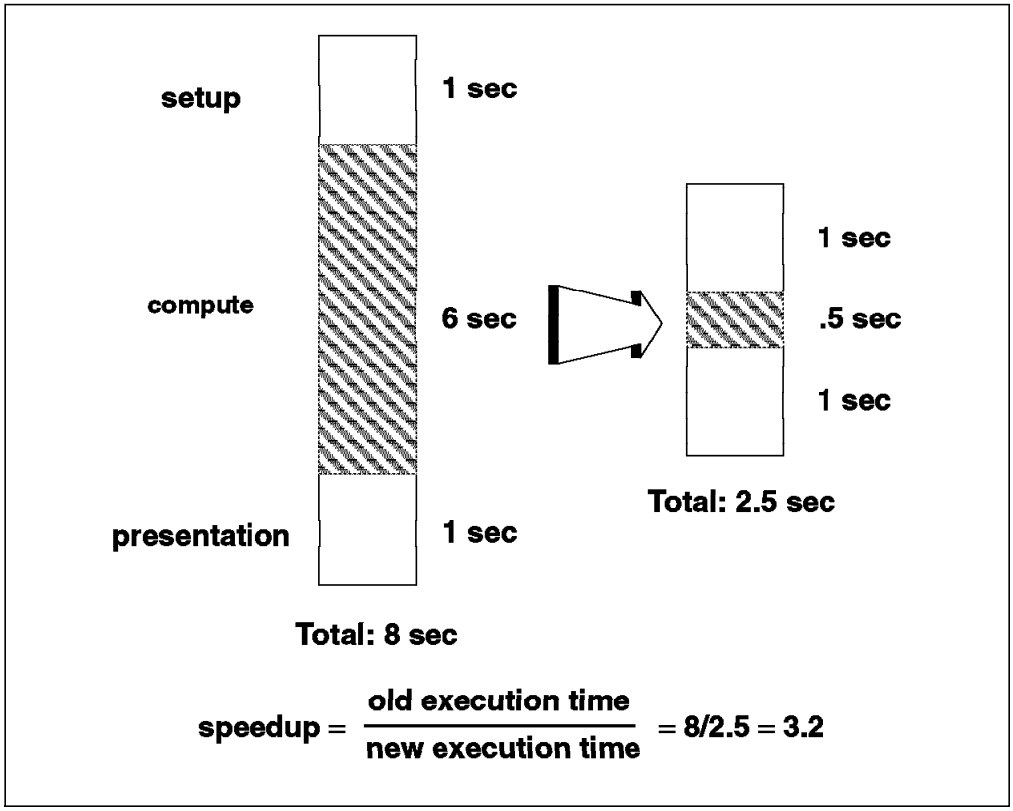


Figure 21. Amdahl′s Law

In the above diagram, making the main part of the program run 12 times faster sounds good, but it only makes the program run 3.2 times faster. There is an upper limit on the increased speed that can be achieved by parallelizing the compute phase of this application.

1.6.3 Commercial versus Technical Applications

Commercial applications have a number of characteristics. They use a large amount of data shared between many different users or programs. They have a low data locality, which means that there is a high level of data traffic between system memory and CPU caches, and there is a high level of I/O activity. There is also a high level of data traffic between caches due to lateral process migration. Therefore, a commercial application needs big L2 caches and very high bandwidth between memory and CPU as well as between the CPUs themselves.

Technical applications are usually CPU bound, and so the processor's speed is key. Code is often made with short loops of instructions and may fit in the L1 cache.

1.7 SMP Summary

SMPs have a number of benefits as follows:

- SMPs are a cost-effective way to increase throughput.
- SMPs offer a single system image, since the operating system is shared between the processors (administration is easy).
- You can apply multiple processors to a single large problem by using parallel programming.
- Load balancing is done by the operating system.
- The same UP processing model can be used in an SMP.
- There is easy and transparent scalability in two dimensions within the limitations of scalability.
- More and more applications and tools are available today, and most UP applications can run on, or are being ported to, an SMP.

There are some limitations to using SMPs:

- They require MP-enabled processors. POWER, POWER2 and PowerPC 603 cannot be used in an IBM SMP.
- Scaling is not linear, and there is a finite limitation in the number of processors.
- New skills are required because of new programming concepts (threads).
- Complex programming is required for device drivers since they have to manage multiple interrupt occurrences at the same time.

Chapter 2. Introduction to AIX V4 Threads

This chapter discusses the AIX V4 threads implementation. It also gives some considerations on programming a multithreaded application using the standard pthreads library.

2.1 What is a Thread?

A thread is an independent flow of control that operates within the same address space as other independent flows of control within a process. In previous versions of AIX and in most UNIX systems, processes could only have one flow of control within the same address space. In AIX Version 4, one process can have multiple threads, with each thread executing different code concurrently, while sharing data and synchronizing each other. In such a multithreaded system, a regular process is seen as a single-threaded process.

2.2 Threads versus Processes

This section discusses the differences between processes and threads.

2.2.1 Processes

A process is a combination of a program (set of instructions and data needed to perform a specific task) plus the current state of its execution, the values of all variables, and the hardware state such as the program counter, registers, condition codes and the content of the address space. In summary, a process is a program in execution.

A process address space consists of four main pieces: program instructions, initialized data, uninitialized data and the stack. In UNIX jargon, the instructions are called the ″text″ segment (it is the code of the process). The initialized data is simply called the ″data″. The uninitialized data is called ″bss″, which takes its name from an old assembler mnemonic called ″Block Started by Symbol″. The stack is simply called the ″stack″. The difference between the initialized data and the uninitialized data is that the initialized data is the global and static program variables that were declared to have an initial value when the program was compiled. The uninitialized data is the program global and static variables that had no explicit initial value. For these variables, the system simply allocates memory in the address space that initially contains zeros. The advantage of this approach is that the uninitialized data need not take up space in the program file.

A conventional UNIX process has only one flow of control. In this model, since there is only one thread in the process address space, no inner scheduling is needed and no inner data communication is necessary. Also, whenever the process runs, its unique thread runs. Any data communication in the memory between processes must be done using the inter-process communication (IPC) mechanisms, such as pipes, semaphores, message queues and so on.

2.2.2 Multithreaded Processes

A multithreaded process has several independent flows of control. All the threads within a process run in the same process address space. Each thread holds the state of a single flow of execution within the process. The state of a thread consists of a minimum of the hardware state and a stack. Also, each thread has its own kernel thread data. Since threads run in the same process address space, data communication between these threads can easily be achieved through shared variables within the process address space. Inter-threads communication does not require the use of the IPC mechanisms.

A multithreaded process starts out with one stream of instructions called the initial thread. Then the process may create other instruction streams (threads) to run several tasks. At some point, it is very similar to one process forking another process, but when a process forks, there is a hierarchical relationship between the parent process and the child process. This is not the case with threads. All of the threads created under a process through the initial thread are peers. There is no hierarchy between threads within a process.

From a programming point of view, facilities are provided to the programmer to do many of the same things you can do with processes. These facilities include calls for threads creation, termination, synchronization, communication, error recovery and management. Thus, the programmer has control over how the threads behave and what services they request of the system within the process. When a thread is running in user space, it can make system calls just like a simple single-threaded process. Figure 22 illustrates the differences between a traditional UNIX process and a multithreaded process.

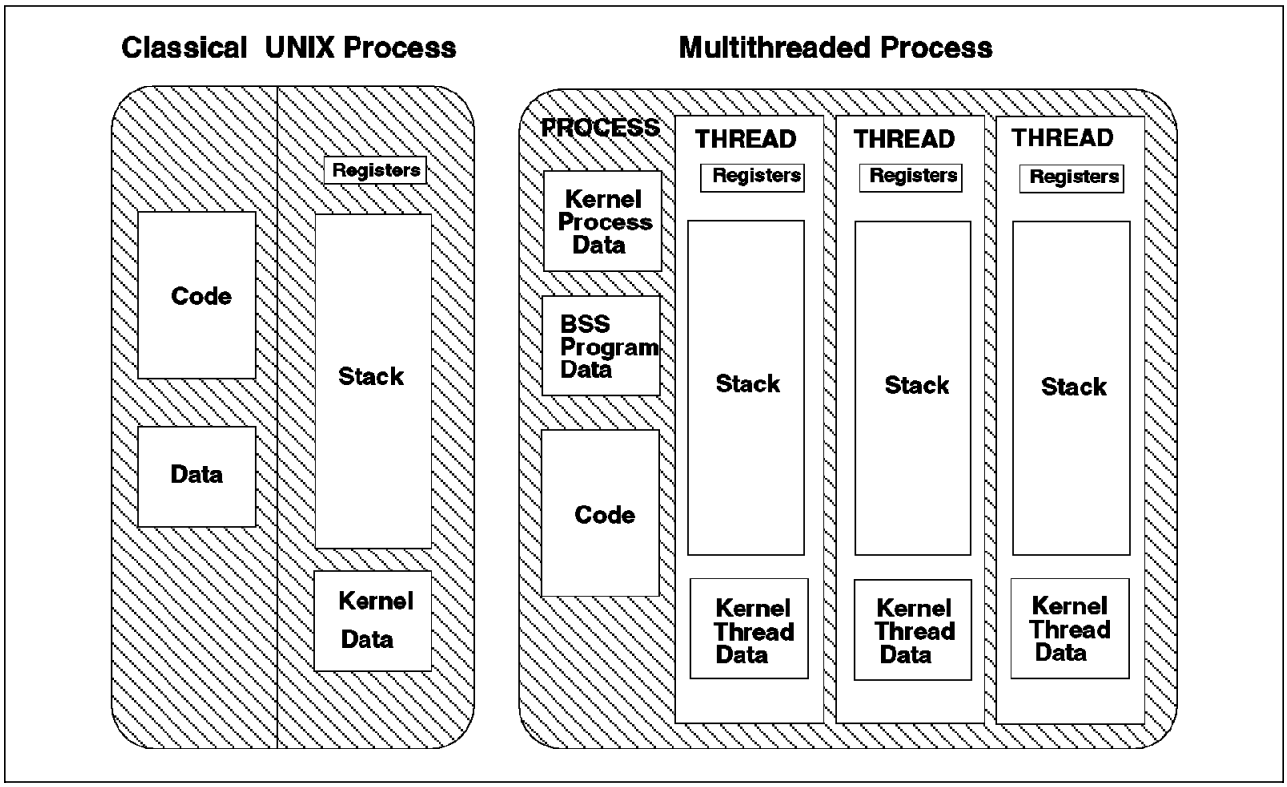


Figure 22. Multithreaded Process

2.2.3 The Initial Thread

We said previously that when a process is created, one thread is automatically created. This thread is called the initial thread, and it is the thread used for the execution of an otherwise non-threaded (single-threaded) process. The initial thread has some special properties that ensure binary compatibility between the single-threaded process and the multithreaded operating system. The initial thread is the one which executes the `main()` subroutine of a multithreaded process.

2.2.4 Process and Thread Properties

In traditional single-threaded process systems, a process has a set of properties. In systems that support multithreaded processes, these properties are divided between processes and threads.

2.2.4.1 Process Properties

A process in a multithreaded system must be considered as an execution frame. It is the `swappable` entity. It has all traditional process attributes, such as:

- The process ID, the process group ID, the user ID
- The environment
- The working directory

A process also provides a common address space and common system resources, such as:

- File descriptors
- Signal actions
- Shared libraries
- Inter-process communication tools (such as message queues, pipes, semaphores, or shared memory)

2.2.4.2 Thread Properties

A thread is the `schedulable` entity. It has only those properties that are required to ensure its independent flow of control. These include the following properties:

- The stack
- Scheduling properties (such as policy or priority)
- Set of pending and blocked signals
- Some thread-specific data

An example of thread-specific data is the error indicator, *errno.* In multithreaded systems, *errno* is no longer a global variable but usually a subroutine returning a thread-specific *errno* value. Some other systems may provide other implementations of *errno.*

Threads within a process must not be considered as a group of processes. All threads share the same address space. This means that two pointers having the same value in two threads refer to the same data. Also, if any thread changes one of the shared system resources, all threads within the process are affected. For example, if a thread closes a file, the file is closed for all threads.

2.2.4.3 What is Shared between Threads

All the threads within the same process share the following characteristics:

- The address space
- The session membership
- The current working directory
- Files descriptors
- Filemode creation masks
- The process ID, parent process ID, process group ID
- The real and effective user ID
- The nice value
- Signal handlers
- Per-process timers

2.2.4.4 What is Not Shared between Threads

Here are some characteristics that are not shared between threads:

- The thread identifier
- The stack
- The signal mask
- The priority
- The scheduling policy
- The errno variable

Figure 23 shows the structure of threads and processes

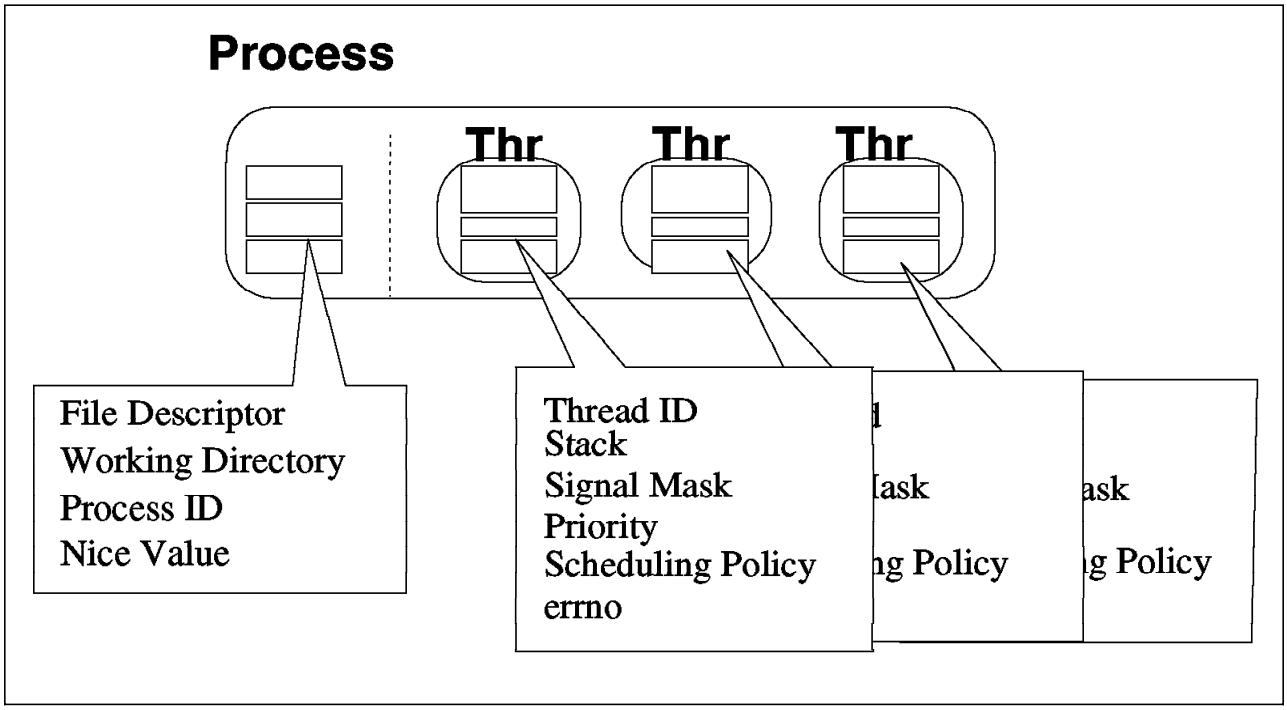


Figure 23. Threads and Processes Properties

2.2.5 Main Benefits of Threads over Processes

We have seen in Chapter 1, “Multiprocessing Concepts” on page 1, that there are two main ways of writing a parallel application: by using multiple processes or by using multiple threads within the same process.

For the following reasons, multithreaded programs can improve performance in many ways compared to traditional parallel programs that use multiple processes.

Managing threads (creating and controlling their execution) requires fewer system resources than managing processes. Creating a thread only requires a single call: `pthread_create()`. Creating a process is far more expensive since the entire parent process addressing space is duplicated. The threads Application Programming Interface (API) is also easier to use than the one for managing processes.

Inter-thread communication is also far more efficient and easier to use than inter-process communication (IPC). Since all threads within a process share the same address space, they can easily share data. Shared data should be protected from concurrent access by using synchronization tools that are usually provided by the threads library. These tools can easily replace traditional inter-process communication facilities. Note that pipes can still be used as an inter-thread communication path.

In summary, a multithreaded application will run faster than an application made with several processes.

2.3 Threads Types

In AIX V4, you will find three types of threads: user threads, kernel threads and kernel-only threads.

2.3.1 User Threads

A user thread is an entity used by application programmers to handle multiple flows of control within a process. The Application Programming Interface for handling user threads is provided by a library, the threads library. A user thread only exists within a process; a user thread in process A cannot reference a user thread in process B.

Note: Even though the library uses a proprietary interface to handle kernel threads for executing user threads, the user threads API is part of a portable programming model. As a result, a multithreaded program developed on an AIX V4 system can easily be ported to another system.

2.3.2 Kernel Threads

A kernel thread is a kernel entity handled by the system scheduler. A kernel thread runs within a process but can be referenced by any other thread in the system. The application programmer has no direct control over these threads, unless writing kernel extensions or device drivers.

2.3.3 Kernel-only Threads

A kernel-only thread is a kernel thread that executes only in the kernel mode environment. Kernel-only threads are controlled by the kernel programmer through kernel services.

2.4 Threads Implementation Models

A multithreaded system (that is a system that supports threads) can have different threads implementations. All implementations have a three-layer architecture with user threads on top of virtual processors (VP) which are themselves on top of kernel threads.

2.4.1 Model Descriptions

User threads are mapped to kernel threads by the threads library. The mapping depends on the model used for implementing threads. There are three possible threads implementation models, corresponding to three different ways to map user threads to kernel threads.

- M:1 model
- 1:1 model
- M:N model

The mapping of user threads to kernel threads is done by using Virtual Processors (VP). A Virtual Processor is a library entity. For a user thread, the Virtual Processor behaves like a CPU for a kernel thread. In the library, the Virtual Processor is a kernel thread or a structure bound to a kernel thread.

2.4.1.1 M:1 Model

In the *M:1* model, all user threads are mapped to one kernel thread; all user threads run on one VP. The scheduling of user threads on this unique VP is done by the library itself. All user threads programming facilities are completely handled by the library. This model can be used on any system, especially on traditional systems which do not support threads. For example, the DCE (Distributed Computing Environment) threads used to have an *M:1* implementation on AIX V3.2. Figure 24 on page 37 is an illustration of this model.

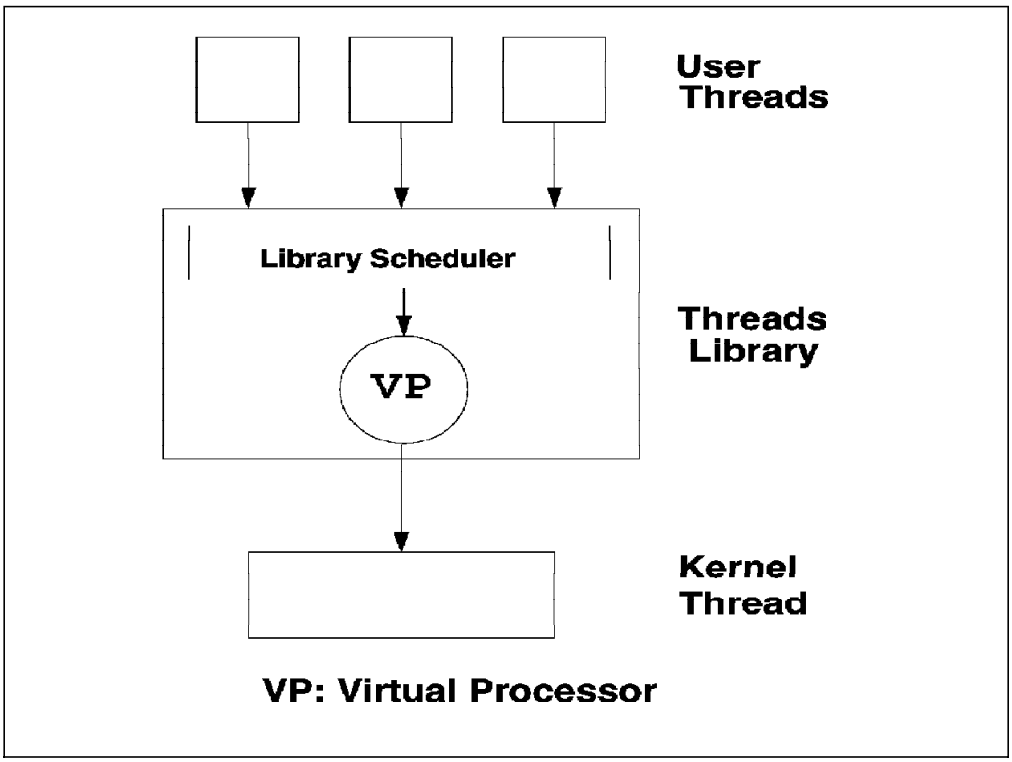


Figure 24. M:1 Model

2.4.1.2 1:1 Model

The *1:1* model is the one which is currently implemented by the AIX V4 threads library. In this case, each user thread is mapped to one kernel thread through a VP.

In fact, the library provides user threads, which simply are a structure describing the thread, a user stack and a link to the VP. When a user thread is created by the user, a Virtual Processor is created, and from that time on, the user thread and the Virtual Processor remain linked until the user thread is deleted. So, a user thread is nothing more than a user abstraction of a Virtual Processor.

When a Virtual Processor is created by the library, a kernel thread is created, and from that time on, the Virtual Processor and the kernel thread remain linked until the Virtual Processor is deleted, which implies kernel thread deletion as well. So, a Virtual Processor is nothing more than a library abstraction of a kernel thread.

However, unlike the binding between Virtual Processors and the kernel threads, which implies simultaneous death, the binding between user threads and Virtual Processors does not imply simultaneous death. In order to be more efficient, when a user thread is deleted, the virtual processor is not destroyed; it is just suspended. It will be reused if a new user thread is created in lieu of a fresh Virtual Processor.

At the user level, each thread has its own stack and a pthread structure with a saving area for its registers; *errno* is thread dependent and is stored at the top of the user stack. Also, each thread has its own signal mask, but all threads within a process will share the same signal handlers. Figure 25 on page 38 illustrates the *1:1* model.

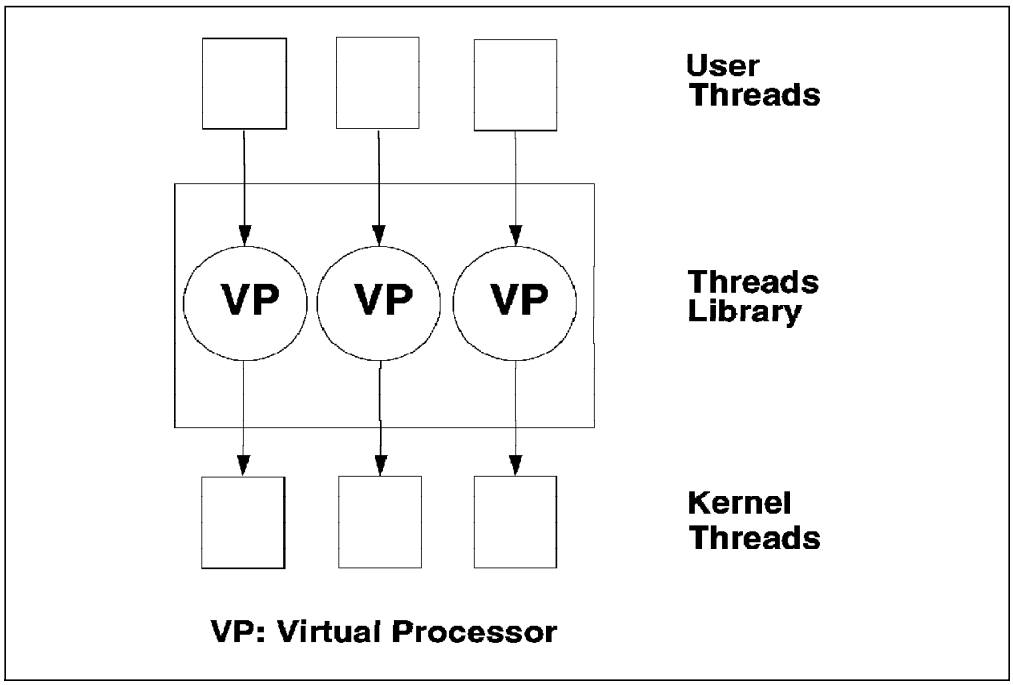


Figure 25. 1:1 Model

2.4.1.3 M:N Model

In the *M:N* model, a user thread is not necessarily bound to a VP. Several threads can share the same VP or the same pool of VPs. Each VP can be thought of as a virtual CPU available for executing user code and system calls. A set of VPs can be thought of as a virtual multiprocessor.

A thread which is not bound to a VP is said to be a *local scope* because it is not directly scheduled with all the other threads by the kernel scheduler.

A scheduler inside the library is responsible for dispatching the local scope threads to the pool of VPs. Whenever it is possible, this scheduler does not enter the kernel. In particular, the user thread context switch should be as fast as possible and should not use any system call.

Threads in the kernel fall in two categories, those used only inside the kernel and those representing a VP. All the kernel threads are scheduled by the kernel onto the available CPU resources according to their class priority.

Figure 26 on page 39 shows the *M:N* model implementation.

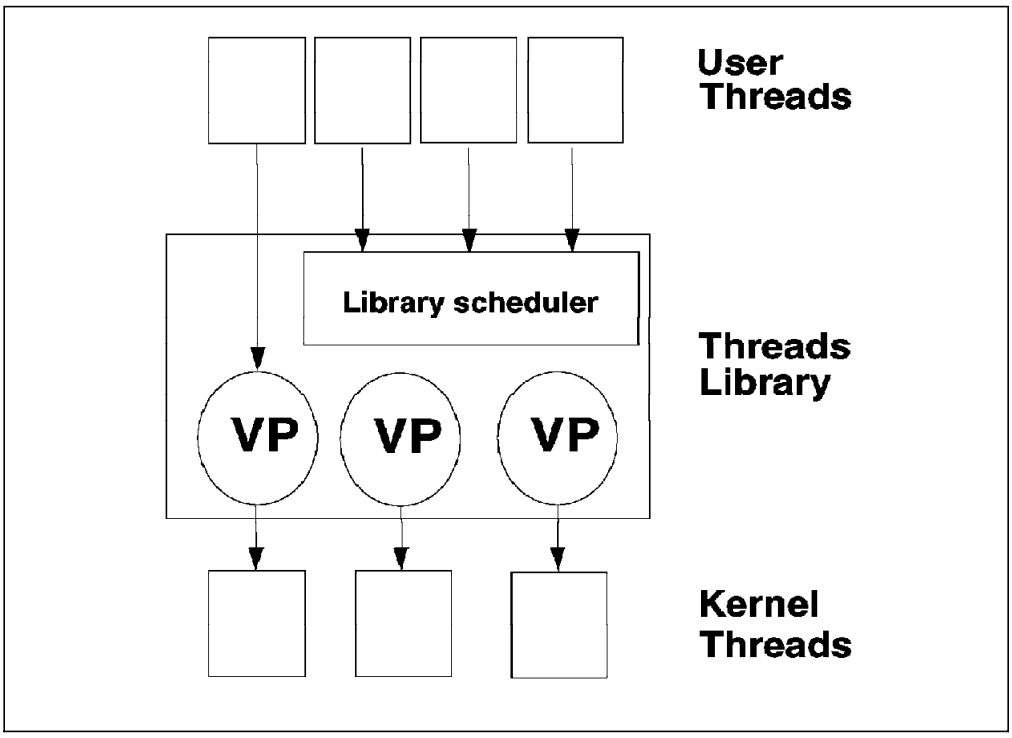


Figure 26. M:N Model

2.5 Contention Scope

The contention scope of a user thread defines how it is mapped to a kernel thread. There are two possible contention scopes: the *system contention scope* and the *process contention scope*.

A *system contention scope* user thread is a user thread that is directly mapped to one kernel thread. All user threads in a *1:1* thread model have system contention scope.

A *process contention scope* user thread is a user thread that shares a kernel thread with other (process contention scope) user threads in a process. All user threads in an *M:1* model have process contention scope, unless there is only one user thread in the process.

In an *M:N* thread model, user threads can have either system or process contention scope. In the previous figure, for example, the user thread on the left side has system contention scope; the other ones all have process contention scope. Therefore, an *M:N* model is often referred as a *mixed-scope* model.

2.6 AIX V4 Kernel Support of Threads

AIX V4 has a multithreaded kernel. It supports a large number of threads throughout the system (up to 256K), and the maximum number of threads per process is 1024.

In order to support the porting of code from OSF/1 1.1 and user level threads, new functions have been added to AIX. The functions, which are said to be called from the user environment, are system calls. The functions that are said

to be called from the kernel environment are kernel services. This set of kernel services allows developers of device drivers or kernel extensions to create and manage kernel threads.

Locking services are also provided to assist the kernel developer working in the multiprocessor environment.

2.7 AIX V4 Threads Library Implementation

The AIX V4 threads architecture was jointly developed by IBM and Bull. AIX provides a threads library, called libpthreads.a. It implements a *1:1* model and follows POSIX 1003.4a Draft 7 specifications. It is IBM′s intention to implement an *M:N* model later and also to move to the official version of the POSIX standard when it is approved.

Any program written for use with a POSIX thread library can easily be ported for use with another POSIX threads library; only the performance and very few subroutines of the threads library are implementation dependent. Figure 27 shows the AIX V4 threads implementation.

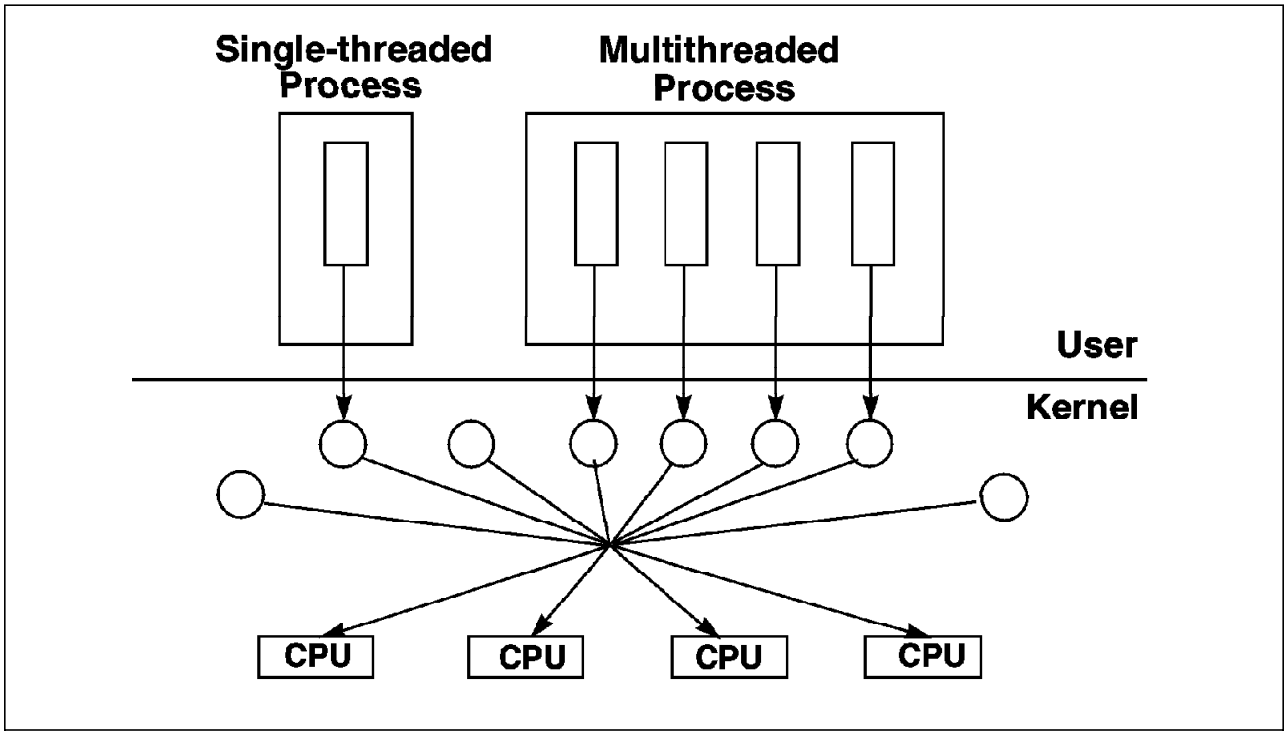


Figure 27. AIX V4 Thread Architecture

2.8 Threads Scheduling

In AIX Version 3.2, the scheduler dispatches processes. In AIX Version 4, the scheduler dispatches threads (a thread is the dispatchable unit for the scheduler).

Each thread created has its own priority, just like a process in AIX Version 3.2. The priority is an integer value in the range from 0 to 127, where 0 is the most favored priority and 127 the least favored. The priority of a thread can be

reported by the ps command. Priority level 0 cannot be used at the user level; it is reserved for the system.

There are three scheduling algorithms that can be selected when creating a thread:

1. **SCHED_RR:**

   This is a round-robin scheduling mechanism. The thread is scheduled for a slice of time (10ms by default) with a fixed priority. It is put back in the run queue of its priority after the end of its time slice. SCHED_RR policy is similar to creating a fixed-priority, real-time process. The thread must have root authority to be able to use this scheduling mechanism. SCHED_RR is a preemptive mechanism.

2. **SCHED_FIFO:**

   This is a non-preemptive scheduling mechanism. A thread created with SCHED_FIFO will run at a fixed priority and will not be timesliced. It will run to completion unless it is blocked or unless it voluntarily yields control of the CPU. If several threads have the same priority, they will be dispatched in a FIFO (First-In First-Out) order. A thread must also have root authority to use a SCHED_FIFO scheduling policy.

   Note: It is possible to create a thread with a SCHED_FIFO policy which has a priority high enough that it could monopolize the processor.

3. **SCHED_OTHER:**

   This is the normal and default AIX scheduling policy. Thread execution is timesliced, and the priority is dynamically modified by the scheduler according to the time already spent on a CPU. The more CPU time consumed, the more the priority is decreased.

In AIX, the time slice value (10ms by default) can be changed with the schedtune command.

2.8.1 Threads Programming Considerations

Facilities are provided for the programmer to do many of the same things that can be done with processes. These facilities include calls for thread creation, termination, synchronization, communication, error recovery and management. When a thread is running in user space, it can make system calls just like in a simple process. Thus, the programmer has the same level of control over how the threads behave and what services they request of the system within the process that they have always had for individual processes.

2.8.2 Thread-Safe Libraries

In single-threaded processes there is only one flow of control. Therefore, the code executed by these processes does not need to be reentrant or thread-safe. Reentrance and thread safety are two different concepts.

In a multithreaded process, two threads can call the same function at the same time, or two threads can access the same resource at the same time.

To avoid data corruption when two threads call the same function at the same time, functions must be reentrant. To avoid data corruption when two functions access the same resource at the same time, functions must be thread-safe; that is, shared resources must be protected by locks.

Therefore, a multithreaded program must use both reentrant and thread-safe functions. Usually, a reentrant function is also thread-safe, and a non-reentrant function is usually thread-unsafe.

We say that a library is thread-safe when multiple threads can be running a function in that library without data corruption. All the functions within that library must be both reentrant and thread-safe.

In the existing C library, most standard functions are reentrant, but some of them are not. Equivalent reentrant functions are provided in a specific library. These functions are characterized by the suffix _r and stored in the libc_r.a library.

These libraries are thread-safe in AIX V4.1:

- libc_r.a
- libC_r.a
- libnetsvc_r.a
- libtli_r.a
- libxti_r.a
- libbsd.a
- libpthreads.a

The library libc_r.a is needed by all multithreaded applications. This library, besides being thread-safe, contains modifications to *crt0* and *fork* to handle threads creation. A variety of other calls were changed internally to handle threads cancellation conditions.

2.8.3 Threads Creation

Creating a thread is accomplished by calling the pthread_create() subroutine. This subroutine creates a new thread and makes it runnable.

When calling the pthread_create() subroutine, you must specify an entry point subroutine. This subroutine, provided by your program, is like the main subroutine for the process. It is the first user subroutine executed by the new thread.

The pthread_create() subroutine returns the thread ID of the new thread. The caller can use this thread ID to perform various operations on the thread.

2.8.4 Thread Attributes

When creating a thread, you can pass some attributes to it. These attributes specify the characteristics of the thread. The attributes' default values fit for most common cases.

The thread attributes are stored in an attribute object at the thread creation. This object must be defined before creating the thread. An example of a thread attribute is the scheduling policy of the thread (SHED_RR, SCHED_FIFO, SCHED_OTHER).

2.8.5 Threads Synchronization

One main benefit of using threads is the ease for using synchronization facilities. Three basic synchronization techniques are implemented in the threads library, mutexes, condition variables and joining.

Mutexes:

A mutex is a mutual exclusion lock. When a thread needs to access a shared resource (a global variable for example), the thread must lock the mutex using the `pthread_mutex_lock()` subroutine. Since only one thread at a time can hold the lock, if the lock is busy (held by another thread), the thread will be blocked. To avoid being blocked if the lock is busy and to continue running, the thread can issue a `pthread_mutex_trylock()`. If the lock is free, the lock is granted to the thread. Unlocking the mutex is done through the `pthread_mutex_unlock()` subroutine.

Condition Variables:

Condition variables allow threads to wait until some event or condition has occurred.

Condition variables use three objects: a Boolean variable (called a predicate) indicating whether the condition is met, a mutex to serialize access to the Boolean variable and a condition to wait for the condition.

Using condition variables requires some effort from the programmer. However, condition variables allow the implementation of powerful and efficient synchronization mechanisms.

The `pthread_cond_wait()` subroutine causes a thread to wait until the condition variable is signaled or broadcast.

Joining:

Joining a thread means waiting for it to terminate. It can be seen as a specific usage of condition variables.

The *pthread_join()* subroutine provides a simple mechanism that allows a thread to wait for another thread to terminate. In fact, the subroutine blocks the calling thread until the specified thread terminates.

A thread cannot join itself. If a thread tried to join itself, a deadlock would occur and would be detected by the library. However, two threads may try to join each other. If this happens, the two threads will deadlock. This situation is not detected by the library.

2.8.6 Threads Termination

A thread automatically terminates when it returns from its entry point subroutine. A thread can also explicitly terminate itself, or it can terminate any other thread in the process.

A thread can exit by calling the `pthread_exit()` subroutine. If a thread within the process calls the `exit` subroutine, this will terminate the entire process, including all its threads.

Therefore, in a multithreaded program, the exit() subroutine should only be used when the entire process needs to be terminated, as in the case of an unrecoverable error, for example. The pthread_exit() subroutine should be preferred, even for exiting the initial thread.

Note that returning from the initial thread using the pthread_exit() subroutine does not terminate the process, only the initial thread. The process will be terminated when all threads in the process terminate.

Also, a thread can terminate the execution of any other thread in the process in a controlled manner by using the pthread_cancel() subroutine. The target thread (that is, the one that's being canceled) can hold cancellation requests pending in a number of ways and perform application-specific clean-up processing when the notice of cancellation is acted upon.

2.8.7 Forking Considerations

There are two reasons why traditional UNIX applications use the fork() system call.

- One is to create a new thread of control within the same program. In a multithreaded environment, this use can be replaced by the creation of a new thread within the multithreaded process by using the pthread_create() subroutine.
- The other reason is to create a new process running a different program. In this case, the fork() system call is followed by an exec() system call.

In a multithreaded system, the semantics of fork() could be either to copy all of the threads into the new process or to copy only the thread that calls the fork system call. The POSIX 1003.4a Draft 7 specification, written as it was by the committee, keeps both alternatives, making the first one optional.

In AIX Version 4, the child process is created with a single thread (the calling thread). This new process contains a replica of the calling thread and its entire address space at the time of the fork().

This means that the address space could contain mutexes held by threads that do not exist in the child process. Likewise, the data protected by these locks might not be in a consistent state. If the child process attempts to acquire one of these mutexes, it could hang. If it attempts to use the data protected by such a mutex, it could malfunction.

Therefore, any application using fork() in a multithreaded application should only execute safe operations between the call to fork() and the call to exec() to avoid errors. Safe operations are those that either do not take locks or are known to be safe by the application.

Simultaneous fork() calls by different threads of the same process are serialized at the kernel level.

2.8.8 Threads Scheduling

The scheduling policy of a thread is a thread's attribute which must be stored in the attribute object before the creation of the thread. This can be done by using the `pthread_attr_setschedpolicy()` subroutine.

As in AIX V3.2, the priority is process-based. The `nice()`, `setpriority()` and `getpriority()` subroutines work at the process level. If a multithreaded application calls one of these subroutines which modifies the nice value for the process, this change will affect all the threads in the process.

2.8.9 Signal Management

Signals in a multithreaded environment are an extension of signals in a traditional single-threaded environment. This allows compatibility with previous single-threaded process. Programs handling signals and written for single-threaded systems will behave as expected in AIX V4.

POSIX.4a Draft 7 defines the following model for signal handling in a multithreaded program:

- Signal handlers are per process: This means that when a thread installs a signal handler, this handler is invoked when the signal occurs in any thread.
- Signal masks are per thread: This means that a thread can block a signal from delivery, but this will not prevent other threads from receiving the signal.
- Single delivery of each signal: This means that a signal is delivered to one thread. If more than one thread is interested in the same signal and this signal occurs, then one thread will receive the signal.

There are two types of signals:

- Synchronous signals:

  In this case, the signal is the result of an event that occurs in the running thread and is delivered synchronously with respect to that event.

- Asynchronous signals:

  The signal is the result of an event that may be external to the current thread or process and is delivered at any point in the thread execution when such an event occurs.

Signal handlers are installed using the `sigaction()` function. This function is used to establish actions to be taken upon receipt of a signal. The process maintains a list of actions associated with each signal number. This list is shared by all the threads in the process. If the action specifies termination, stop or continue, the entire process is affected. The signals *SIGSTOP* and *SIGKILL* are never caught, ignored or masked.

The `pthread_kill()` function is used to send signals to a particular thread within the process. If the receiving thread has blocked the signal, it remains pending on the thread. Once a signal becomes pending for a thread, it will not become pending for, or delivered to, another thread.

The `sigwait()` function is used by the thread to wait synchronously for a set of asynchronous signals. If more than one thread is using `sigwait()` to wait for the

same signal, only one of these threads will return from sigwait() with the signal number.

2.8.10 Compiling Multithreaded Programs

In order to compile a multithreaded program, you must use the cc_r command instead of the cc command, or the xlc_r instead of the xlc command.

When compiling a multithreaded program, you must link at least the libc_r.a and the libpthread.a libraries. You can look at the sample Makefile file which is provided in Appendix B, “Sample Programs” on page 333.

2.8.11 Debugger Threads Support

The AIX V4.1 dbx debugger allows the developer to debug multithreaded applications. The debugger has some new commands that display information on threads, condition variables, attributes and mutexes. These commands are thread, mutex, condition and attribute.

2.8.12 A Multithreaded Program Sample

This multithreaded program is very short. It displays ″Hello!″ in both English and French for five seconds. The initial thread (executing the main subroutine) creates two threads. Both threads have the same entry point subroutine (the Thread subroutine) but a different parameter. The parameter is a pointer to the string that will be displayed. Some other multithreaded program samples are provided in Appendix B, “Sample Programs” on page 333.

```
# include <pthread.h>
# include <stdio.h>
# include <unistd.h>

void *Thread(void *string)
{ while (1)
        printf("%s\n", (char*) string);
  pthread_exit(NULL);
}
int main()
{ char e_str[] = "Hello !";
  char f_str[] = "Bonjour !";

  pthread_t e_th;
  pthread_t f_th;
  int rc;

  rc = pthread_create(&e_th, NULL, Thread, (void *)e_str);
  if (rc) exit(-1);
  rc = pthread_create(&f_th, NULL, Thread, (void *)f_str);
  if (rc) exit(-1);
  sleep(5);
  exit(0);
}
```

Figure 28. Multithreaded Program Sample

2.8.13 AIX V4 Threads Programming Interface

Following are examples of subroutines provided by the threads library:

- read_atfork
- pthread_attr_destroy
- pthread_attr_getdetachstate
- pthread_attr_getinheritsched
- pthread_attr_getschedparam
- pthread_attr_getschedpolicy
- pthread_attr_getscope
- pthread_attr_getstackaddr
- pthread_attr_getstacksize
- pthread_attr_init
- pthread_attr_setdetachstate
- pthread_attr_setinheritsched
- pthread_attr_setschedparam
- pthread_attr_setschedpolicy
- pthread_attr_setscope
- pthread_attr_setstackaddr
- pthread_attr_setstacksize
- pthread_cancel
- pthread_cleanup_pop
- pthread_cleanup_push
- pthread_condattr_destroy
- pthread_condattr_getpshared
- pthread_condattr_init
- pthread_condattr_setpshared
- pthread_cond_broadcast
- pthread_cond_destroy
- pthread_cond_init
- pthread_cond_signal
- pthread_cond_timedwait
- pthread_cond_wait
- pthread_create
- pthread_equal
- pthread_exit
- pthread_getschedparam
- pthread_getspecific
- pthread_join
- pthread_key_create
- pthread_key_delete
- pthread_kill
- pthread_mutexattr_destroy
- pthread_mutexattr_getprioceiling
- pthread_mutexattr_getprotocol
- pthread_mutexattr_getpshared
- pthread_mutexattr_init
- pthread_mutexattr_setprioceiling
- pthread_mutexattr_setprotocol
- pthread_mutexattr_setpshared
- pthread_mutex_destroy
- pthread_mutex_getprioceiling
- pthread_mutex_init
- pthread_mutex_lock
- pthread_mutex_setprioceiling

- pthread_mutex_trylock
- pthread_mutex_unlock
- pthread_once
- pthread_self
- pthread_setcancelstate
- pthread_setcanceltype
- pthread_setschedparam
- pthread_setspecific
- pthread_testcancel
- pthread_yield
- sigthreadmask

Chapter 3. SMP Servers Architecture

This chapter discusses the way the IBM SMP servers are designed for use in a commercial environment. It first highlights the technical issues in designing an SMP system and then describes the actual IBM SMP hardware architecture.

3.1 SMP Design Issues in a Commercial Environment

There are many technical aspects you must look at when designing an SMP system. You must have adequate input/output facilities, adequate memory size, reliability, availability, serviceability, manufacturability and so on. A key performance issue on which much attention must be lavished is the way that processors perform memory communication, not just directly with memory but among each other.

The scalability of an SMP system (that is its ability to get much more performance when adding a new processor) depends a lot on the way processors communicate with each other and with the memory. Inter-processor communication is one of the main performance and scalability factors of commercial symmetric multiprocessors.

3.1.1 Memory Hierarchy

In Chapter 1, “Multiprocessing Concepts” on page 1, we introduced the memory hierarchy. Since having caches improves data locality, most systems implement a multilevel cache structure.

The memory hierarchy is composed of:

- Processor’s registers
- A first level of cache (L1), which is a very fast memory with a small capacity
- A second level of cache (L2), which is bigger than L1 but a bit slower
- The main memory, this is slower than L2 but has a much greater capacity
- Disks

Let us compare the different latencies of the different memory components in a typical system equipped with a processor running at a clock rate in the range of 75 to 150 MHz. The latency is the time it takes to access data from the memory component.

On a typical implementation, it will take one cycle to access data from L1 if there is a cache hit in L1. It will take between 7 to 10 cycles to access data from L2 in case of a cache miss in L1 and a cache hit in L2. It will take between 20 to 50 cycles to get data from memory in case of a cache miss in L2. And finally, if you need to access data from disk, it will take between 750000 to 1.5 million cycles to access data. To highlight this point, if we assume that one cycle is one second on a processor running at 150 MHz, it will take 17 days, 8 hours and 40 minutes to access data from disk whereas accessing data in L1 cache would take 1 second!

Clearly, in terms of performance, accessing data from disk must be avoided at all costs. Thus, a commercial system will have to be designed in such a way that misses to disks are avoided as much as possible. Since the memory

latency is much better than the disk's latency, the memory itself should be used as a huge cache. Therefore, there is a need for high memory capacity.

3.1.2 Scientific vs. Commercial Environment

When designing an SMP system, it is important to understand the differences between a commercial environment and a technical environment.

In an engineering/scientific environment, programs are often made of short loops working with data organized by the compiler so they fit into the first level of cache (L1). In such an environment, the hit ratio is very good and usually around 97%. This means that 97 percent of the time, data will be found in L1 and 3 percent of the time, the processor will have to fetch data out of L2 (if there is an L2 cache) or in the main memory. We will see later that the use of an L2 cache does not increase the program speed by much.

In a transactional database environment, the hit ratio is not as good. Large programs, frequent branches, widely dispersed data references, large numbers of users, large number of processes and high process switch rates all combine to produce a high miss-rate for the L1 cache. Typically, in such an environment, the hit ratio for L1 is around 85 percent. This means that 15 percent of the time, the processor will have to fetch data from L2 (if L2 exists) or from the main memory. We will see that in this case, an L2 cache helps to improve the performance of the system because L2 cache latency is lower than the memory latency.

Figure 29 illustrates the memory hierarchy and the differences between a commercial and a scientific environment in terms of hit ratios.

.

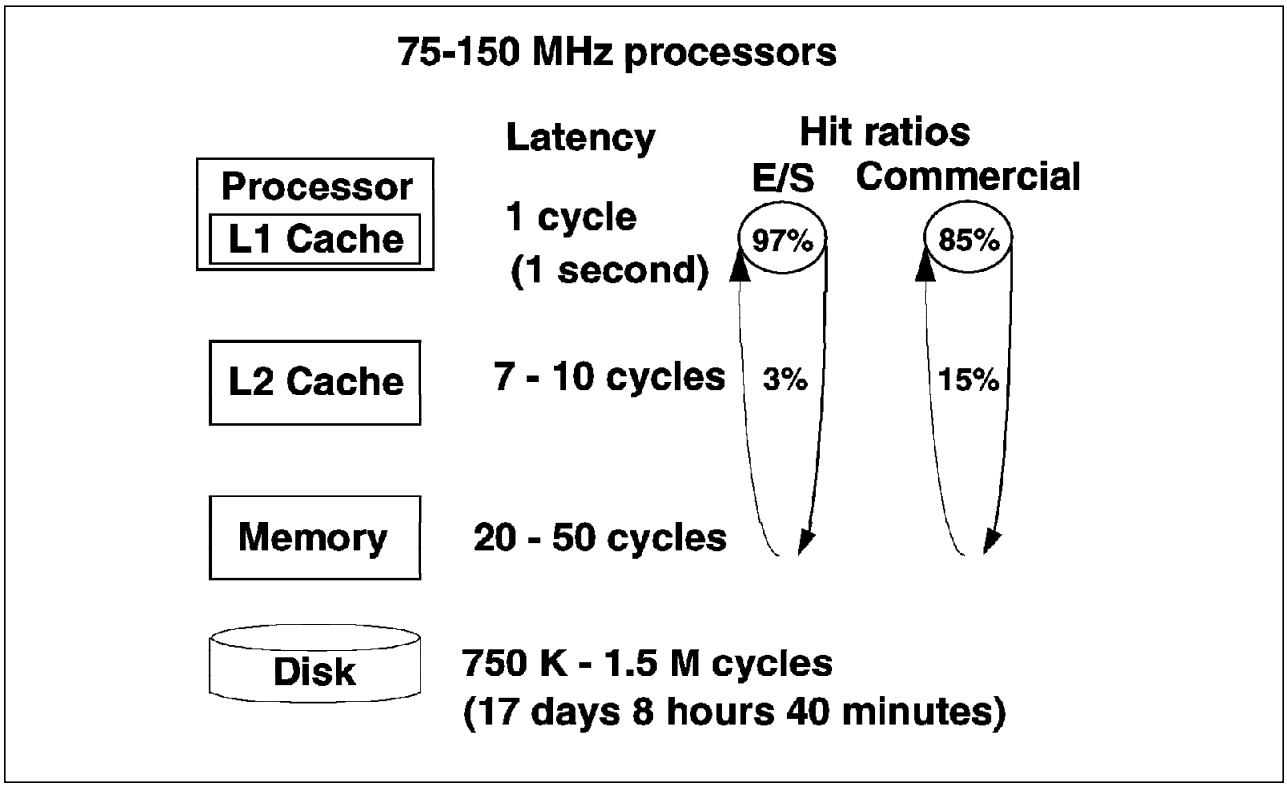


Figure 29. Scientific vs. Commercial Environment

3.1.3 Typical Memory Cycles

In order to better understand the importance of the memory hierarchy's performance, let us look in detail at the number of memory cycles required for a typical processor to access data from the different memory components.

Note that in the rest of the document, data can be instructions (the code of the program itself) or real data. In a commercial environment, you will mostly find instructions in the cache because commercial applications' codes are big. In a technical environment, you will find both instructions and data.

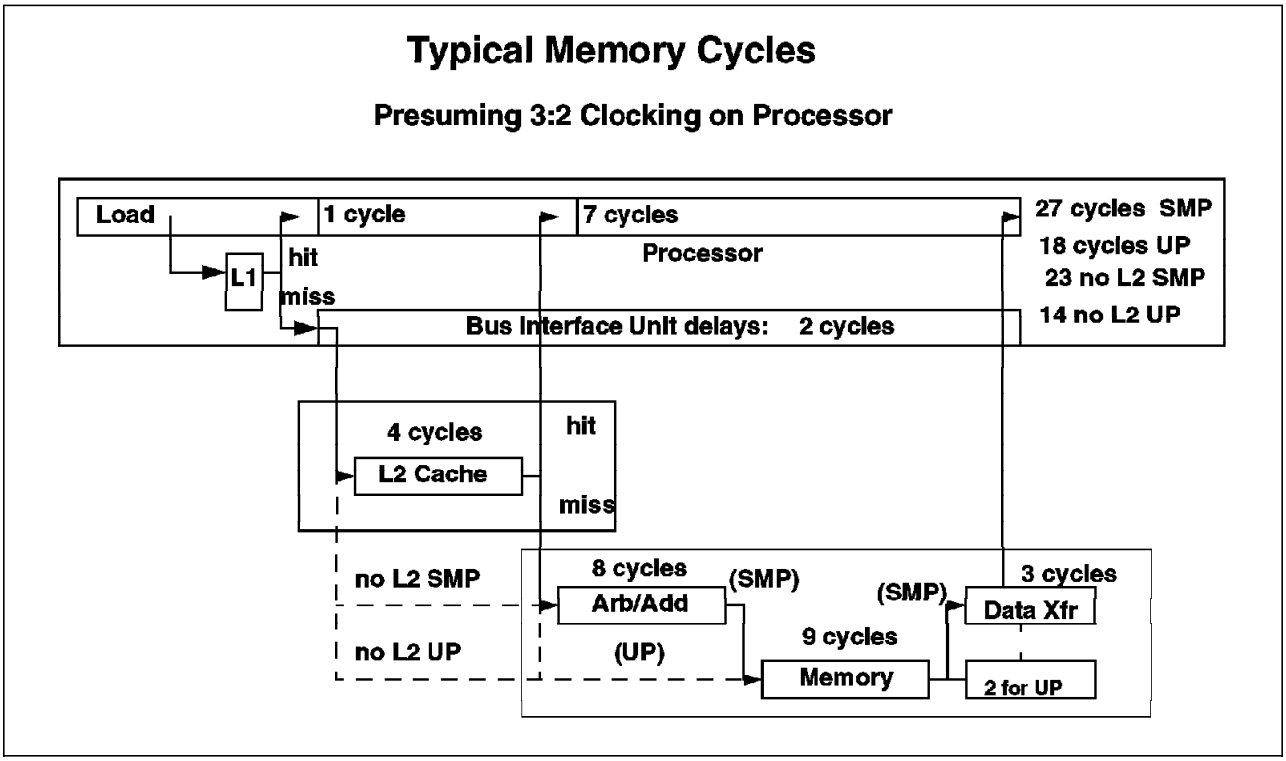


Figure 30. Typical Memory Cycles

In Figure 30, you can see that when there is a hit in L1, it takes only one cycle to access the data.

- If the system does not have any L2 cache, it takes 14 cycles on a uniprocessor to load data from the memory to the processor and 23 cycles for an SMP.
- If the system has an L2 cache, it takes 7 cycles to access data if it is already in the L2 cache (cache hit in L2), but it takes 18 cycles for a UP and 27 cycles for an SMP to access data if there is also a cache miss in L2. Note that there is a two-cycle delay between the processor and L2 or the memory.

Figure 30 shows the memory cycles required for getting data from the memory subsystem on a typical system. A 3:2 clocking rate on the processor means that when the processor runs at 100 MHz, the system bus runs at 66 MHz. The 3:2 is the frequency ratio between the processor frequency and the system bus frequency. A Phase Lock Loop (PLL) technology is used to match the bus and processor operating frequencies.

3.1.4 Miss-Rate Penalty

Since one of the main differences between a commercial environment and a scientific environment is the higher miss rate, it is important to understand the effect of a high miss rate on the performance of a system.

Let us take a 100 MHz processor without an L2 cache. Let us also assume that the measured infinite cache CPI (Cycles Per Instruction) is 1.3 on that processor. The infinite cache CPI is a gauge that gives the relative efficiency of the processor on a specific workload. Its value depends on the workload as well as on the processor itself. The main advantage of using Cycles Per Instruction (CPI) is that it is additive with other CPI components. Depending on the miss rate, each component (L1, L2 and memory) will add its number of CPI. This helps in estimating the overall number of CPI for a given workload and thus in estimating the power of the system on that workload.

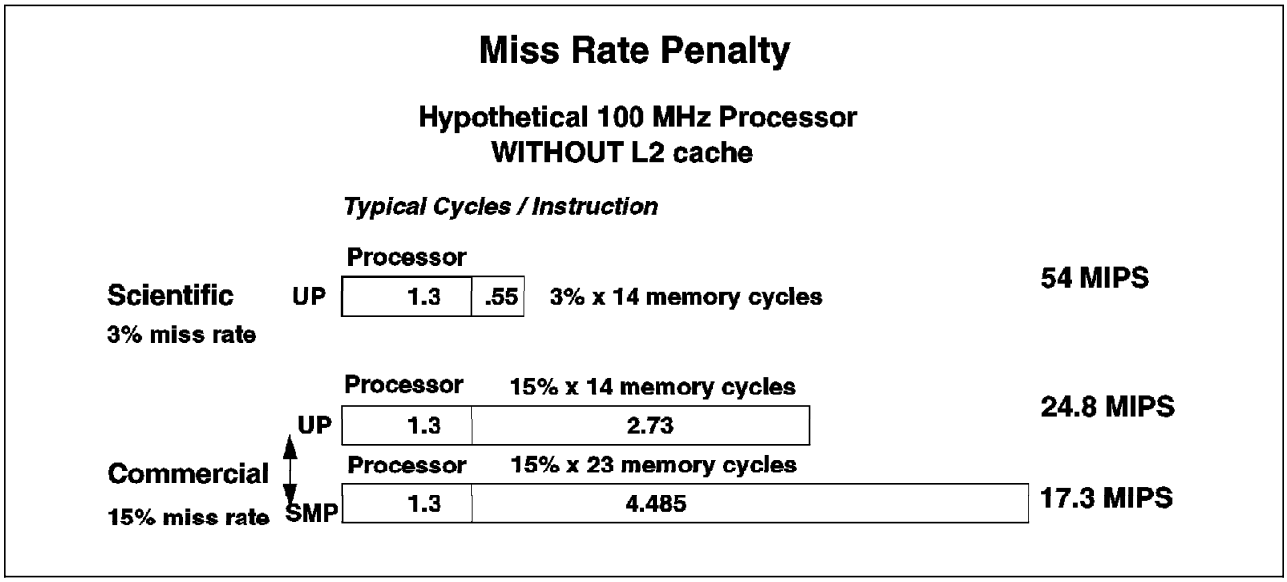


Figure 31. Miss-Rate Penalty

In an engineering and scientific environment, L1 miss rate is around three percent. This means that three percent of the time, you will need 14 cycles on a UP to access your data from the memory. In this case, the total number of CPI will be:

```
1.3 + 0.03x14x1.3 = 1.3 + 0.55 = 1.85 CPI
```

Note: The first 1.3 value is the infinite cache CPI, which can be measured, while the second 1.3 value is the average number of memory requests per instruction. This second value comes from typical instruction mixes where about 30 percent of instructions are either LOADs or STOREs. Each instruction fetch consumes one memory reference; adding in 0.3 memory references due to LOADs and STOREs results in an average value of 1.3 for the average number of memory references per instruction.

At 100 MHz, the machine will deliver 54 MIPS (millions of instructions per second).

In a commercial environment where the miss rate is usually around 15 percent, it will take 14 cycles for a UP to access data from the memory. For an SMP, you will need 23 cycles to access data from the memory.

Therefore, for a UP, the number of CPIs will be:

1.3 + 0.15x14x1.3 = 1.3 + 2.73 = 4.03 CPI

At 100 MHz, the UP system will deliver only 24.8 MIPS. The higher miss rate lowers the performance of the system by 54 percent.

For an SMP, the number of CPIs will be:

1.3 + 0.15x23x1.3 = 1.3 + 4.485 = 5.785 CPI

At 100 MHz, the SMP system will deliver only 17.3 MIPS per processor. Therefore, a high miss rate lowers the performance on both UP and SMP systems. SMPs have a disadvantage due the higher number of cycles required to access data from memory. Figure 31 on page 52 shows the miss-rate penalty for UP and SMP systems which do not have an L2 cache.

3.1.5 Effect of L2 Cache

What happens if you add an L2 cache to a UP or to an SMP system. Adding an L2 cache has a different effect according to the environment. Let us assume that in case of an L1 cache miss, the probability of finding the data in L2 is 80 percent. This means the miss rate is 20 percent in L2.

3.1.5.1 Scientific Environment

On a UP system equipped with an L2 cache, it takes seven cycles to get data from L2 and 18 cycles to get data from the memory. We can calculate the average number of additional cycles per instruction needed in case of an L1 cache miss.

0.03x0.8x7x1.3 + 0.03x0.2x18x1.3 = 0.22 + 0.14 = 0.36 CPI versus 0.55

Adding an L2 cache to a UP system only saves 0.19 CPI in a scientific environment. An L2 cache does not significantly improve the performance of the system in a scientific environment (60.2 MIPS instead of 54.0 MIPS).

3.1.5.2 Commercial Environment

On an SMP equipped with an L2 cache, it takes seven cycles to access data from L2 and 27 cycles from the memory. Thus, with a 15 percent L1 miss rate and a 20 percent L2 miss rate, the number of additional cycles per instruction due to the 15 percent L1 miss rate is:

0.15x0.8x7x1.3 + 0.15x0.2x27x1.3 = 1.09 + 1.05 = 2.14 CPI versus 4.485

In this case, the L2 cache has a great effect and can increase the performance of the system up to 67 percent. Figure 32 on page 54 shows the L2 cache effect for a UP in a scientific environment and the L2 cache effect of an SMP in a commercial environment.

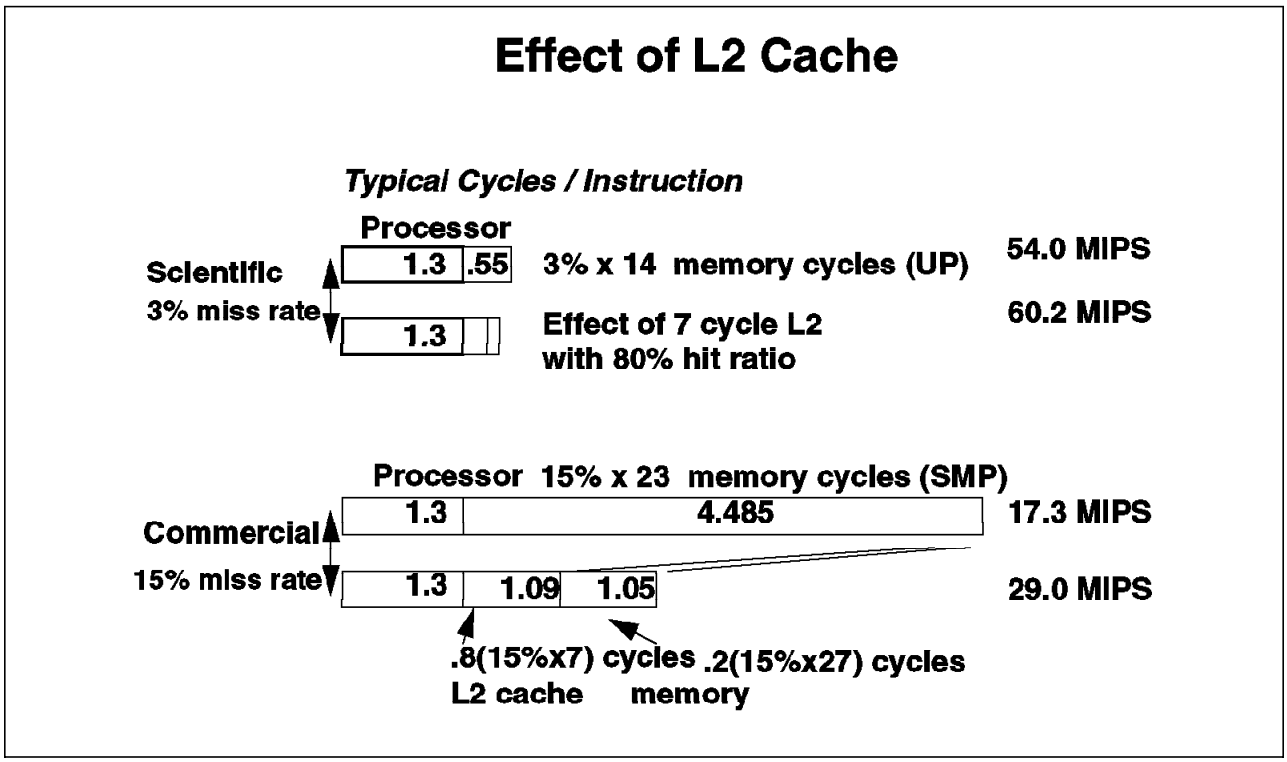


Figure 32. Effect of L2 Cache

3.1.6 Processor Speed Effect

What happens to the system if we double the processor speed? Within the processor itself, with its L1 cache, the number of CPIs will be the same as previously (note that this is true if L1 is embedded in the processor chip itself). Thus, the running time will be divided by two if all data is found in L1. But in case of a cache miss, because the L2 cache and the memory latencies are still the same, L2 and the memory will add more CPI. As a matter of fact, L2 cache latency and memory latency are technology dependent. It is very difficult, and very expensive, to increase the speed of L2 and memory.

Therefore, for a defined L2 and memory technology, the overall time needed to run a specific workload will not be divided by two if you double the processor speed.

Doubling the processor speed will only yield a 23 percent improvement in performance. If the processor speed is multiplied by four, the improvement will be 38 percent. If the speed is multiplied by eight, the improvement will be 48 percent. Figure 33 on page 55 illustrates the effect of doubling the processor speed.

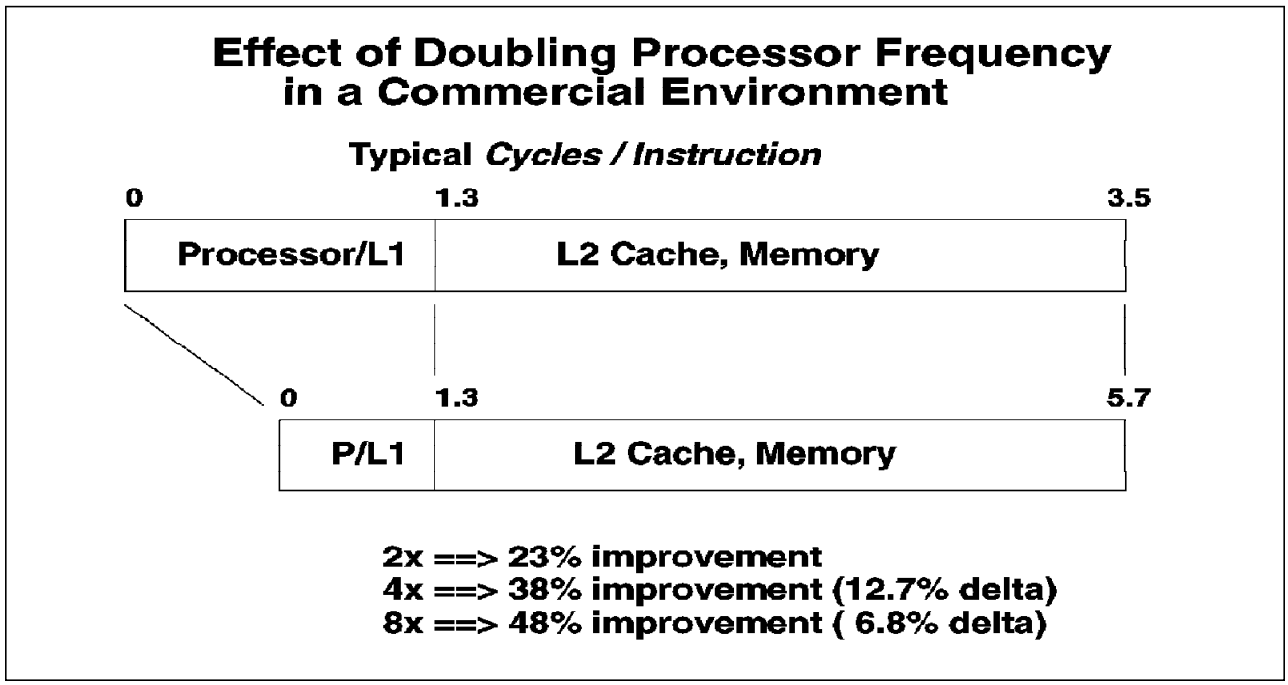


Figure 33. Processor Speed Effect

The conclusion is that, in a commercial environment where cache misses ratios are very high, the **performance of the memory subsystem is key**. Performance and scaling can be achieved by improving the memory subsystem performance.

3.2 SMP Hardware Architecture

This section introduces the hardware design of the SMP servers.

3.2.1 SMP Design Rationale

All the IBM SMP design rationale is based on the fact that the memory subsystem is key to the performance and the scaling of the system.

IBM SMP design is the result of extensive research into the performance characteristics of today′s commercial applications. Typically, this includes manipulating vast amounts of data and sharing data between many users and/or programs. At a programmatic level, this is known as a large data working set with low data locality. If such an application is running on an SMP system, two particular effects will be noticed.

- Due to the low probability of finding the appropriate data already in the cache, there will be a high level of data traffic generated between system memory and CPU caches.
- In an SMP system, the default behavior of the scheduler is to execute the next runnable thread on the first processor to become free. This lateral process migration causes a dynamic increase in the level of data traffic between CPU caches. The physical implementation of cache coherency, therefore, becomes a key to global system performance.

Therefore, the memory subsystem will require high-speed caches, large caches, a high bandwidth between processors and memory, a high bandwidth between

the processors themselves (for cache-to-cache transfers) and a high bandwidth between the processors and the I/O subsystem.

Traditionally, in SMP architecture, the interconnection between the CPU cache and global memory is met by means of a common memory bus shared among various resources. This is typically the most stressed point of the architecture, and it tends to become saturated as the number of processors in the system increases. This situation occurs because the data traffic between caches and memory increases along with cache-to-cache transfers, and they all compete for bandwidth on the memory bus.

3.2.2 Why a Switch?

If all the processors are tied together using a bus, you will find different types of activities on the bus: the snooping activity, the addressing and data transfer between processors or between a processor and the memory, and the data transfer between a processor and the I/O subsystem.

We have seen previously, in Chapter 1, “Multiprocessing Concepts” on page 1, that snooping solves a major problem in SMP design, cache coherency. It keeps caches consistent. Whenever a processor modifies data in its own cache, it broadcasts that information to the other processors so they can invalidate the corresponding memory address in their caches. Also, whenever a processor needs data that is not in its local, fast-cache memory, it broadcasts that fact throughout the system bus. The cache that has the data gives it to the requestor via the bus and, while doing so, notifies the memory - which has not found anything yet because it is slower - to quit trying. It is possible that megabytes of data are moved from one cache to another cache.

In a snoopy-bus scenario, a bus must be used twice for each memory load request: once to make the request and again to return the data. Both operations cannot take place at the same time. When the number of processors increases, the snooping activity increases as well. When the workload on the system increases, the data traffic also increases. Therefore, a bus can be a limiting factor in terms of performance and scaling.

The IBM SMP is designed in a different way. There is still a bus for the snooping activity and the addressing. But a new component has been added for data transfers. That component is a switch called a Data CrossBar (DCB). The switch allows point-to-point connections between a processor and another processor or between a processor and the memory. It also allows several simultaneous transfers.

With such a technology, once the data is found, a point-to-point transfer can be done from the source to the requestor through the switch. While the switch is busy doing the data transfer, the bus is free for another processor request for data.

A switch has the following advantages:

- It removes work from the snoopy bus.
- It can transfer data among several units simultaneously.
- Connections are point-to-point, which allows a greater speed.

Figure 34 illustrates the use of the switch for data transfers.

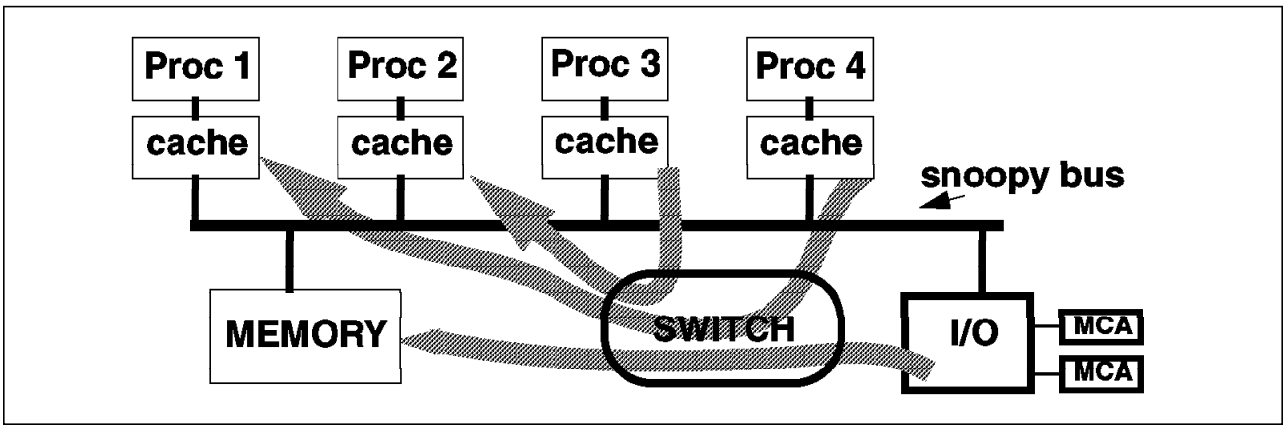


Figure 34. Using a Switch for Data Transfer

3.2.3 SMP Architecture Description

Figure 35 describes the SMP architecture. This figure shows an SMP system with four CPU boards (each CPU board having two processors), the Data CrossBar switch, the 16 memory modules, the System Memory Controller (SMC) and the I/O Controller.

The data path between a CPU board and the crossbar switch is 64 bits wide. The data path between the crossbar and the memory is 256 bits wide. The I/O Controller is able to drive two independent MCA buses, each operating at 160 MB/s (160 MB/s is the peak rate; each MCA bus can sustain 110 MB/s). The memory is totally shared by all of the processors. Each processor can access any memory module through the Data CrossBar switch.

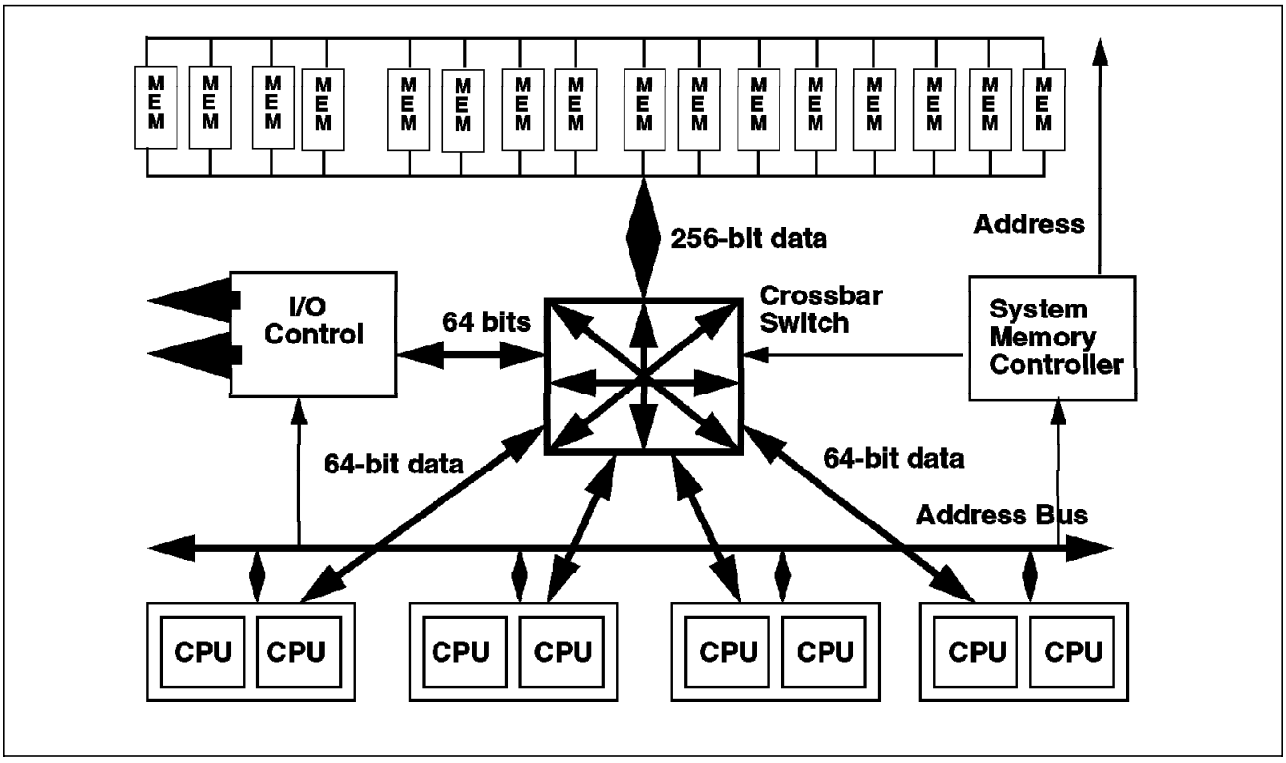


Figure 35. SMP Architecture

3.2.4 Memory Subsystem

We said previously that the memory subsystem is key. The memory subsystem is in fact composed of the System Memory Controller (SMC), the Data CrossBar (DCB) and the memory array (or physical memory).

In order to improve the overall performance of the memory subsystem, each component has to be improved. Let us see what kind of technique is used.

3.2.5 Memory Array Interleaving

One of the techniques used to improve the memory latency is to interleave the memory. This is not specific to the IBM SMP; this technique is generally used by the industry. But the IBM SMP implements a very high level of interleaving.

Following is an explanation of memory interleaving:

Let us suppose that the system has four 256 MB memory modules. Without any interleaving, each memory module would store a contiguous block of physical address space. In our example, the first module would store data for physical addresses *0x0* through *0xFFFFFFF*; the second would store *0x10000000* through *0x1FFFFFFF,* and so on.

While this is simple, the main disadvantage is that accesses to adjacent addresses, which often happen within a short time due to spatial locality, will go to the same memory module. The memory module will be busy and will not be able to handle the request. Busy modules will increase the overall memory latency.

To overlap the memory cycle times better, memory is interleaved such that address space is striped across the modules. In this case, the amount of contiguous memory stored in a module is usually equal to the cache line size or the cache sector size. Since the cache sector is 32 bytes in our SMP implementation, the data for physical addresses *0x0* through *0x1F* would be stored in the first module, addresses *0x20* to *0x3F* in the second, addresses *0x40* to *0x5F* in the third and addresses *0x60* to *0x7F* in the fourth. Then, the addresses would wrap around to the first module again for addresses *0x80* to *0x9F*, and so on. The exact way in which the physical address space is interleaved among the modules is invisible to the software.

In summary, memory interleaving is a technique developed to allow simultaneous access to adjacent areas of memory. When interleaving is done with four modules, we say it is a four-way interleaving.

The IBM SMP system has a very high level of interleaving. According to the memory configuration, interleaving can be a one-way, two-way, four-way, eight-way and 16-way.

The most important point here is that interleaving is automatically optimized by the system at boot time, according to the memory configuration. Interleaving is done down to the cache sector level that is 32 bytes. This ensures minimum contention within the memory subsystem between processors to guarantee minimum latency.

In the IBM SMP, the memory array is divided into memory banks. A memory card can have one, two or four memory banks. The interleaving level (one-way, two-way, four-way, eight-way, 16-way) depends on the number and the size of

banks installed on the system, not on the number of memory cards. Since the IBM SMP can have up to four memory cards, a system can reach up to 16 banks. The size of a bank can be 32 MB, 64 MB or 128 MB. The new 1GB memory card for G40 has 4 banks of 256 MB.

A system can also have banks with different sizes. In this case, the system will automatically optimize the interleaving scheme. Figure 36 shows an example of a system having two 32 MB banks and two 64 MB banks. In this case, the system will create different zones; one will be four-way interleaved, the other one two-way interleaved. Note in this example that two sets of memory chips participate in two different interleave zones.

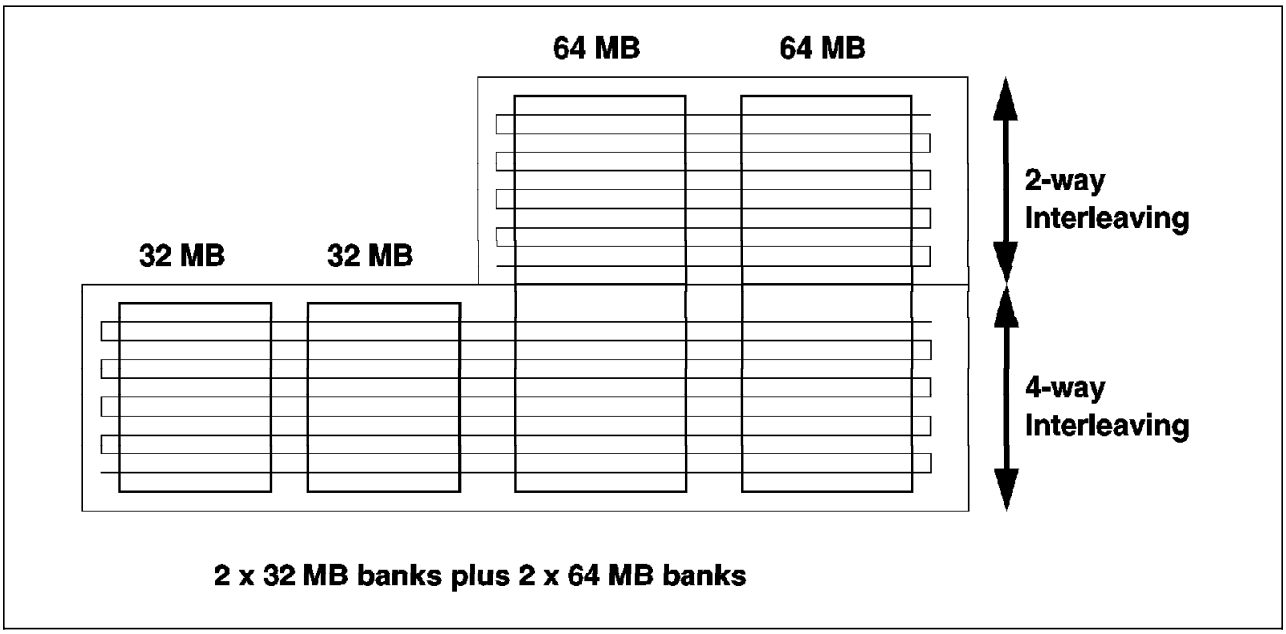


Figure 36. Interleaving Optimization

Again, the interleaving technique on the IBM SMP system is extremely advanced; memory is viewed as a random collection of 16 banks of 256 bits.

3.2.6 Memory Array Characteristics

The memory array bandwidth is 288 bits wide (256 + 32 ECC). A system can have up to 16 memory banks. Banks of memory use industry-standard JEDEC (Joint Electronic Device Engineering Council) 1 MB, 2 MB and 4 MB x 72-bit ECC DIMMs (Dual In-Line Memory Modules). These memory modules use 4 Mb or 16 Mb memory chips.

Memory does not need to be in pairs. In the uniprocessor, where memory has to be in pairs or quad, this provides a memory bandwidth of 160 or 320 bits.

The DIMMs memory used on the SMP systems are 72-bit SIMMs, whereas the 128/256 MB SIMMs in the uniprocessor are 80-bit SIMMs. However, these uniprocessor SIMMs can be used on SMP machines, but a new memory card is required to be able to plug in these SIMMs modules.

When the old 128/256 MB SIMMs are used on an SMP, the last eight bits are ignored.

The memory subsystem is able to perform four memory error corrections (ECC) at the same time. The BUMP processor will wake up AIX if there is an address/memory failure syndrome and log the error and location.

The memory array consists of rows and banks of memory. A memory board is considered as a row. Up to four boards are supported in the system, except on the G40. Each board is considered as a row of the global memory array. Memory boards are increments of one.

Different memory boards have a different number of memory banks:

In the G40:

- The 32 MB, 64 MB and 128 MB cards have one bank.
- The 256 MB card has two banks.
- The 512 MB and 1 GB cards have four banks.

In the J40 and R40:

- The 64 MB, 128 MB and 256 MB cards have two banks.
- The 512 MB card has four banks.

Note: Different card sizes can be mixed in the same system (except on the G40, which has only one memory slot). If a memory bank has failed (memory error that cannot be corrected), the memory board can be degraded to run without that memory bank.

The interleaving scheme between banks and rows (low interleave and/or high interleave) is determined by firmware and controlled by the SMC according to the memory board′s configuration. The best interleaving scheme is adopted when there are two, four, eight or 16 banks of homogeneous DRAMS.

The following tables show memory combinations available for use in the SMP servers:

CARD SIZE	SIMM SIZE	NUMBER OF SIMMS	NUMBER OF BANKS
32 MB	8 MB	4	1
64 MB	16 MB	4	1
128 MB	32 MB	4	1
1 GB	64 MB	4x4	4

Table 3. MRE Memory Cards for the G40

CARD SIZE	SIMM SIZE	NUMBER OF SIMMS	NUMBER OF BANKS
R2 64 MB	8 MB	8	2
MR4 256 MB	32 MB	8	2

Table 4. R2 and MR4 Memory Cards for the J40 and R40

CARD SIZE	SIMM SIZE	NUMBER OF SIMMS	NUMBER OF BANKS
64 MB	8 MB	2x4	2
128 MB	8 MB	4x4	2
256 MB	16 MB	4x4	2
512 MB	32 MB	4x4	4

Table 5. NFX Memory Cards for the G40, J40 and R40

3.2.7 Crossbar Main Characteristics

The non-blocking Data CrossBar switch provides four CPU ports in the J40 and R40, and two CPU ports in the G40. One extra port is dedicated to the I/O subsystem, allowing up to two I/O channels.

The crossbar has been designed to support three generations of PowerPC processors (601, 604 and 620) with a scalability to eight 620 processors. This means that the crossbar bandwidth can support up to eight PowerPC 620 processors.

3.2.8 Crossbar Switch Interconnection

The idea of a crossbar is to provide multiple buses that can be in use simultaneously in order to reduce contention and to provide multiple memory banks that can be operated in parallel, which increases overall memory bandwidth.

Each circle in the crossbar array represents a switch that can be turned on or off by the hardware to connect the two intersecting buses temporarily. Normally, all switches are off until a processor or I/O device needs to access the memory or another processor. For example, if processor card 1 needs to transfer data to memory, the switch is turned on. When the access is complete, the switch is turned back off.

If there are other processors that need to access the same memory bank at the same time, then the crossbar hardware arbitrates this request in much the same way as a standard bus, allowing one memory access at a time to the memory bank.

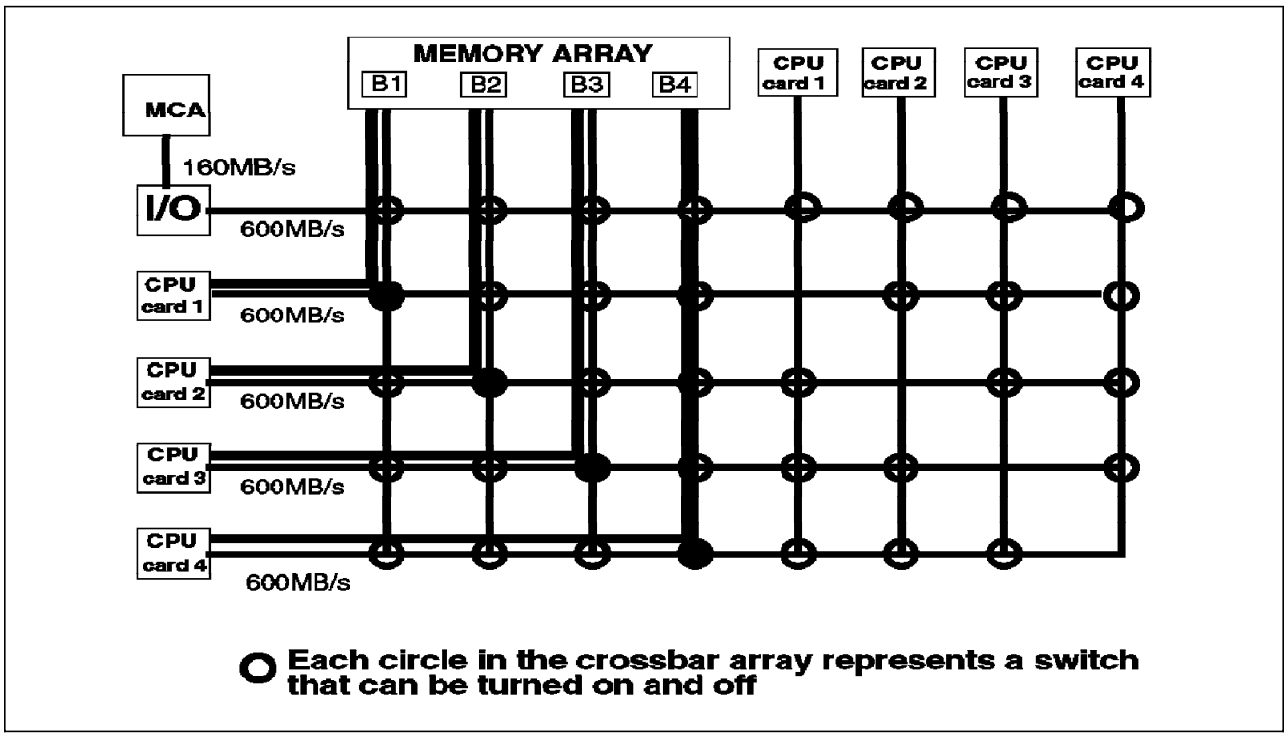


Figure 37. Crossbar Switch Interconnection

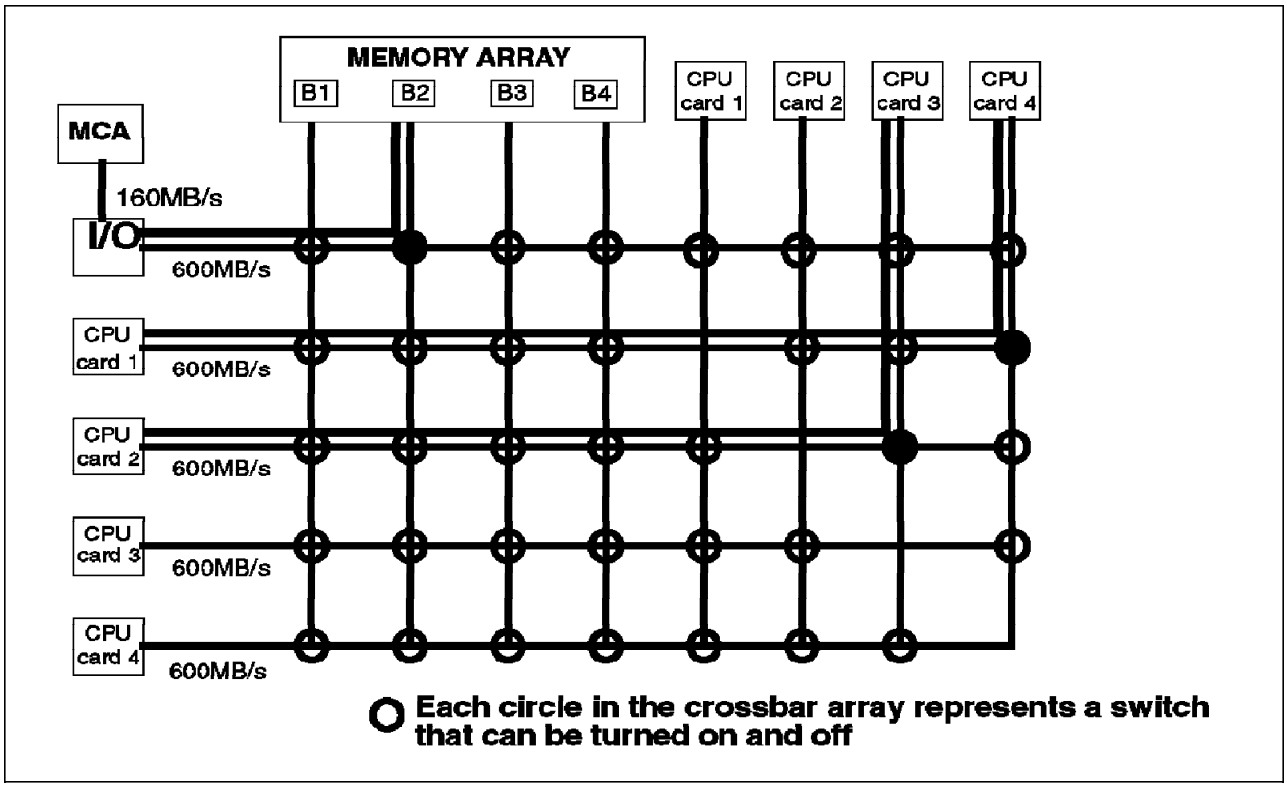


Figure 38. Crossbar Switch Interconnection

Figure 38 shows the interconnection between the I/O subsystem and the memory, CPU cards 1 and 4 and CPU cards 2 and 3.

3.2.9 Crossbar Architecture

Figure 39 shows the architecture of the crossbar in a configuration with four CPU cards (eight processors).

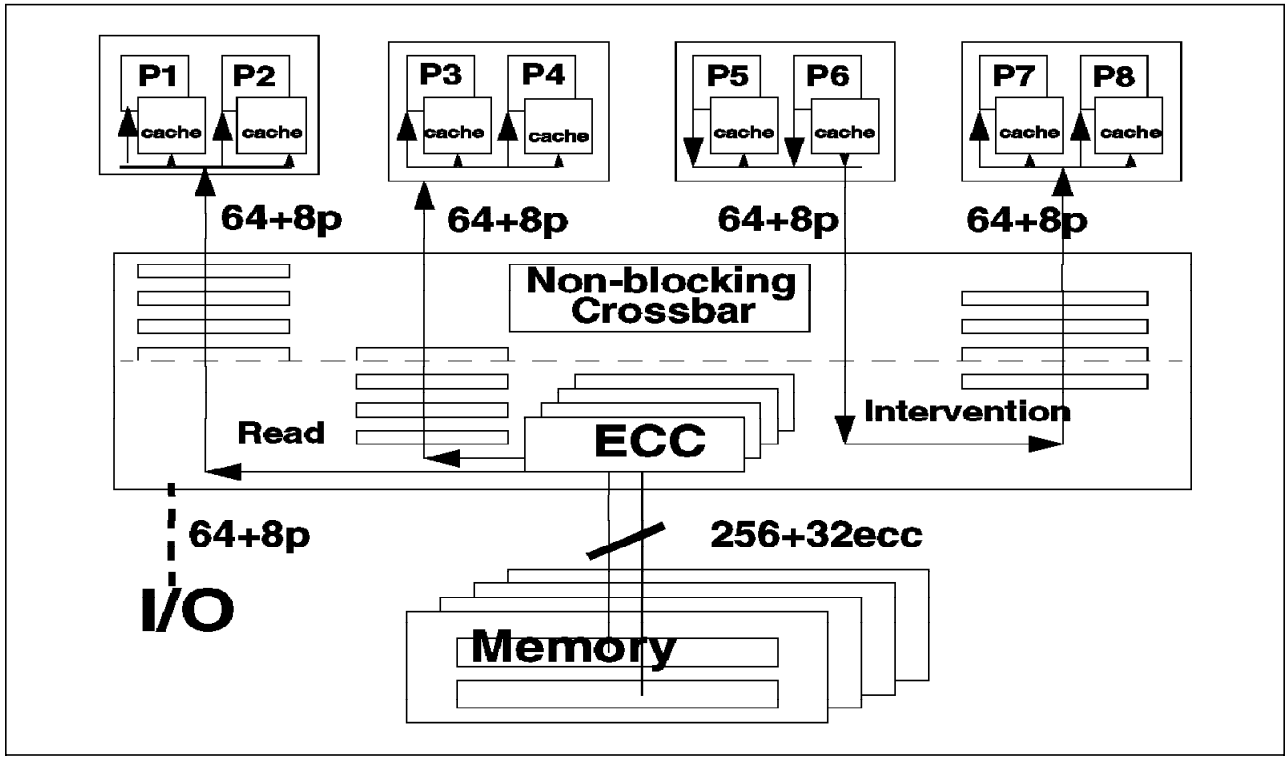


Figure 39. Crossbar Architecture

The crossbar data path width at the memory array level is 256 + 32 bits (32 bits used for ECC), while the data path width for each CPU and I/O is 64 + 8 bits (eight bits for parity).

At the memory level, the system can transfer 32 bytes every three cycles from the memory to the crossbar and can sustain such a throughput.

At the CPU card level, the system can transfer eight bytes (64 bits) every cycle, and it can sustain that rate.

When a transfer from the memory to a CPU occurs, the crossbar establishes a connection between the memory bank and the CPU card. Once 32 bytes are transferred from the memory to the crossbar, it takes four cycles to transfer these 32 bytes to the processor card. A second transfer from the memory to another CPU card can start while the first transfer is being accomplished. There is a one-cycle overlap between both memory-to-CPU transfers.

The fully overlapped, non-blocking Data CrossBar switch (DCB) provides each processor card with its own direct path into memory. In an eight-way system, each of the four processor cards can access memory concurrently.

The DCB operation is driven by a command bus provided by the SMC (System Memory Controller) to establish the proper data path interconnections between data clients.

The non-blocking Data CrossBar switch carries data between processor caches and memory and between caches and caches. Its four-deep pipelined architecture is used to provide the highest degree of concurrence in memory and cache operation, it operates in conjunction with the data crossbar.

3.2.10 Crossbar Performance Characteristics

The processor clock speed in the x40 servers is 112 MHz, and the bus speed is 75 MHz.

Since the physical memory interface is 32 bytes wide, and since it is possible to transfer 32 bytes every three clock cycles, the sustained transfer rate at the memory level is 800 MB/s (32 x 75/3 x 10**6 bytes/s).

To be able to reach an 800 MB/s rate, you need at least four banks of memory because each memory bank has a bandwidth of 267 MB/s. So you will have to populate your system with four banks to have the full capability. Two two-bank cards or one four-bank card is required.

The 800 MB/s rate for memory is maintained by the use of buffers in the switch which enable several operations to be pipelined.

Each CPU port is eight bytes wide and is capable of transferring eight bytes per clock cycle; thus, the sustainable rate at the CPU card level is 600 MB/s (8 x 75 x 10**6 bytes/s).

If the system is equipped with eight processors (four CPU cards), you can simultaneously have two memory-CPU card transfers and one cache-to-cache transfer (called intervention). See Figure 39. Each memory-CPU card transfer has a 600 MB/s rate. These two transfers overlap during one cycle. This gives a 1200 MB/s peak rate for the memory-CPU card transfers. At the same time, a cache-to-cache transfer can occur between processor card 3 and 4 at a 600 MB/s rate.

Therefore the crossbar peak rate is 1800 MB/s.

This brings to light an important distinction: 1800 MB/s is the crossbar peak rate. This is not the memory bandwidth; that is 800 MB/s. This distinction is not necessary in bus memory subsystems since all traffic uses the same bus. However, in a switch memory subsystem, this distinction becomes important. Although the individual bus bandwidth is 800 MB/s, several transfers can take place at once, giving the peak rate of 1800 MB/s.

3.2.11 System Memory Controller

The System Memory Controller provides systemwide arbitration functions; it orchestrates all bus and crossbar operations by issuing crossbar commands at appropriate times so that data transfers occur synchronously with system bus control operations. It issues simple commands, such as transfer port A to port B to the crossbar. The crossbar blindly executes these commands with fixed latencies.

3.2.12 Crossbar Operations

This is a description of the different crossbar operations. Figure 40 is an illustration of these operations. Note that in this description, each component connected to a port of the crossbar is referred to as a node.

- **(a) Memory mode:**

  A processor or an I/O node is routed to the Memory Array (MA). This is the interconnection used to support memory read or write operations between a processor and the memory or between the I/O subsystem and the memory.

- **(b) Intervention - RWNITM:**

  This operation happens when a processor wants to read data with no intent to modify it (RWNITM: Read With No Intent to Modify) and gets a modified snoop response. In other words, another processor has that data in its own cache and has modified it. Since the operation is a RWNITM, the processor will take data from the other processor cache, and the memory will be updated. Data sourced from the cache-line owner is presented to the reading agent and written back to the MA.

  If the reading agent and intervening CPU are within the same node (same CPU card), data is looped back at node level and written back to memory.

- **(c) Intervention - RWITM:**

  A processor or an I/O node (reading node) is routed to another processor or I/O node. This is the case in which a RWITM (Read With Intent To Modify) operation gets a modified snooped response; data sourced from the cache-line owner is presented only to the reading agent and is not written back to memory. Since the reading node wants to modify the data, updating the memory is not necessary.

- **(d) Programmed I/O mode - PIO:**

  A processor node is routed with an I/O node. This is the case of PIO operations: data is exchanged just between a processor and an I/O agent.

- **Memory Mapped I/O mode:**

  A processor node is routed with I/O nodes (slave nodes) which are mapped into the memory space. Examples include boot ROM (Read Only Memory), NVRAM (Non-Volatile Random Access Memory), system registers and so on.

Intervention is the action taken when the owner of modified cached data detects a snoop request for that data. The intervention is signalled by the modified MESI state response and followed by the data being sent to the requester and, potentially, to memory.

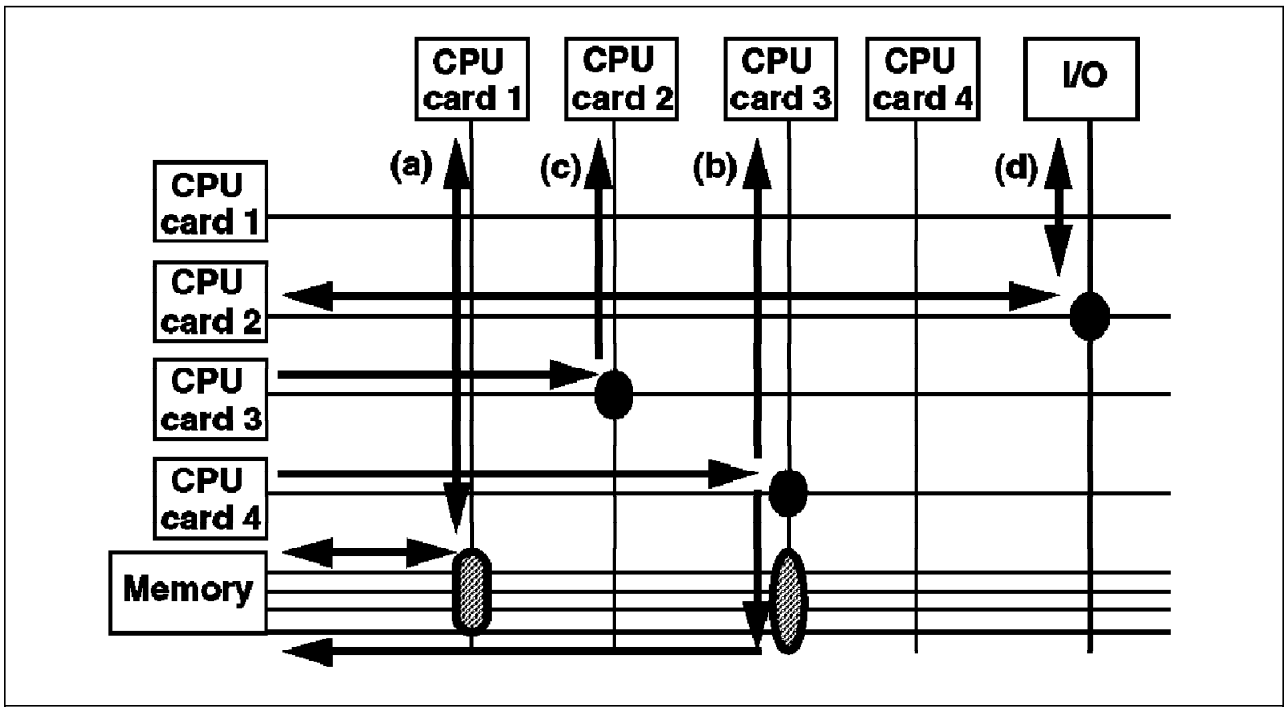


Figure 40. Crossbar Operations

3.2.13 MCA I/O bus

The MCA bus in an IBM SMP server is able to run at 160 MB/s, allowing high-speed adapters like the Fiber Channel Switch (FCS) or Serial Storage Architecture (SSA) to use 160 MB/s MCA mode. This is achieved by multiplexing the 32-bit address bus and then coupling this with the 32-bit data bus, thereby doubling the data path to 64 bits. This increases the transfer rate from 40 MB/s to 80 MB/s, 160 MB/s streaming uses a similar method, but the clock cycle is halved.

3.2.14 Interrupt Processing

Interrupt processing on an SMP system has the additional complication of having to decide which processor an interrupt should go to. The following section describes how this decision is made:

Each Micro Channel bus has 16 separate interrupt lines. Devices can share them or use them separately. Relatively slow devices, for example multiport async adapters, usually share a single line, while very fast I/O devices require dedicated lines. When a device wants to own one of the 16 interrupt lines, the device must be configured to avoid conflicts with other devices. Since there are two MCA buses on SMP systems, there are 32 interrupt lines.

Each of the 32 interrupt lines has its own software-assignable priority, with 255 possible priority levels. The interrupt level of a line is a software assignable characteristic that is loaded into a table inside the RS/6000 interrupt controller, it is not something permanently attached to particular interrupt lines coming in off a Micro Channel bus.

In addition to a priority, each of the 32 lines also has a software-assignable setting that gives *either*:

1. a specific processor number that identifies one and only one processor to which that interrupt will ever be directed (this is the funnel mode); *or*
2. an indicator that any processor can be interrupted by that interrupt line (MP safe mode).

The interrupt implementation also maintains its own copy of the priority table (that is, the level at which every processor in the system is operating). This is used in case 2 above, when any processor can be interrupted.

When an MP safe interrupt occurs, the system must decide which processor should receive it. The processor that is running at the lowest priority will usually be interrupted unless even this processor is running at a higher priority than the interrupt. In this event, the interrupt will be queued at the requesting device until a processor becomes available.

This implementation allows I/O work to be sent to the processor currently working at the lowest priority; it takes place semi-automatically using hardware support.

Processor-to-processor (p-p) interrupts behave in the same way as I/O interrupts, with the exception that if a p-p interrupt is queued, the requesting processor can cancel the interrupt. This is not necessary for I/O interrupts, as devices do not cancel their interrupts.

3.2.15 Crossbar Advantages Summary

A crossbar allows point-to-point connections between its different ports (CPU cards, I/O) and the memory array in two directions; thus, the bandwidth is not shared. When a transfer occurs between two ports (6 available), all the bandwidth is available. Several transfers can occur at the same time. For example, one I/O-to-memory transfer can occur at the same time as two CPU-to-CPU transfers.

When several transfers occur simultaneously, the high degree of overlap provided by the crossbar allows the peak throughput to be very high (1800MB/s).

When transferring data through a bus, the latency depends on how much the bus is loaded. With a crossbar, the latency is fixed, meaning that loading data from memory takes a determined amount of time.

3.3 Architecture Implementation

Figure 41 shows the physical implementation of the SMP architecture. The following is a description of the different components:

- **Multiprocessor Board (MPB):**

  The base Multiprocessor Board (MPB) acts as a back-plane to host the CPU boards, the memory boards and one I/O daughter board (IOD).

- **Processor Daughter Board (CPU) Module:**

  Each PowerPC 604-based CPU module includes two microprocessors and their associated caches. L1 cache is on the processor chip itself, while L2 is on the board. There is physically one L2 cache per processor. Each L2 cache is divided in two parts: the TAG which keeps track of the physical address of the data stored in L2 and the L2 cache itself, containing data.

It will be possible to upgrade a system based on 604 processors to one based on 620 processors.

Note: Mixed 601 and 604 configurations are not supported.

- **Input/Output Daughter Board (IOD):**

  The IOD acts as the bridge between the MCA buses and the CPUs or the memory array. Two MCA buses operating a 160 MB/s (peak rate) are supported. Also, the IOD hosts the SystemGuard service processor and native I/O (serial ports, parallel ports and so on).

- **System Memory Controller (SMC):**

  The SMC acts as the Multiprocessor Board System Bus (MPB-SysBus) arbiter for four processor nodes and the two IONIAN address bus requests. It also acts as the MPB-SysBus snoop Status/Request collator and dispatcher, as the data path controller for the four Data CrossBar (DCB) chips and as a dynamic RAM controller.

- **Cache Controller Address (CCA):**

  The CCA (one chip) manages the entire address/control bus protocol, directly interfacing the processor address/control bus and L2 address/control for both processors on the same CPU board.

- **Cache Controller for Data (CCD):**

  The CCD (four chips) allows the processor to access the L2 cache and drives the 64-bit data bus to the Data CrossBar (DCB).

- **L2 TAG**

  The L2 is a storage array which maps a cache directory tag (a tag is an entry in a cache line) and its associated cache line.

- **L2 Static Random Access Memory (SRAM):**

  The L2 SRAM is a Static Random Access Memory. The L2 SRAM contains the actual data stored in L2. The L2 SRAM for the first processor is on one side of the processor card, while the second L2 SRAM, for the second processor, is on the other side of the card. The L2 SRAM consists of nine chips, 64 K x 18 or 32 K x 18, depending on the size of the cache.

  Note that the 620 chip will include an embedded L2 cache controller that interfaces with the SRAM chips.

- **Data CrossBar (DCB):**

  The DCB is integrated in multiple (4) ASICs (Application Specific Integrated Circuits, 4 x 16 bit slice). It functionally interconnects the MPB_SysBus (Multiprocessor Board System Bus) data path to the memory subsystem, to the CPU boards and to the I/O board.

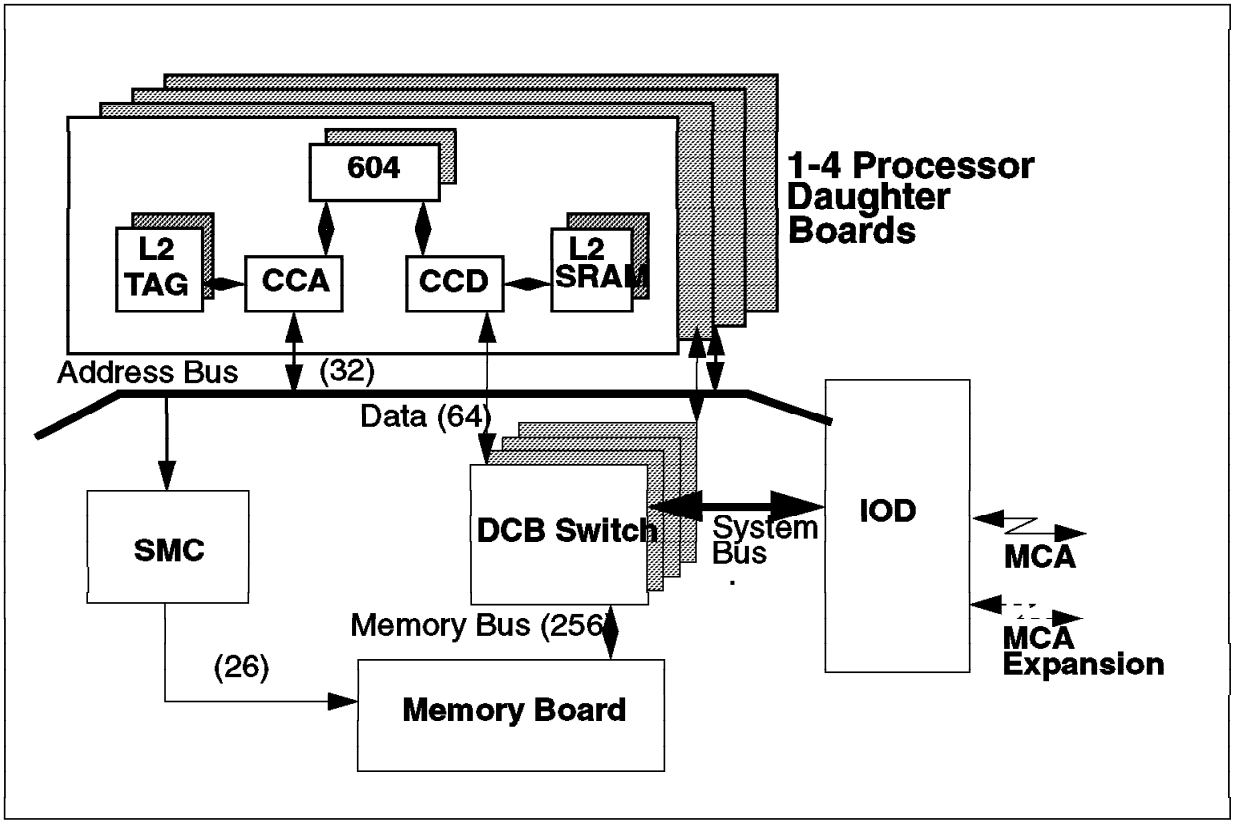


Figure 41. SMP Architecture Implementation

Chapter 4. SMP Servers Hardware Features

This chapter introduces the main hardware features of the IBM RS/6000 SMP Servers. It highlights some of the hardware specifics that might help you in configuring or implementing these systems.

4.1 IBM RS/6000 SMP Servers

The RS/6000 SMP server range consists of three models, the G40, J40 and R40. The G40 is a mini-tower model, the J40 is a deskside model and the R40 is a rack drawer. All these servers are based on PowerPC 604 technology. They are supported by AIX V4.1.4 with 604 SMP updates or later, and AIX V4.2.

The three SMP models incorporate a Data Crossbar (DCB) or switch. This provides a dedicated high-performance communication path between the processors and the main memory, resulting in a very high scalability factor.

The SMP servers also provide a service processor called SystemGuard. This service processor has its own firmware. It is responsible for controlling the SMP hardware at boot time and when AIX is running. It also allows local or remote power-on/off, diagnostics, reconfiguration and maintenance of the system. For more information on SystemGuard, refer to Chapter 5, “SystemGuard” on page 105 in this redbook.

Another feature offered with the SMP servers is the Cluster Power Controller (CPC). Even though the CPC was introduced with the RS/6000 SMP models, this feature is not specific to the SMPs. In the case of SMP systems, the CPC is a solution which allows you to have one BUMP Console and one Service Console for several SMPs. This allows the system administrator to centrally administrate several SMP systems and centrally control their power.

The SMP servers are intended to be commercial servers. Only the G40 can support a graphic display, a keyboard and a mouse. The J40 and the R40 do not support keyboard, mouse and graphic displays. On these systems, graphical applications can be displayed through the use of an Xstation.

All three models have three serial ports. The G40 has only two physical ports; so the first port can be split into serial port S1 and S2 through a splitter cable.

Readers are welcome to browse the RS/6000 home page for more information at `http://www.rs6000.ibm.com`.

4.2 Model G40 Server

The RS/6000 Model G40 is the smallest SMP server in this product line. It comes in a mini-tower cabinet.

The G40 comes standard with one 32-bit PowerPC 604 processor, L1 cache consisting of 16 KB for data and 16 KB for instructions, 0.5 MB of L2 cache, 64 MB of memory, a SCSI-2 Fast/Wide disk, two disk bays and two media bays. A single system image is provided via AIX V4.1.4 with 604 SMP updates or AIX V4.2. Despite its small size, the G40 can be upgraded to a two-way or four-way SMP and the memory increased to 1GB. The system has one memory slot and

two processor card slots. If the initial G40 is ordered with a single processor, then when another processor is needed, it will be replaced by a two-way processor card occupying one slot. Thus, for a four-way SMP model, there will be two two-way processor cards, filling up all available CPU slots in the G40. The server has a total of six microchannel slots, five of which are available for optional adapters.

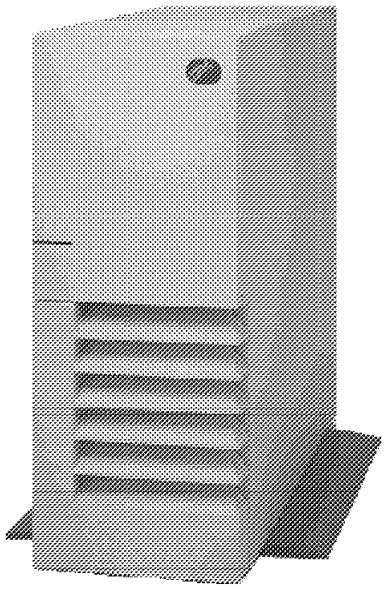


Figure 42. IBM RS/6000 Model G40 SMP Server

Because of its small size and compactness, the G40 can be configured as a stand-alone desktop system or as a high-performance server in a commercial environment. The G40 model is a machine type 7012, which is the same as the desktop models (series 3xx).

4.2.1 Standard Configuration for the G40

The standard configuration for the G40 is the following:

Microprocessor	: Single 112 MHz PowerPC 604 Processor
Level 1 (L1) Cache	: 16 KB data / 16 KB instruction
Level 2 (L2) Cache	: 0.5 MB
RAM (memory)	: 64 MB
Memory bus width	: 256 bits
Memory slots	: One
Diskette drive	: 1.44 MB 3.5 inch diskette drive
Internal disk drive	: 2.2 GB SCSI-2 Fast/Wide SE
CD-ROM drive	: Quad-speed
Disk/media bays	: Four (three available)
Expansion slots	: Six 160 MB/second Micro Channel (five available)
Ports	: One parallel and three serial
Service Processor	: IBM SystemGuard

Power Control Port : One RS-485 for connecting the G02 expansion unit

System Dimensions : 17.75” H x 6.9” W x 28.2” D (450 mm x 173 mm x 613 mm)

System Weight : 43 lbs to 55 lbs (19 Kg to 25 Kg)

4.2.2 System Expansion

The G40 expansion capabilities are the following:

RAM : Up to 1 GB.

Microprocessor : Up to four 112 MHz PowerPC 604 processors.

Internal disk storage : Up to 13.5 GB using the optional disk/media bays.

Total disk storage : Up to 121.5 GB using four G02 Expansion Cabinets. Up to 350 GB using five IBM 7134 Model 010 High-Density SCSI-2 Disk Subsystem. By utilizing SSA disk technology, the storage can be increased to over 1TB.

4.2.3 Select and Optional Features

The following is a representative list of available features and products that are supported in the G40:

Memory

- Expandable to 128 MB, 256 MB, 512 MB, 768 MB and 1 GB

Internal Storage

- CD-ROM Drives
- Disk drives
  - 2.2 GB SCSI-2
  - 4.5 GB SCSI-2

Tape Drives

- 5/10 GB, 8mm
- 4/8 GB, 4mm
- 1.2 GB, 1/4 inch cartridge

SCSI interfaces

- SCSI-2
- SCSI-2 Differential
- SCSI-2 Fast/Wide
- SCSI-2 Fast/Wide Differential

Adapters and controllers

- 128-Port Asynchronous Controller
- 4-Port Multiprotocol Communications Controller
- 8- and 16-Port Asynchronous Adapters
- ARTIC960 Co-Processor Adapter
- Block Multiplexer Channel Adapter

- Digital Trunk Adapter (single and dual)
- Ethernet High-Performance LAN Adapter
- FDDI Fiber Adapter (single and dual ring)
- High-Performance Subsystem Adapter
- IBM Auto Token-Ring LANstreamer 32 MC Adapter
- IBM Ultimedia Audio Adapter
- Realtime Interface Co-Processor Portmaster Adapter
- S/390 ESCON Channel Emulator
- TURBOWAYS 155 ATM Adapter
- X.25 Interface Co-Processor/2 Adapter
- System/370 Host Interface Adapter
- High-Performance 4-port SSA Adapter

4.2.4 Supported Devices

The following list of devices is only a representative sample. For a complete and detailed list of supported devices, contact your IBM Marketing Representative or your IBM Business Partner.

External Storage Devices

- Rewritable optical drives
- CD-ROM drives
- Disk drives
  - 1 GB, 1.1 GB, 2 GB, and 2.2 GB SCSI-2
  - 4.5 GB SCSI-2 Fast/Wide
  - 4.5 GB SCSI-2 Fast/Wide (1-inch high)
- Tape drives
  - 5 GB, 8 mm
  - 4 GB, 4 mm
  - 525 MB and 1.2 GB, 1/4 inch cartridge

External Storage Systems

- Disk array subsystems
- High-performance disk drive subsystems
- High-density SCSI-2 disk subsystem
- Model G02 Expansion Cabinet
- Optical libraries
- SCSI-2 disk expansion units
- Tape subsystems and libraries

Other Devices

- 7318 Serial Communications Network Server Model P10 and S20

- Ethernet 10Base-T and 10Base-2 Transceivers

Displays

- P50, P70, P200 and P201 color monitors

Terminals

- A wide range of ASCII terminals, as well as X11 terminals

Printers and modems

- A wide range of IBM and non-IBM printers, plotters and modems

4.2.5 Additional Information on the G40 Server

Figure 43 shows the front panel and outer casing of the G40.

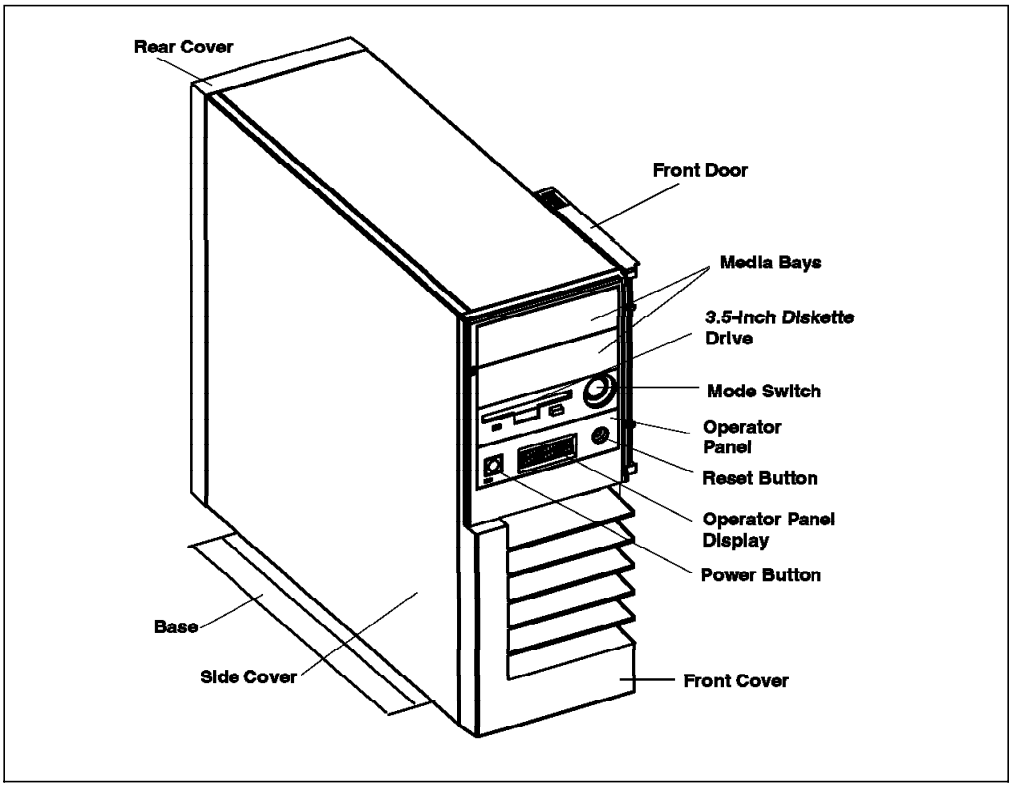


Figure 43. Model G40 SMP Server

The Operator Panel is different from the uniprocessor systems. It has a 2 x 16-digit LCD used for:

- Event indications and problem reporting during power-on tests and configuration steps
- Progress and command indication when loading diagnostics
- Problem reporting during diagnostics when a console is not available
- Checkstop when the machine cannot recover from a checkstop
- Power problem reporting
- Run-time problem reporting

Figure 44 on page 76 gives an internal view of the G40 once the covers have been removed. The procedure for removing covers is the following:

- Remove the front cover.
- Remove the side cover by first removing the retaining screws located on the front of the side cover.
- Remove the rear cover by first loosening the retaining screws located inside the unit in the rear.

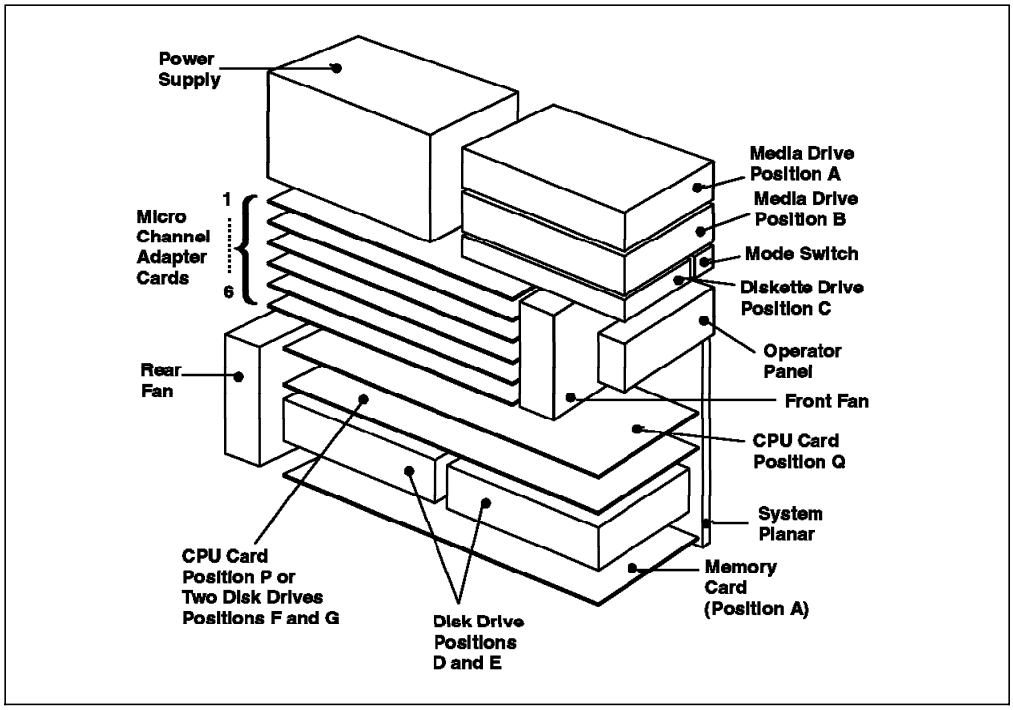


Figure 44. Internal View of the Model G30 or G40 Server

You can see in this picture that the G40 has two physical locations for processor cards (position Q and P).

On top of the system are located two media bays. The upper media bay (position A) can be used for an optional disk. It requires feature code 6511. Note that this feature code is added automatically when more then two disks are requested.

Thus, by using feature code 6511, you can have up to three disks installed on the machine.

Note: The position P can no longer be used in the G40 to install disk drives as in the G30.

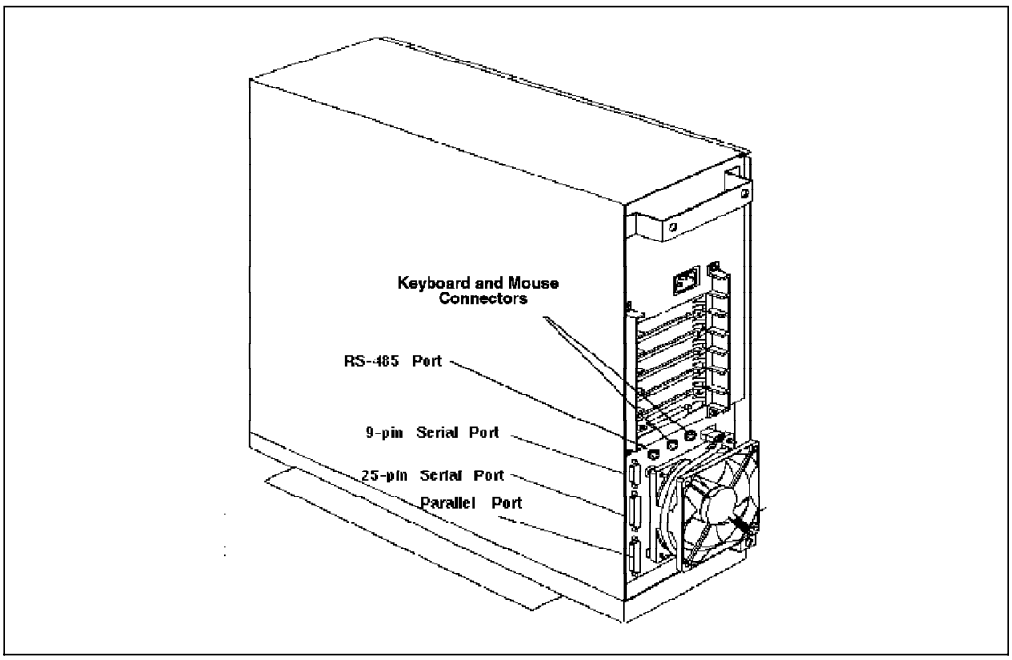


Figure 45. Rear View of the Model G40 Server

You can see that the G40 has three ports located at the back:

- One 9-pin serial port
- One 25-pin serial port
- One parallel port

The 25-pin serial port can provide two EIA RS-232 ports by installing a splitter cable. This splitter cable has the part numbers 31F4126 or 31F4590 (feature code 3107). Normally the S1 port is used to connect the BUMP (Bring Up Micro-Processor) Console. The S2 port is used to connect a regular ASCII terminal or a remote Service Console.

The BUMP Console, connected to the S1 port, is normally used as the system console. This console can be a local terminal, a remote terminal or a terminal emulator. It also allows the performance of offline maintenance.

The Service Console, connected to the S2 port, allows support personnel to remotely perform diagnostics or maintenance from AIX or from the SystemGuard utilities, such as the SystemGuard MAINTENANCE MENU or the SystemGuard STAND-BY MENU. It also allows remote power control.

For more information on how to use the BUMP Console and the Service Console, refer to Chapter 5, “SystemGuard” on page 105 in this redbook.

In the G40, all the internal media and disk devices have fixed SCSI addresses. The following diagram shows the SCSI device locations and their addresses.

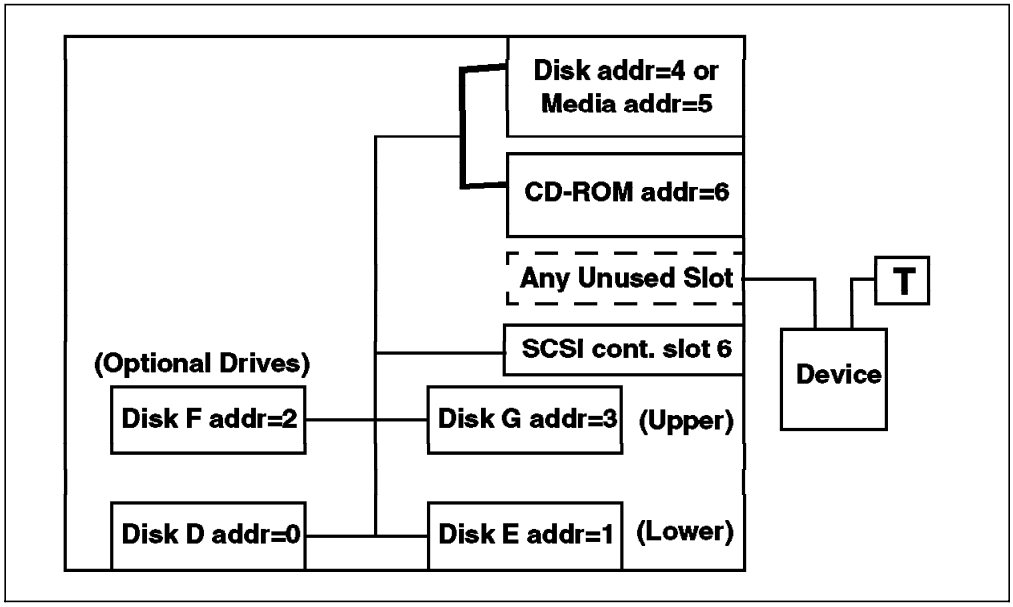


Figure 46. SCSI Device Addresses and Location for the G30 or G40 Server

Note: Dummy blank plates should be used to protect the unit from dust and debris if an adapter is removed from a slot.

4.3 Model G02 Expansion Cabinet

The G02 expansion unit is used to increase the number of media devices and/or hard disks in the G40.

All hardware components inside the disk expansion unit are accessible from the left side after removing the expansion-unit covers.

The expansion unit is similar in design to the G40. A G40 can have up to four G02 expansion units. The G02 provides two disk/media bays and four disk bays. Each disk unit must be connected to the base unit through a dedicated SCSI cable.

The G02 expansion unit has the following features:

- Four disk bays
- Two media/disk bays
- Up to six 1.1 GB and/or 2.2 GB disks (if the two media bays are used for disks with feature code 6511)
- Maximum four 4.5 GB disks
- Media bays only support half-height devices

4.3.1 Installation

Looking at the unit from the front, the expansion units must be installed starting on the left side of the base unit. Appropriate cables, according to the system configuration, must be used to connect the expansion unit.

The G02 expansion cabinet comes standard with a power interface cable and a (single-ended) SCSI cable for attachment to the G40.

The 6-pin RS-485 port present in the rear of the G40 must be connected to the G02 for power control. Depending on the key position of the G02 expansion unit when the power is first introduced, the G02 can be powered on and off automatically with the G40, or the G02 can be powered up in a standalone mode. When the key is in the Normal position, the G02 power will be controlled by the G40. If the key is in the Secure position, the G02 will be set as standalone mode.

It is important that the key position of the G02 be set correctly before you plug in the wall plug of the G02 expansion unit, as this is the only time the G02 will determine the status of the key position of the G02.

The new expansion unit must then be recognized and configured by the operating system through SMIT.

Please refer to the *7012 G Series Service Guide* for detailed installation instructions. Also refer to the *7012 G Series Operator Guide* for configuration and disk location information. There is also a new *Supplemental Information Guide for 7012 G Series Models with Adapters, Devices, Cable Information and Diagnostic Information.* The part number for this book is 40H7073.

4.4 Model J40 Server

The RS/6000 Model J40 is a deskside system which provides for greater expandability than the G40.

It features four processor card slots and four slots for memory cards. As a result, the J40 is can support up to eight processors and 2048 MB of memory.

The system comes standard with two PowerPC 604 processors, 128 MB of memory, six available microchannel slots, three media bays, and seven disk bays. An optional expansion cabinet can be purchased to increase the total number of microchannel slots to 15 and to provide additional disk storage.

The J40 has internal hot-pluggable SCSI disk drives. This means that if a disk were to fail, it could be removed and replaced while the rest of the system continues operation.

The SystemGuard service processor is also a standard feature of this model. It continuously monitors the hardware as well as the software.

The J40 can also provide graphics capabilities. This requires the keyboard/mouse attachment card, feature code 2734 and a Power GXT150M graphics adapter card, feature code 2650. Installing this feature requires two empty MCA slots for the adapters.

The model J40 is well suited for enterprises that anticipate the need for scalable, uninterrupted long-term growth. Starting with two processors, the system can be upgraded to a four-way, six-way or an eight-way SMP.

For the existing 5xx RS/6000 systems, an upgrade path is offered to the model J40.

Figure 47 on page 80 shows the J40 base unit. The J40 model is very different from the existing 5xx models as far as the industrial design, packaging and expansion capabilities are concerned. The J40 is a deskside model so it has the machine type 7013, just like the 5xx models.

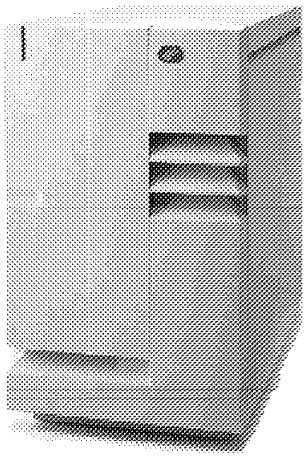


Figure 47. IBM RS/6000 Model J40 SMP Server

4.4.1 Standard Configuration

The standard configuration for the J40 is as follows:

Microprocessor	: Dual 112 MHz PowerPC 604 Processor
Level 1 (L1) Cache	: 16 KB data / 16 KB instruction
Level 2 (L2) Cache	: 1 MB
RAM (memory)	: 128 MB
Memory bus width	: 256 bits
Memory slots	: Four
Diskette drive	: 1.44 MB 3.5 inch diskette drive
Internal disk drive	: 4.5 GB SCSI-2 Fast/Wide, hot pluggable
CD-ROM drive	: Quad-speed
Disk/media bays	: Seven disk (all hot swappable); three media
Expansion slots	: Seven 160 MB/second Micro Channel (six available)
Ports	: One parallel and three serial
Service Processor	: IBM SystemGuard
System Dimension	: 24” H x 14.7” W x 29.5” D (610 mm x 360 mm x 750 mm)
System Weight	: 100 lbs to 150 lbs (43.5 Kg to 67.9 Kg)

4.4.2 System Expansion

The J40 expansion capabilities are as follows:

RAM	: Up to 2 GB.
Microprocessor	: Up to eight 112 MHz PowerPC 604 processors.
Internal disk storage	: Up to 36 GB using the optional disk/media bays.
Total disk storage	: Up to 99 GB using the J01 Expansion Cabinet with eight Micro Channel expansion slots and 12 disks and two media bays (all hot swappable). By utilizing SSA disk technology the storage can be increased to several TBs.

4.4.3 Select and Optional Features

The following is a representative list of available features and products that are supported in the J40.

Memory

- Expandable to 2G using 128 MB, 256 MB and 512 MB memory cards

Internal Storage

- CD-ROM Drives
- Disk drives
  - 2.2 GB SCSI-2
  - 4.5 GB SCSI-2 Fast/Wide

Tape Drives

- 5/10 GB, 8mm
- 4/8 GB, 4mm
- 1.2 GB, 1/4 inch cartridge

SCSI interfaces

- SCSI-2
- SCSI-2 Differential
- SCSI-2 Fast/Wide
- SCSI-2 Fast/Wide Differential

Adapters and controllers

- 128-Port Asynchronous Controller
- 4-Port Multiprotocol Communications Controller
- 8- and 16-Port Asynchronous Adapters
- ARTIC960 Co-Processor Adapter
- Block Multiplexer Channel Adapter
- Digital Trunk Adapter (single and dual)
- Ethernet High-Performance LAN Adapter
- FDDI Fiber Adapter (single and dual ring)
- High-Performance Subsystem Adapter
- IBM Auto Token-Ring LANstreamer 32 MC Adapter
- IBM Ultimedia Audio Adapter
- Realtime Interface Co-Processor Portmaster Adapter
- S/390 ESCON Channel Emulator
- TURBOWAYS 155 ATM Adapter
- X.25 Interface Co-Processor/2 Adapter
- System/370 Host Interface Adapter
- High-Performance 4-port SSA Adapter
- Power GXT150M Graphics Adapter

- Keyboard/Mouse Attachment Card

4.4.4 Supported Devices

The following list of devices is only a representative sample. For a complete and detailed list of supported devices, contact your IBM Marketing Representative or IBM Business Partner.

External Storage Devices

- Rewritable optical drives
- CD-ROM drives
- Disk drives
  - 1 GB, 1.1 GB, 2 GB, 2.2 GB, 4.5 GB SCSI-2
- Tape drives
  - 5 GB, 8 mm
  - 4 GB, 4 mm
  - 1.2 GB, 1/4 inch cartridge
  - 800 MB, 1/2 inch cartridge
  - 10 GB, 1/2 inch cartridge
  - 1/2 inch reel

External Storage Systems

- Disk array subsystems
- High-performance disk drive subsystems
- High-density SCSI-2 disk subsystem
- Model J01 Expansion Cabinet
- Optical libraries
- SCSI-2 disk expansion units
- Tape subsystems and libraries

Other Devices

- 7318 Serial Communications Network Server Model P10 and S20
- Ethernet 10Base-T and 10Base-2 transceivers

Displays

- P50, P70, P200 and P201 color monitors

Terminals

- A wide range of ASCII terminals, as well as X11 terminals

Printers and modems

- A wide range of IBM and non-IBM printers, plotters and modems

4.4.5 System Interface Board on the J40 Server

One system interface board is required on both the base and expansion units. In addition, a second system interface board is required on the expansion unit if any device is placed on the B bus. The second system interface board on the expansion unit is not involved in the power supply management.

The System Interface Board (SIB) provides four functions:

1. Power microcontroller - The power microcontroller helps the Bring Up Micro-Processor (BUMP) to monitor each unit and its power supply, assuring communication between the units.
2. Removable disks switch - This function enables you to remove one or more SCSI devices (disk drives and media devices) while the machine is still powered on and operational.
   The System Interface Board controls the power domains for all of the internal SCSI devices. The removable disks switch (RDS) removes both the +5 volts and +12 volts from the SCSI devices selected by the RDS function in SMIT.
3. System basic lines physical interface - It checks the asynchronous lines drivers and receivers (S1, S2, S3) and is the asynchronous lines bulkhead. The System Interface Board also is the bulkhead for the parallel port. The signals come from the I/O card through the unit back plane and reach the dedicated connector on the System Interface Board.
4. SCSI bulkhead - The System Interface Board is a bulkhead for the first (A) and second (B) SCSI buses, for the terminator or for the second external SCSI controller connection (dual initiator). The signals come from the SCSI controller through the unit back plane and reach the dedicated connector on the System Interface Board.

The System Interface Board (SIB) also provides the following connectors:

- Two 68-pin SCSI-2 F/W Differential connectors marked A for the upper connector and B for the lower connector (A for terminating bus A and B for terminating bus B).
- Two 6-pin mini-connectors for communication between the base unit and the expansion units. They are marked IN for the upper connector and OUT for the lower connector and use the RS-485 interface asynchronous protocol.
- Three RS-232 serial ports (25-pin connectors). Their typical use is:
  - S1 for the BUMP Console
  - S2 for a remote Service Console
  - S3 as an optional external Uninterruptable Power Supply (UPS)
- One parallel port

Figure 48 on page 84 shows the System Interface Board (SIB) which is located at the rear of the J40. You can see the two SCSI-2 F/W Differential connectors that terminate the internal SCSI-2 F/W Differential buses. These connectors must be terminated by a terminator. You also can see the three serial port connectors and the two RS-485 connectors used for daisy-chaining the expansions units.

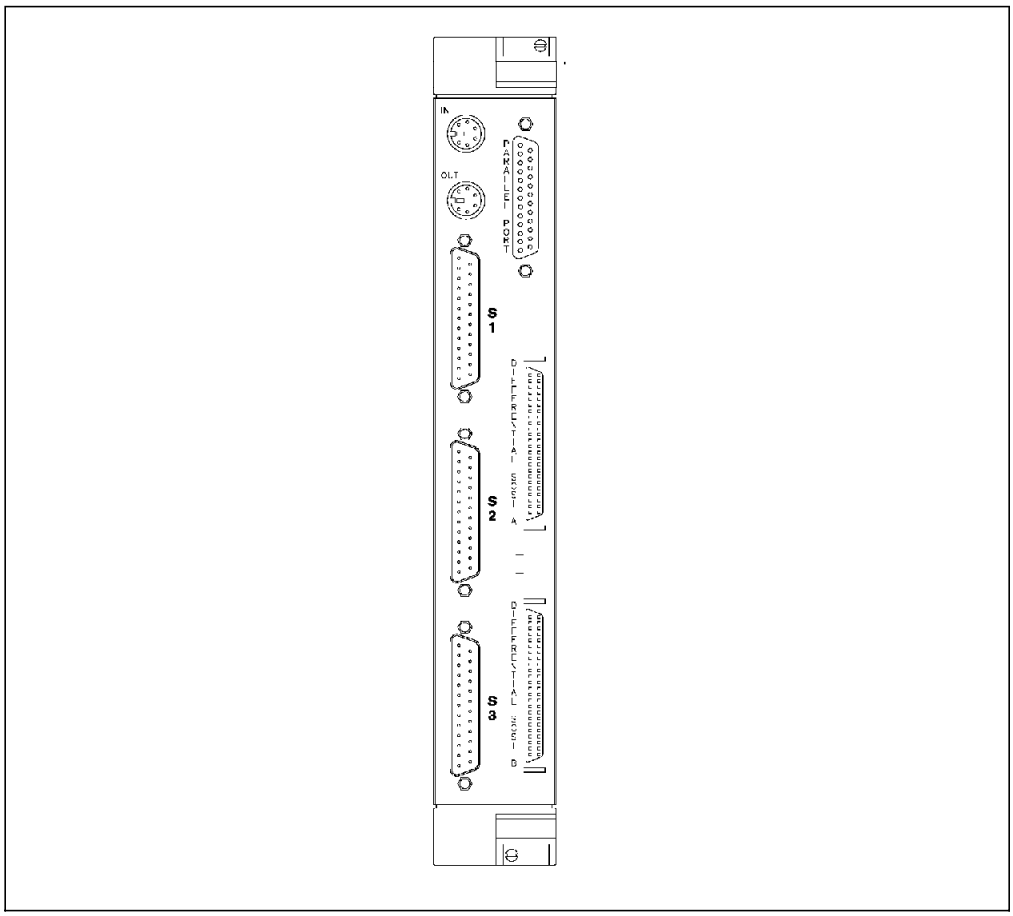


Figure 48. Rear View of the System Interface Board on J40 Server

4.4.6 High Removability Feature

All disk drives can be removed from, or inserted into, the appropriate drive position while the system is operating.

There are no restrictions relative to the drive position in which each disk drive can be installed. But there is a difference between the three disk drive positions in the front of the machine and the disk drive positions in the back. The three drives in the lower front portion of the base and expansion units have a High Removability feature. Each of these drive positions is equipped with a special lubricated connector that makes it possible to regularly remove and insert the corresponding disk drives.

A disk drive which has been previously installed in a normal slot can be installed in a slot featuring the High Removability feature, and vice versa.

Disk drives that need to be removed frequently - such as disk drives that contain confidential data that needs to be removed from the system and stored in a safe place regularly - should be installed in these positions.

Figure 49 on page 85 is an internal front view of the J40. You can see the three disk drives in the lower position and the three media bays. Two of the media bays can be used for disk drives.

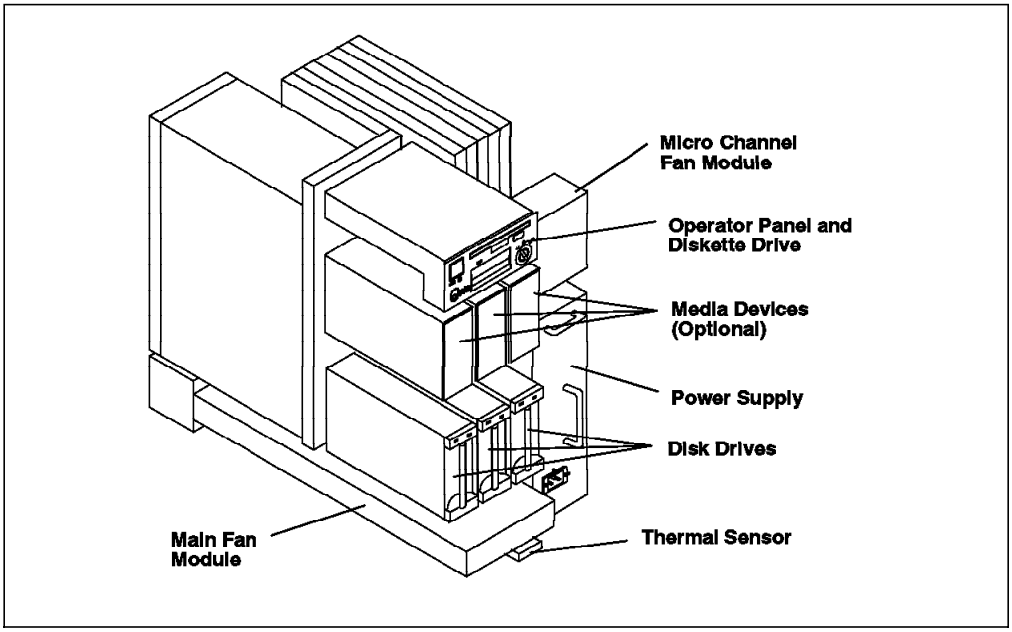


Figure 49. J40 Front Internal View

Figure 50 shows the rear of the J40. Four disks can be installed at the rear of the system. You can also see the SIB located at the back of the system.

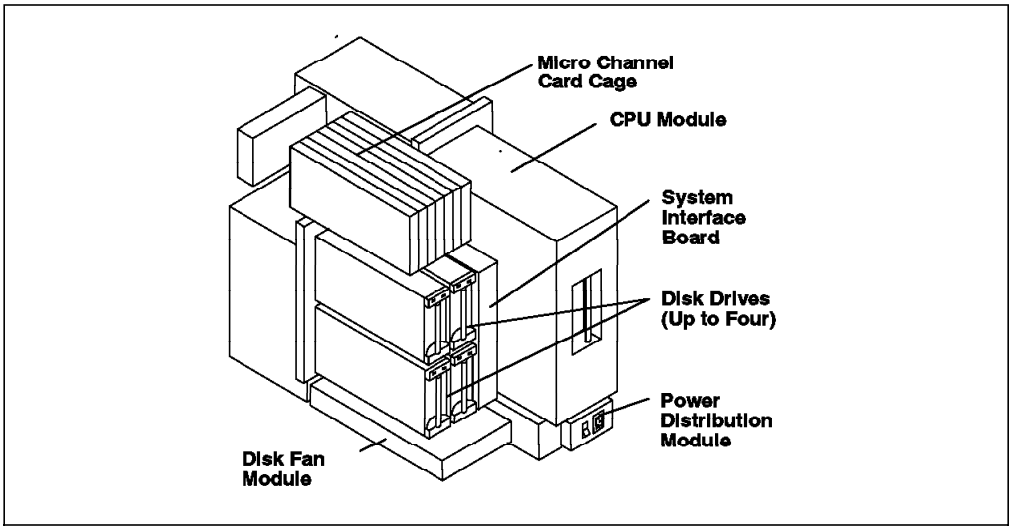


Figure 50. J40 Rear Internal View

4.4.7 Hot-Pluggable Disk Configuration Considerations

Attention!

Prior to removing a disk, you must make sure that appropriate software procedures have been carried out so that when the disk is removed, the ODM database does not get corrupted.

The two LEDs located on the disk must be off before removing the disk.

The software procedure to remove a disk depends on the logical volume layout of the system. All the procedures can be carried out through the SMIT System Storage Management menu. From this menu, you can select the Removable Disk Management option. You must go through this option before removing a disk.

```
                System Storage Management (Physical & Logical Storage)

Move cursor to desired item and press Enter.

  Logical Volume Manager
  File Systems
  Files & Directories
  Removable Disk Management
  System Backup Manager

F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

If the disk you want to remove belongs to the rootvg volume group, you must do the following:

- Go to the SMIT Removable Disk Management menu.
- Unmount all the file systems located on the disk you want to remove by using the Unmount File Systems on a Disk menu or by using the umount command.
- Move the contents of the disk to another disk within the same volume group by using the Move Contents of a Physical Volume menu. This assumes, of course, that you have enough space on your system to move the whole contents of your disk to other disks. The Move Contents of a Physical Volume option can be accessed as follows:

  ```
  System Storage Management (Physical & Logical Storage)
  --> Logical Volume Manager
      --> Physical Volumes
          --> Move Contents of a Physical Volume
  ```

- Once data has been moved to another disk, you can go back to the Removable Disk Management menu and proceed through the following path:

  ```
  --> Removable Disk Management
      --> Remove a Disk from the Operating System
          --> Remove a Disk without Data
  ```

  You will get the following screen:

```
                          Remove a Disk without Data

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  DISK Name                                            hdisk0
  VOLUME GROUP Name                                    rootvg
  FORCE deallocation of all partitions  on             no            +
    this physical volume?
  CABINET Number                                       0
  BUS ID                                               A
  CONNECTION Address                                   3,0
  KEEP definition in database?                         no            +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

At this step, you can keep the definition of the disk in the ODM database. Also you do not have to force deallocation of all the partitions on this physical volume since there is no longer any data on the disk.

Note: If you did not move data to another disk, forcing deallocation of all partitions on this physical volume will erase all data on the disk.

- Wait until the two LEDs on the disk are off. Then you can remove the disk safely. You might have to remove the front cover first before you can remove the hard disk.
- For a rootvg volume group hard disk migration, you will have to issue a bosboot command to get AIX to point to the new hard disk that you have just migrated your data to. For example, if hdisk1 is the new hard disk, and hdisk0 is the old hard disk, then the command will be:

  ```
  # bosboot -a -d /dev/hdisk1
  ```

If the disk you want to remove belongs to a non-rootvg volume group, the procedure is slightly different.

- Identify which disks belong to that non-rootvg volume group. Let's assume that we have three disks on the system and that hdisk0 belongs to rootvg and hdisk1 and hdisk2 belong to a non-rootvg volume group called nonrootvg.
- Unmount all the file systems located on hdisk1 and hdisk2.
- Deactivate the nonrootvg volume group using the following path:

  ```
  --> System Storage Management (Physical & Logical Storage)
      --> Logical Volume Manager
          --> Volume Groups
              --> Deactivate a Volume Group
  ```

- Once the volume group is deactivated, you can proceed by exporting the volume group by using the Export a Volume Group option.
- Then you can go back to the Removable Disk Management menu and remove each disk using the Remove a Disk with Data option. We suggest that you keep the definition of the disks in the ODM database.
- When all the LEDs on the disks are off, you can remove the disks.

If you want to reconfigure the disks belonging to the nonrootvg volume group, you need to configure them using the Configure a Defined Disk option. This option can be accessed as follows:

```
--> System Management
    --> Devices
        --> Fixed Disks
            --> Configure a Defined Disk
```

Once the disks are configured, you can import the nonrootvg volume group.

Note

It is easier and quicker to remove a disk from a non-rootvg volume group when there is only one disk within the non-rootvg volume group. In this case, you just need to go the Removable Disk Management menu, unmount the file systems located on this disk, and use the Remove a Disk with Data option. This will deactivate the volume group, export it and remove the disk from the system.
We then recommend, if suitable, to create a volume group for each disk you want to remove often.

4.4.8 J40 SCSI Device Addresses

As in the G40, SCSI device addresses in the J40 are fixed. Both media and disk drives do not have any jumpers or dials which allow you to change the SCSI addresses of the drives. The connectors to the base unit backplane determine the addresses of the SCSI devices.

All the devices located in the front of the system are connected to the first SCSI-2 Differential F/W bus, referred to as A on the backplane. The rear devices (if any) are connected to a second SCSI-2 Differential F/W bus, referred to as B on the backplane.

This means that if there are devices at the front as well as at the rear, a second SCSI-2 DE F/W (feature code 2416) and a SCSI internal cable (feature code 2441) are required.

The following is an example of a disk address:

```
hdisk0 Available 00-07-A1-3,0   2.2 GB SCSI Disk Drive
```

In this example, 3 is the SCSI address and 0 after the comma is the Logical Unit Number (LUN). The 7 is the slot number on the microchannel bus, and A is the SCSI bus A on the backplane.

Figure 51 on page 89 shows that devices on the front are connected to the first SCSI-2 Differential F/W controller (A), and the disks on the back are connected to the second SCSI-2 Differential F/W controller (B).

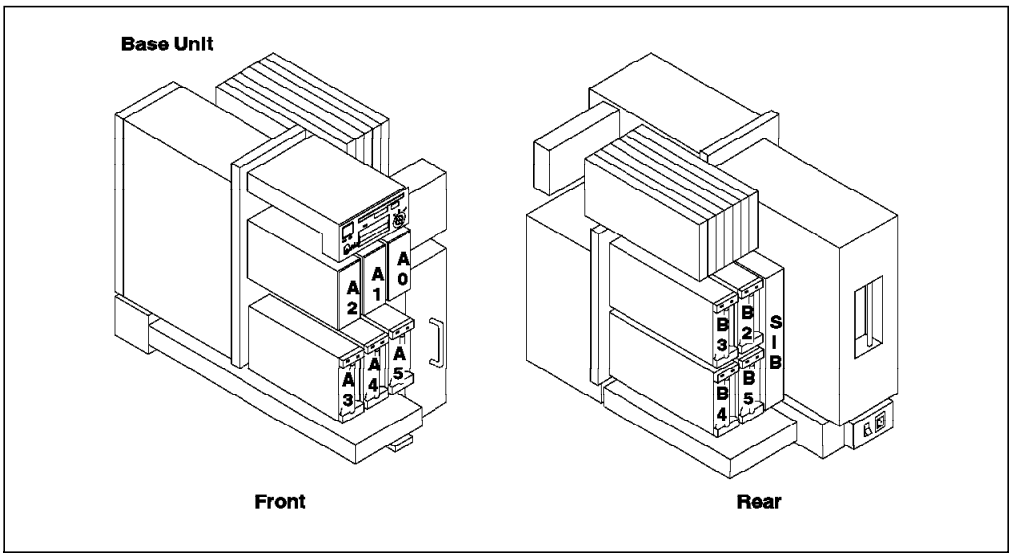


Figure 51. Front and Rear SCSI Devices Locations on the J40 Server

4.5 Model J01 Expansion Cabinet

The model J01 is an expansion unit for the J40 that expands the number of microchannel slots and disk drives. The J01 provides eight additional microchannel slots, which allows the J40 to reach a total of 15 microchannel slots. Thus, more microchannel adapters and more devices can be supported on the J40.

Figure 52 is a front view of the J01. You can see in this picture that the J01 has three disk bays and two media/disk bays on the front.

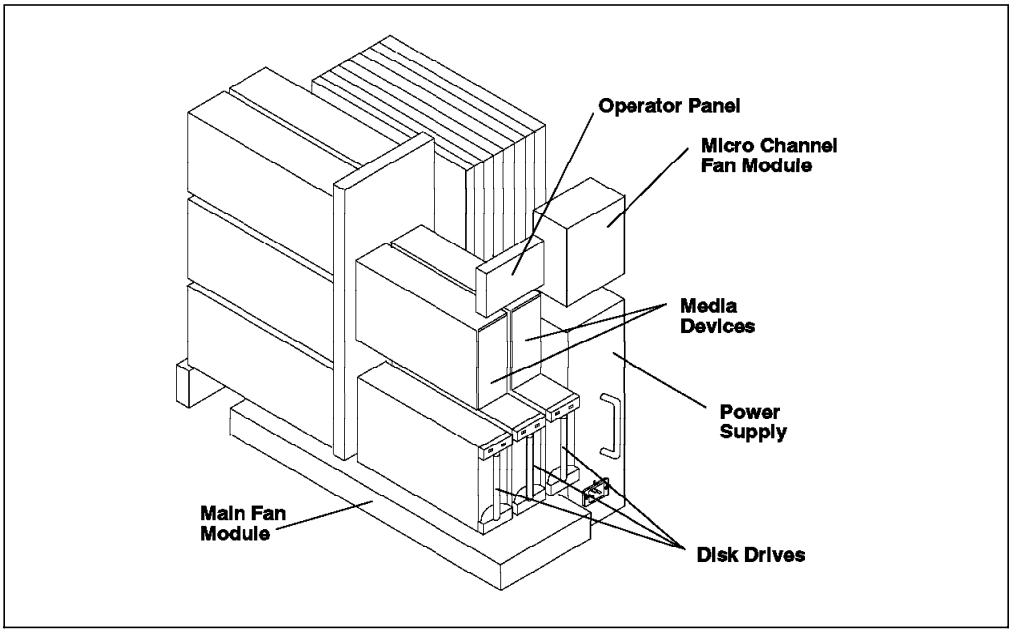


Figure 52. Front Internal View of the J01 Expansion Cabinet

Figure 53 on page 90 is the rear view of the J01. It shows that the J01 has nine disk bays at the back of the unit.

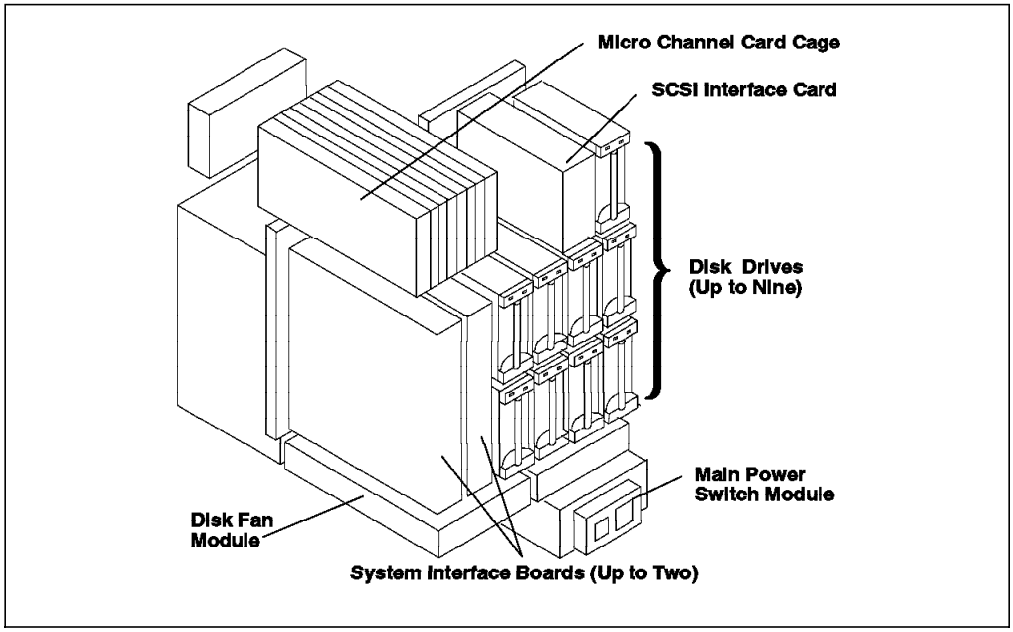


Figure 53. Rear Internal View of the J01 Expansion Cabinet

In summary, the J01 expansion unit provides:

- Eight additional microchannel adapters
- Up to 14 disks (if the two media bays on the front are used for disks)

In the expansion unit, J01, if all 14 disk drives are used, you need:

- Two SCSI-2 F/W DE adapter, (feature code 2416)
- Two SCSI cables for internal devices (feature code 2441)
- Two media-to-disk bay conversion kits (feature code 6511) in order to use the two media bays for disks

Note: Connecting external SCSI devices will require a separate SCSI adapter.

The J01 can have up to two SIBs. In fact, the SIB contains some hardware that allows SystemGuard to control the power of the unit and its devices. One SIB can control the power of up to 10 devices. Thus, when more than 10 devices are installed in the expansion unit, a second SIB is required.

As in the J40 base unit, the J01 has the same High Removability feature. All devices are hot-pluggable, but the lower three disks on the front have the same High Removability feature as seen in the base unit. These disks can be removed frequently.

Attention!

The system administrator must use the SMIT Removable Disk Management menus before removing a disk. Also, the LEDs must be off before removing a disk. Please refer to 4.4.7, “Hot-Pluggable Disk Configuration Considerations” on page 85 in this chapter.

As in the J40, the devices are connected to connectors located on the backplane. These connectors must be driven by two SCSI-2 Differential F/W adapters.

A dedicated bulkhead, called BHS, allows a connection between the SCSI-2 Differential F/W controller located in one microchannel slot and the backplane where the media or disk drives connectors are located. There is one BHS per expansion cabinet.

The J01 has the same dimension as the J40, that is:

- 610 mm (H)
- 360 mm (W)
- 750 mm (D)

4.5.1 J40 and J01 Interconnection

Figure 54 shows a J40 base unit with a J01 expansion unit connected to it. Note that the expansion unit must be connected on the left side of the base unit.

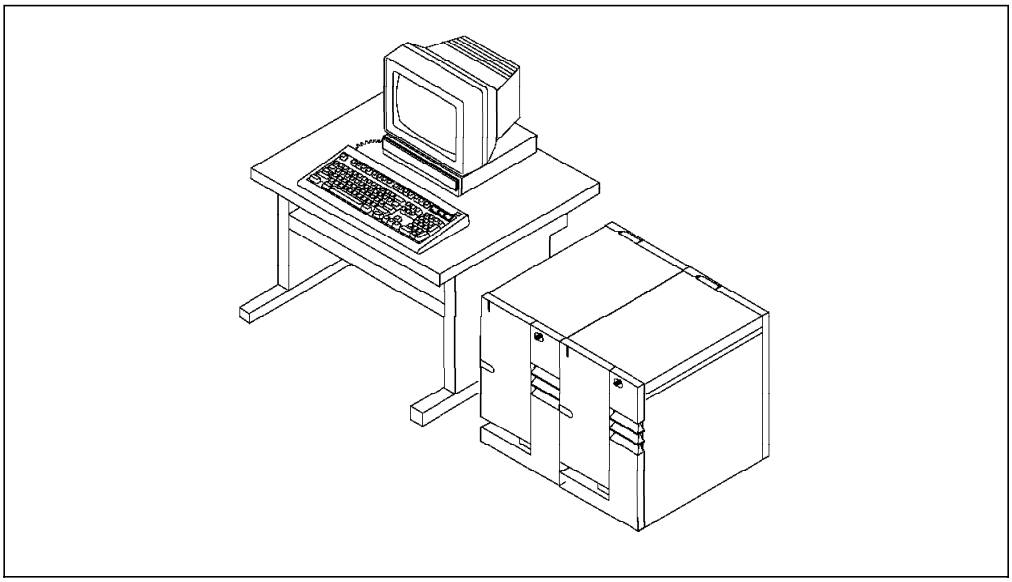


Figure 54. J40 and J01 Interconnection

To connect the J40 with the J01, follow this procedure:

- Connect the RS-485 cable between the J40 and the J01 (OUT port of J40 to IN port of J01).
- Connect the terminators to the IN connector on the J40 and to the OUT connector on the J01.
- Connect the FXE cable between the base and the expansion unit.
- For each SCSI bus in the expansion unit backplane, a dedicated controller (feature code 2416) must be installed. This dedicated SCSI controller is only used to drive internal disks. It cannot be used for external SCSI devices.
- On the expansion unit, the connection between the SCSI controller and the SCSI bus on the backplane is made through the SCSI interface card (BHS module) using feature code 2441. This bulkhead has two labelled connectors: A and B.
- Terminate, on the SIB board, the differential SCSI A and B connectors in both the base and the expansion units.

Note

The SCSI differential connectors located on the SIB board of both the base unit and the expansion unit cannot be used for connecting external devices. They have to be terminated.

The serial and parallel ports on the first SIB can be used to attach a printer or a terminal.The second SIB board (if required) on the J01 expansion unit is used only for power management.

Please refer to the *7013 J Series Service Guide* for a detailed description, and see the *7013 J Series Operator Guide*, chapter 9, for detailed installation instructions.

Figure 55 shows how to connect the expansion unit to the base unit.

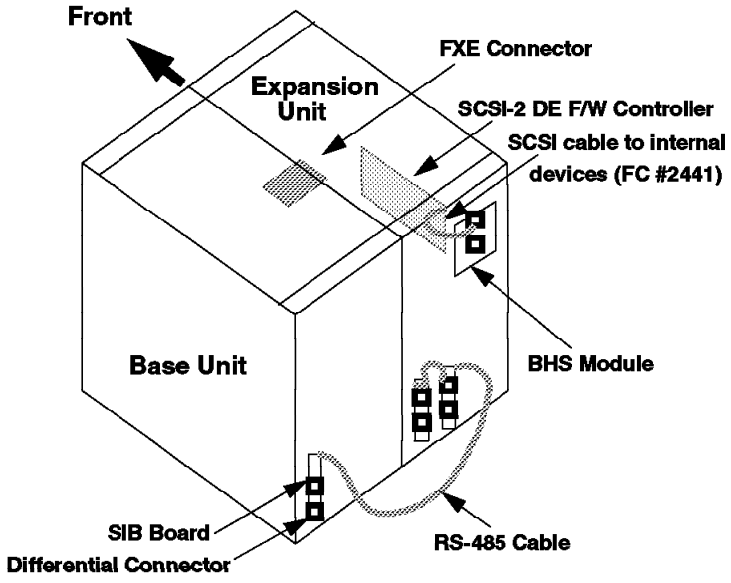


Figure 55. J40 and J01 Interconnection

4.5.2 Model J40/J01 Specifics

This section provides some useful additional information concerning the installation and administration of the J40.

To be able to boot a J40 system, the top cover must be closed. The J40 cannot boot if the top cover is not in place. In fact, the top cover, when installed, pushes an interlock switch that allows you to power-on the system.

The internal SCSI bus is a SCSI-2 Differential bus. The advantages of having this are the following:

- Longer cable
- Higher noise immunity
- Tolerance to the bus loading changes due to insertion and removal of hot-pluggable devices during bus activity (living bus)

On the J01 expansion cabinet, the disks have fixed addresses (as in the J40). The physical disk location determines the parent adapter and thus determines its SCSI address.

Figure 56 shows on which SCSI bus (A or B) the devices are connected according to their physical location. The way it works is different from the base unit. In the base unit, all devices in the front are connected to SCSI A, and all devices in the back are connected to SCSI B. On the J01, some devices in the front are connected to the SCSI A, and some are connected to SCSI B.

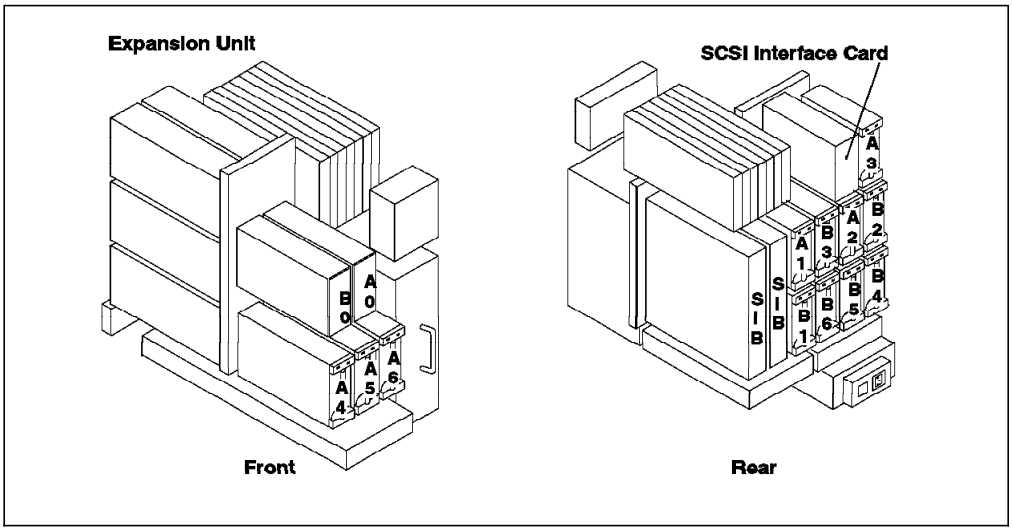


Figure 56. J01 SCSI Device Location and Addresses

4.6 Model R40 Rack Server

The IBM RS/6000 Model R40 is a drawer unit that can be mounted in an industry-standard rack.

Like the J40, the R40 comes with four slots for processor cards and four slots for memory cards.

However, the R40 comes standard with a total of 16 microchannel slots, 15 of which are available for other adapters, making it an ideal system for customers that need to attach a large number of peripherals.

The drawer itself has one disk bay and two media bays. Additional disk drawers can be purchased separately and inserted into the same rack.

The R40 is similar in design to the J40. The strong focus of the R40 is on high availability. The R40 provides high-availability features, such as redundant power supply and redundant fans. For increased availability, the rack can be equipped with an optional redundant power supply or an uninterruptable power supply (UPS). The optional redundant power supply will enable the R40 to continue operation even when one of the power supplies in the R40 stops functioning. A powerfail alert message will then be broadcast to the console. The failed power supply can then be replaced at a convenient scheduled time.

The R40 drawer uses the same rack, R00, as the R10, the R20 or the R24. This allows four R40 drawers to fit into the same rack, which is very suitable for an HACMP cluster.

Existing RS/6000 models, such as the R10, R20, R21 and R24, can be upgraded to the SMP server R40. For existing IBM RS/6000 models in the 9XX group, it can be upgraded to the SMP server R4U.

Figure 57 shows the R40 drawer installed in a rack.

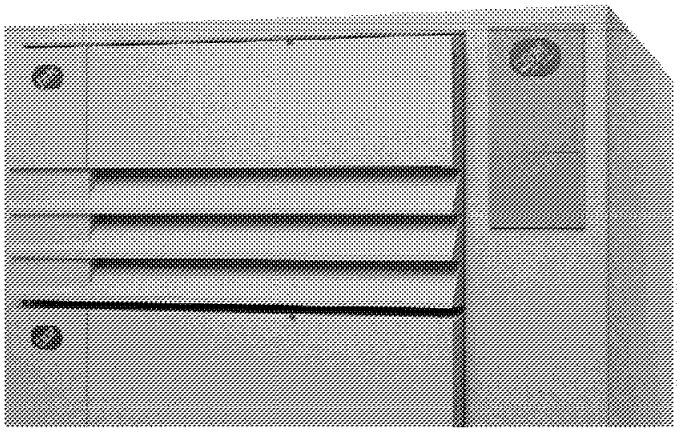


Figure 57. IBM RS/6000 Model R40 SMP Rack Server

4.6.1 Standard Configuration

The standard configuration for the R40 is as follows:

Microprocessor	: Dual 112 MHz PowerPC 604 Processor
Level 1 (L1) Cache	: 16 KB data / 16 KB instructions
Level 2 (L2) Cache	: 1 MB
RAM (memory)	: 128 MB
Memory bus width	: 256 bits
Memory slots	: Four
Diskette drive	: 1.44 MB 3.5 inch diskette drive
Internal disk drive	: 2.2 GB SCSI-2 Fast/Wide
CD-ROM drive	: Quad-speed
Disk/media bays	: Three (one available)
Expansion slots	: 16 160 MB/second Micro Channel (15 available)
Ports	: One parallel, three serial, two RS-485, one battery backup unit connector
Service Processor	: IBM SystemGuard
Power Supply	: Dual 220 VAC or -48 VDC (redundant and hot swappable)
System Dimensions	: 10.5” H x 17.52” W x 36.42” D (266.7 mm x 445 mm x 925 mm)
System Weight	: 130 lbs (59 Kg)

4.6.2 System Expansion

The R40 expansion capabilities are as follows:

RAM : Up to 2 GB

Microprocessor : Up to eight 112 MHz PowerPC 604 processors

Internal disk storage : 2.2 GB SCSI-2

Total disk storage : More than 4.5 TB with R00 System Rack expansion units

4.6.3 Select and Optional Features

The following is a representative list of available features and products that are supported in the R40.

Memory

- Expandable to 2G using 128 MB, 256 MB and 512 MB memory cards

Internal Storage

- CD-ROM Drives
- Disk drives
  - 2.2 GB SCSI-2

Tape drives

- 5/10 GB, 8mm
- 4/8 GB, 4mm
- 1.2 GB, 1/4 inch cartridge

SCSI interfaces

- SCSI-2
- SCSI-2 Differential
- SCSI-2 Fast/Wide
- SCSI-2 Fast/Wide Differential

Adapters and controllers

- 128-Port Asynchronous Controller
- 4-Port Multiprotocol Communications Controller
- 8- and 16-Port Asynchronous Adapters
- ARTIC960 Co-Processor Adapter
- Block Multiplexer Channel Adapter
- Digital Trunk Adapter (single and dual)
- Ethernet High-Performance LAN Adapter
- FDDI Fiber Adapter (single and dual ring)
- High-Performance Subsystem Adapter
- IBM Auto Token-Ring LANstreamer 32 MC Adapter
- IBM Ultimedia Audio Adapter
- Realtime Interface Co-Processor Portmaster Adapter

- S/390 ESCON Channel Emulator
- TURBOWAYS 155 ATM Adapter
- X.25 Interface Co-Processor/2 Adapter
- System/370 Host Interface Adapter
- High-Performance 4-port SSA Adapter

4.6.4 Supported Devices

The following list of devices is only a representative sample. For a complete and detailed list of supported devices, contact your IBM Marketing Representative or your IBM Business Partner.

External Storage Devices

- Rewritable optical drives
- CD-ROM drives
- Disk drives
  - 1 GB, 1.1 GB, 2 GB, 2.2 GB, 4.5 GB SCSI-2
- Tape drives
  - 5 GB, 8 mm
  - 4 GB, 4 mm
  - 1.2 GB, 1/4 inch cartridge
  - 800 MB, 1/2 inch cartridge
  - 10 GB, 1/2 inch cartridge
  - 1/2 inch reel

External Storage Systems

- Disk array subsystems
- High-performance disk drive subsystems
- High-density SCSI-2 disk subsystem
- Model G02 Expansion Cabinet
- Optical libraries
- SCSI-2 disk expansion units
- Tape subsystems and libraries

Other Devices

- Digital Trunk Processors
- Ethernet 10Base-T and 10Base-2 transceivers
- Telephone company central office features

Terminals

- A wide range of ASCII terminals, as well as X11 terminals

Printers and modems

- A wide range of IBM and non-IBM printers, plotters and modems

4.6.5 Additional Information on the R40 Server

This section provides some additional information on the R40 that might be useful when configuring, installing or using it:

- Redundant fans:

  There are a number of fans in the R40: three media fans, three CPU fans and four I/O module fans. If any one of these fans were to fail, the system would continue to run. An error entry may be posted in the AIX error log with a label of EPOW_SUS. This error entry can then be decoded by the Customer Engineer to locate the exact fan that fails.

- Dual Power Supply Option:

  This provides an option for a second AC power supply, or for a DC power supply, for power redundancy. If any one of these power supplies were to fail, the system would continue to run. An error log entry may be posted with a label of EPOW_SUS. This error log entry can then be decoded by the Customer Engineer to locate the exact power supply that fails.

- Uninterruptable power supply (UPS):

  This UPS is called a Powerware Prestige 3000 Rackmount UPS for RS/6000. It has a machine type/model of 9910-U33. This UPS will protect hardware and software by detecting electrical anomalies and providing power for a limited period of time. The UPS has a fully rated battery hold-up time of seven minutes. It supports a maximum load of 3000 VA (2100 W).

  The UPS is used in conjuction with a software called OnliNet Multi-Host Support (MHS). This software allows control of up to four processor drawers.

  The OnliNet software runs in the background as an application program initiated during AIX system load. When the primary power is lost, the UPS communicates with the OnliNet software. After determining the nature of the failure (bad battery, over temperature, defective inverter or loss of the primary power), the software can issue a shutdown to preserve data integrity. The delay between the loss of the primary power and the shutdown can easily be customized.

  The maximum number of U33 UPSs that may be installed in a single 7015 rack is two. When a second U33 UPS is ordered for a single rack, an additional PDU must be ordered (feature code 6171 or 6173).

  AIX V4.2 adds a new option to the shutdown command: `shutdown -p`. This option prevents control from being handed back to SystemGuard once the machine has shut down. This is neccesary when using intelligent UPS monitoring as a normal shutdown would require operator intervention on the BUMP console to restart the server. With the `shutdown -p` command, as soon as the main power is returned, the server will start to boot. We recommend that you use the `shutdown -p` command to perform any automated shutdowns, as part of an intelligent UPS monitoring system.

- Cluster Power Controller (CPC):

  A CPC option (feature code 6175) is available on the R00 rack. The CPC allows the user to connect several CPUs or disk drawers to a single ASCII terminal as a console for all the CPUs. From this console, you can power-off or power-on each of the CPUs or disk units.

  The console can be local or remote, through the use of a modem. Up to two CPCs can be configured in a R00 rack. For more information on the CPC, refer to Chapter 7, “Cluster Power Controller” on page 175 in this redbook.

- Layout:

  The layout of the R40 is similar to a R24 rack-mounted uniprocessor.

  The operator panel display is located behind the front bezel door. To access the operator panel, you need to rotate the top of the front bezel door downward. The functions of this operator panel display are similar to those of the J40.

  The R40 CPU enclosure is a rack-mounted drawer containing a CPU module, I/O module, media module and a power supply with a possible extra, optional power supply.

  The rear of the R40 provides 16 I/O slots (one is taken up by the standard SCSI-2 F/W adapter) and the SIB board. The SIB is similar to the J40.

  One of the power supplies can be replaced with a cooling unit or a DC supply input.

  There are two microchannel buses, providing 16 slots.

Figure 58 shows an internal view of the front of the R40 drawer.

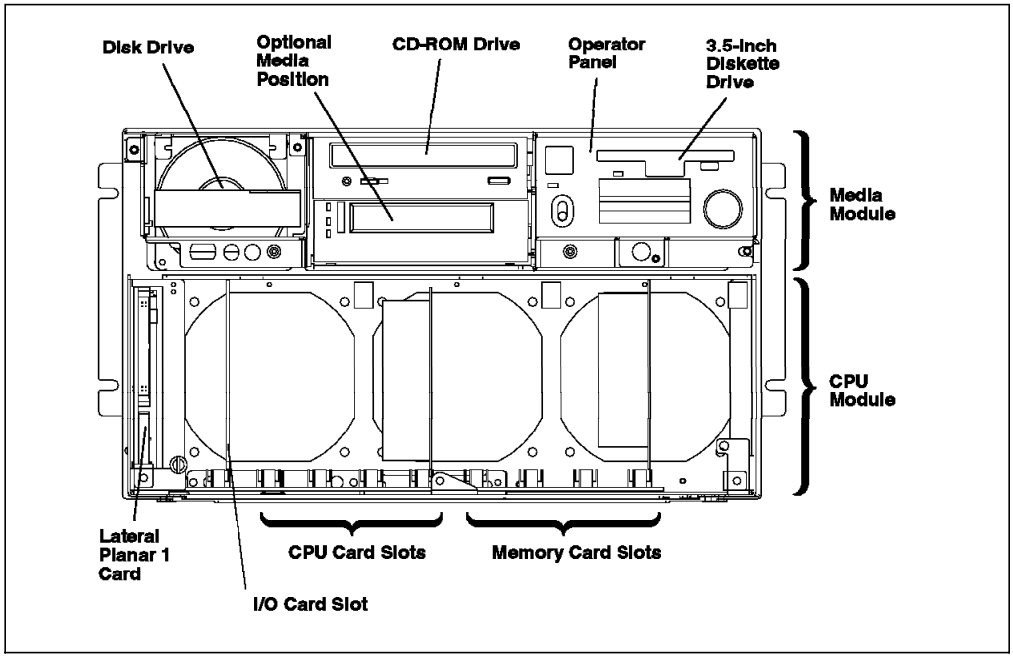


Figure 58. Front Internal View of the R40 Server

Figure 59 on page 99 shows the rear of the R40 drawer with its SIB (on the right).

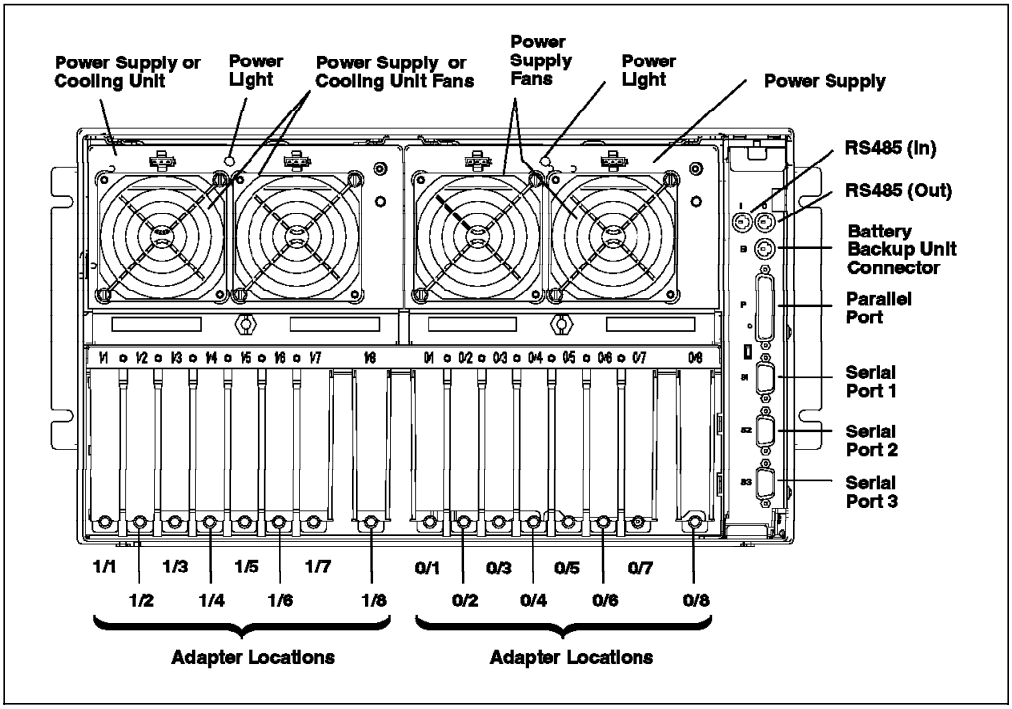


Figure 59. Rear View of the R40 Server

4.7 Using a UPS with the G30, G40, J30, J40, R30 and R40

The G30, G40, J30, J40, R30 and R40 have a unique built-in power supply feature. If you shut down any of the above models running AIX V4.1.4, AIX will give control to the service processor, SystemGuard. Even if the power switch is set to on, rebooting them will still require a manual intervention on the BUMP console, which is the ASCII console directly or remotely connected to S1. If you power-off and then power-on the UPS, the system will not reboot automatically. The BUMP console will display a > prompt. If the > prompt does not show, you will have to press the Enter key on the console to display it. At this prompt you will need to enter the power on string, which is `power` by default.

If you are running AIX V4.2, the shutdown command has a `-p` option. This option allows you to shut down AIX without giving control to SystemGuard. If the power goes out and comes back, the system should reboot without manual intervention at the BUMP console. However, the system will boot up in the normal boot process.

UPSs that have a hardware intelligent connection and monitoring software to the SMP Servers, such as OnliNet Multi-Host Support (MHS), should include this `shutdown -p` option into the shutdown script files.

4.8 Upgrading G30, J30 and R30/R3U from 601 to 604 Processors

There is a no-charge upgrade to the PowerPC 604 processors for the RS/6000 PowerPC 601 SMP Models G30, J30, R30 and R3U. Each dual PowerPC 601 processor card currently installed may be replaced with a dual PowerPC 604 processor card. The dual PowerPC 604 112 MHz processor cards come standard

with 0.5 MB L2 cache per processor on the Model G30 and with 1 MB L2 cache per processor on the Models J30, R30 and R3U.

To install the Dual PowerPC 604 112Mhz Processor Card upgrades, you will have to install at least AIX V4.1.4 with APAR IX57164 or install AIX V4.2 in the system prior to the physical hardware upgrades.

Attention!

The information below is aimed at planning and understanding 601 to 604 upgrades for SMPs. It is not a step by step procedure and information might change in the future. When performing such an upgrade the procedures shipped with the hardware upgrade should be followed carefully. Also all necessary ECs, firmware, diskettes and parts will be automatically shipped together with the ordered upgrade.

4.8.1 Upgrading a G30 601 to 604 Processors

Some early shipped machines required EC D73676 to be installed before the upgrades can be performed. EC D73676 updates the system firmware and provides mechanical improvements.

Following is the summary for a G30 upgrade from a 601 to 604:

- Ensure that the System Planar (ELM) is at the right level for the upgrade.
- Determine if the firmware is at the proper level.
- Remove any media from the tape drive, CD-ROM drive or diskette drive and insert the Vital Product Data (VPD) diskette that is provided with the upgrade into the diskette drive.
- Boot the system up in Service mode and select y to start the VPD install program when prompted at the system console. The VPD install program will place the system in “Stand-by” mode.
- Disconnect power. Remove and replace the two PROMS on the System Planar at location U5 and U143.
- Replace a 2-way 601 SMP CPU card with a 2-way 604 SMP CPU card in the same location as before. There may be more than one CPU card per system.
- Insert the Firmware Diskette as provided in this upgrade to the diskette drive.
- Boot the system in Service mode. The Flash Firmware will be updated and the system will reboot and present the Maintenance menu.
- Remove the firmware diskette from the diskette drive and select **System Boot**, then **Boot From List** and continue.
- Go to **Display Vital Product Data** and refer to the `ioplanar0` section to determine that the Flash EPROM is at the updated level. The flash firmware level has to be at 09.22 or higher. Otherwise, the firmware did not install correctly, and you will have to repeat the firmware update.
- Go to **Function Selection** and select **Advance Diagnostics Routines, System Verification** and **System Checkout**.
- Upon successful completion of the above diagnostics, you can shut down the system and reboot in the Normal mode.

Note

Mixing of 601 and 604 CPU card(s) is not supported.

A 601 CPU Card can be identified by looking at the top edge of the card for “C1D” or “E1D”.

A 604 CPU Card can be identified by looking at the top edge of the card for “C4D”.

4.8.2 Upgrading a J30 601 to 604 Processors

Some early shipped machines required EC D29704. EC D29704 upgrades the system logic to level 7.01 firmware, and provides the required chassis changes. This EC is a pre-requisite for several SMP features that includes:

- NFx Memory -- feature code 4124, 4125, 4126, 4127, 4156, 4157 and 4158
- X.25 Adapter -- feature code 2960
- 6/8 Way System Multiprocessor -- feature code 4302

The above new features should not be installed until after the EC has been installed.

Following is the summary for the J30 upgrade from 601 to 604 CPU:

- Ensure that the Power Supply Interface (SIB) and the I/O Planar (IOD) is at the right level for the upgrade.
- Determine if the firmware is at the proper level.
- Remove any media from the tape drive, CD-ROM drive or diskette drive and insert the Vital Product Data (VPD) diskette that is provided with the upgrade into the diskette drive.
- Boot the system in Service mode and select **y** to start the VPD install program when prompted at the system console. The VPD install program will place the system in “Stand-by” mode.
- Disconnect power. Remove and replace the two PROMS on the System Planar at location U1 and U50.
- Replace a two-way 601 SMP CPU card with a two-way 604 SMP CPU card in the same location as before. There may be more than one CPU card per system.
- Insert the firmware diskette as provided in the upgrade into the diskette drive.
- Boot the system in Service mode. The flash firmware will be updated and the system will reboot and present you with the Maintenance menu.
- Remove the firmware diskette from the diskette drive and select **System Boot**, then **Boot From List** and continue.
- Go to **Display Vital Product Data** and refer to the `ioplanar0` section to determine that the Flash EPROM is at the updated level. The flash firmware level has to be at 09.22 or higher. Otherwise, the firmware did not install correctly, and you will have to repeat the firmware update.
- Go to **Function Selection** and select **Advance Diagnostics Routines**, **System Verification** and **System Checkout**.

Upon successfully completion of the above diagnostics, you can shut down the system and reboot in the Normal mode.

Note

Mixing of 601 and 604 CPU card(s) is not supported.

A 601 CPU Card can be identified by looking at the top edge of the card for “C1D” or “E1D”.

A 604 CPU Card can be identified by looking at the top edge of the card for “C4D”.

4.8.3 Upgrading a R30/R3U 601 to 604 Processors

Some early shipped machines required EC D28567A to be installed prior to the upgrade. EC D28567A will provide a new CPU fan assembly needed for the 604 CPUs.

Following is the summary for the R30/R3U upgrade from 601 to 604:

- Ensure that the Power Supply Interface (SIB) and the I/O Planar (IOD) is at the right level for the upgrade.
- Determine if the firmware is at the proper level.
- Remove any media from the tape drive, CD-ROM drive or diskette drive and insert the Vital Product Data (VPD) diskette that is provided in this upgrade into the diskette drive.
- Boot the system in Service mode and select **y** to start the VPD install program when prompted at the system console. The VPD install program will place the system in “Stand-by” mode.
- Disconnect power. Remove and replace the two PROMS on the System Planar at location U1 and U50.
- Replace a two-way 601 SMP CPU card with a two-way 604 SMP CPU card in the same location as before. There may be more than one CPU card per system.
- Insert the firmware diskette as provided in this upgrade into the diskette drive.
- Boot the system in Service mode. The flash firmware will be updated and the system will reboot and present the Maintenance menu.
- Remove the firmware diskette from the diskette drive and select **System Boot**, then **Boot From List** and continue.
- Go to **Display Vital Product Data** and refer to the `ioplanar0` section to determine that the Flash EPROM is at the updated level. The flash firmware level have to be at 09.22 or higher. Otherwise, the firmware did not install correctly, and you will have to repeat the firmware update.
- Go to **Function Selection** and select **Advance Diagnostics Routines**, **System Verification** and **System Checkout**.
- Upon successfully completion of the above diagnostics, you can shut down the system and reboot in the Normal mode.

Note

Mixing of 601 and 604 CPU card(s) is not supported.

A 601 CPU Card can be identified by looking at the top edge of the card for “C1D” or “E1D”.

A 604 CPU Card can be identified by looking at the top edge of the card for “C4D”.

4.9 UP to 604 SMP Upgrade Paths

Model conversion from various existing uniprocessor systems to the three SMP servers described in this book are offered. The upgrade path for all the models requires chassis replacement while retaining the current serial number. In some cases, I/O adapters, memory SIMMs and disk drives can be reused.

4.9.1 UP Upgrades to G40

These are the available upgrade paths to the SMP G40:

- Models 340, 34H, 350, 360 and 370 are converted to a two-way G40
- Models 380, 390 and 39H are converted to a four-way G40

New memory cards must be used for the G40. You should be aware of the following:

- Only one memory slot is available on the G40.
- Memory exchange options are available. Since old memory cards are not compatible with the G40, IBM exchanges currently installed memory cards for an equal or greater capacity of memory card compatible for the G40.
- Old SIMMs from 128 MB and 256 MB memory cards can be reused. If the machine to be upgraded has two 128 MB or two 256 MB memory cards installed, then the SIMMs can be reused to create a single 256 MB card or a single 512 MB card. This requires feature code 4061 and 4062, respectively.

The chassis gets replaced, but the serial number is retained. Installation instructions are shipped with the upgrade.

4.9.2 UP Upgrades to J40

Model conversions from deskside models 5xx to the J40 are also offered. For all 5xx servers, except 58H, 590 and 59H models, the upgrade will come with one dual 604 112 MHz processor card, while for the 58H, 590 and 59H, the upgrade to the J40 provides a second dual 604 processor card.

The uniprocessor memory is not compatible with the existing J40 model; a minimum of 128 MB is required for the SMP model J40.

These are the available upgrade paths to the SMP J40:

- Models 520, 52H, 530, 53E, 53H, 540, 550, 55E, 55L, 55S, 560, 56F, 570, 57F, 580 and 58F are upgraded to a two-way J40.
- Models 58H, 590 and 59H are upgraded to a four-way J40.

The 3.5-inch disk drives can be reused, but they require the hot-pluggable disk enclosure, also referred to as disk carrier.

The chassis gets replaced, but the original serial number is retained.

4.9.3 UP Upgrades to R40

These are the available upgrade paths to the R40:

- Models 930, 950, 95E, 970, 97B, 97E, 97F, 980, 98B, 98E and 98F are converted to a two-way R4U.
- Model R10 is converted to a two-way R40.
- Models 990, 99E, 99F, 99J and 99K are converted to a four-way R4U.
- Models R20, R21 and R24 are converted to a four-way R40.
- Model R4U includes the same enclosure as the R40 but supports the 900 series type of model configurations.

New memory cards must be used in the R40. SIMMs from old 128 MB and 256 MB cards can be reused.

The serial number of the original machine is retained through the upgrade.

4.10 System Interface Board (SIB) Functions

Each SMP model has a SIB which is slightly different from the others. The SIB provides the following functions or features:

- A power microcontroller helps the service processor to monitor each unit and its power supply and also guarantees communication between units.
- A System Basic Lines Physical Interface checks the asynchronous line driver and receiver.
- Three serial ports.
- One parallel port.
- An RS-485 interface for connecting expansion units.

On the J40/J01, the SIB has some additional functions:

- The Removable Disk Switch enables the removal of one or more SCSI devices while the machine is still powered and operational.
- The SIB is also a bulkhead for the first (A) and second (B) SCSI buses. The signals come from the SCSI controller through the unit backplane and reach the dedicated connector on the SIB. The two SCSI-2 connectors are 68-pin SCSI-2 Differential Fast/Wide connectors. Both SCSI buses must be terminated with a SCSI Differential terminator at the rear of the J40.

Finally, in the G40, the SIB is part of the system planar, while in the J40 and R40, it is in the form of a pluggable module.

Chapter 5. SystemGuard

This chapter introduces the SystemGuard service processor which is included in the IBM RS/6000 SMP server models G40, J40 and R40.

5.1 Introduction

IBM′s Symmetric Multiprocessor (SMP) products are designed to be servers in commercial environments. Commercial servers run applications shared by many users. Availability of those applications is usually very important. Also, servers often operate in unattended or remote locations. In this case, providing remote diagnostics and service, while at the same time protecting access to sensitive data, is very important.

IBM′s family of enterprise SMP servers includes a service processor, called *SystemGuard*, as a standard feature.

SystemGuard continually monitors the hardware as well as the operating system. If, for instance, a CPU were to fail, the system would detect this, reboot itself automatically, and run without the failed CPU. Likewise, if there were a memory error that could not be corrected, the system would detect this, reboot itself automatically, and run without the failed memory component.

SystemGuard allows diagnostics and some maintenance to be performed either locally or remotely. This is especially useful to customers with distributed systems who may not have personnel with computer skills at their remote sites. The IBM SystemGuard processor makes it possible for these remote systems to be managed from a central location. The RS/6000 SMP servers can be set up to automatically call an IBM Service Center if they fail to boot successfully.

In addition, the SystemGuard processor operates on its own power boundary, making it possible to work on the system even if it’s powered off.

The main features of SystemGuard are:

- Initialization process flow management
- Remote as well as local control of the system (power-on/off, diagnostics, reconfiguration, and some maintenance)
- Console mirroring to make actions performed by a remote service technician visible to and controllable by the customer
- Dial-out to a support center in case of system boot failure
- Run-time hardware and operating system surveillance

5.2 SystemGuard Power

SystemGuard has its own power boundary. This means that even if the system power is off (the power indicator on the Operator Panel is off), SystemGuard may still be powered on. This allows work on the system even though it’s shut down. The only way to power-off SystemGuard is to turn off the switch on the back of the J and R models or disconnect the power cord on the G model. In order to power-on the entire system, you must first power-on SystemGuard by turning on the back switch or by plugging in the power cord.

5.3 SystemGuard Components

SystemGuard introduces new hardware and firmware components that you need to understand when installing, booting, or maintaining an SMP server.

New hardware components are:

- A bring-up microprocessor (BUMP) with its Electrically Programmable Read-Only Memory (EPROM) and its RAM
- A Flash EPROM
- A Backup EPROM

Part of the SystemGuard firmware is stored in the BUMP EPROM; part is in the Flash EPROM. The Backup EPROM contains a subset of the Flash EPROM that allows the system to boot in case of a Flash EPROM failure.

The BUMP has access through the Inter-Integrated Circuit (I2C) bus to existing components, such as the system EEPROM (Electrically Erasable Programmable Read-Only Memory), all the boards′ EPROMs, the power supply, and so on. The EEPROMs contain the Vital Product Data (VPD) of the machine. There is one EEPROM per board.

The BUMP also interfaces with the Operator Panel, the nonvolatile memory (NVRAM), the Flash EPROM, the Backup EPROM, and the S1 and S2 serial ports.

Physically, the BUMP and its associated components are located on the I/O board.

10760 shows the SystemGuard hardware components. The MPB board stands for Multiprocessor Board. The COP is the Common On-chip Processor that is used for testing the PowerPC 604 chip. The IONIANs are components that handle the MicroChannel (MCA) buses.

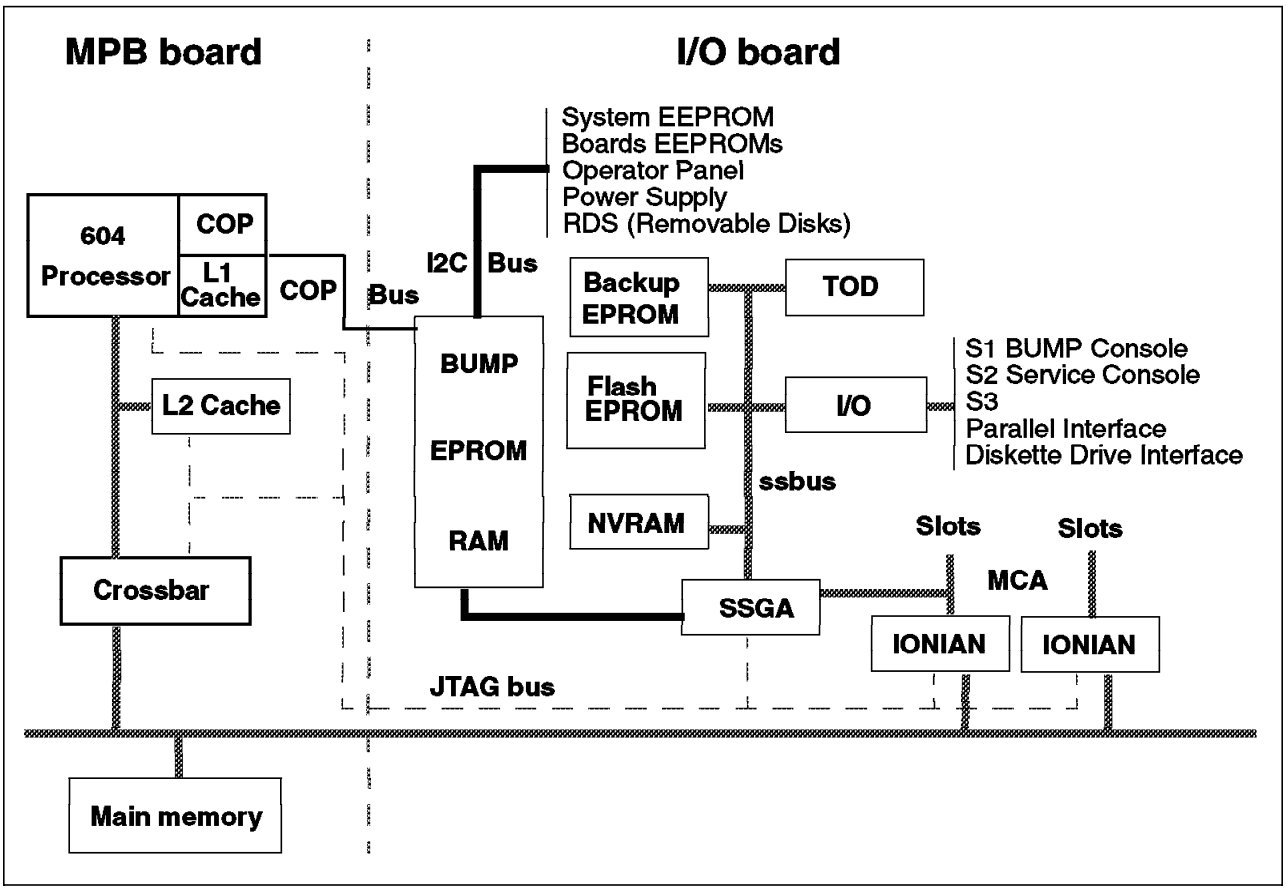


Figure 60. SystemGuard Hardware Components

.*

5.4 The Operator Panel

The Operator Panel is the first level of user interface to SystemGuard. It houses the Physical Key (Key Mode Switch), the power switch, the reset button, a green LED (power indicator), and a 2x16-character LCD (Liquid Crystal Display).

The Operator Panel card contains the NVRAM, the NVRAM battery, and the TOD (Time-Of-Day). Pulling out the Operator Panel will result in a reset of the TOD to 1969, which will affect applications protected by a license key and cause a loss of configuration data in the NVRAM.

The Operator Panel has the following features:

- Power button: It should generally stay on all the time if you want to power-on or power-off the system remotely.
- Reset button: It resets SystemGuard to the IPL phase and, depending on the key position, reboots the system.
- LCD display: It is made of two rows of sixteen characters. It displays the word Stand-By in the Stand-By phase, the usual three-digit LED codes while the system is booting or if an error occurs, and is blank during normal system operation.
- Physical Key: It uses the international symbols for Normal, Secure, and Service modes. This key should generally stay in the Normal position since

the modes can be changed electronically when the Physical Key is in Normal position.

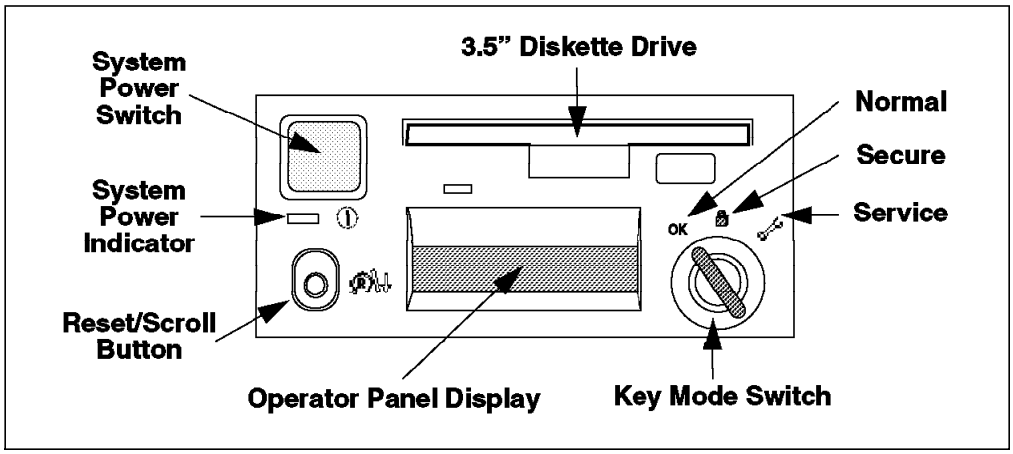


Figure 61. Operator Panel

5.5 SystemGuard Consoles

Apart from the Operator Panel, which can be viewed as a small console with a minimal user interface, SystemGuard works with two types of consoles:

- The *BUMP Console* is the ASCII terminal or terminal emulator connected to the S1 serial port. This console provides the normal input to the BUMP. It can be either local or remote. The line speed for the BUMP Console must be set to 9600 Baud.
- The *Service Console* is the ASCII terminal or emulator connected to the S2 serial port. This console is usually remote, located in a Customer Support Center or an IBM Service Support Center. Basically, this console allows remote support of SystemGuard and AIX. The support center needs authorization from the customer via SystemGuard flags to access the system.

It is now possible to attach a graphical display (Low Function Terminal or LFT) to a G or J series machine equipped with a POWER GXT150M graphics adapter. If the graphical display is used as the console, the Service Console and mirroring are not supported. The LFT will be used, in ASCII mode, as the BUMP Console.

5.6 SystemGuard Functions

SystemGuard controls the system when:

- The system power is off. In this state, SystemGuard allows the system administrator or service personnel to run tests, set SystemGuard parameters, reconfigure elements of the system, and power-on the system.
- The system is booting. SystemGuard controls the hardware Power-On (PON) tests and the loading of AIX.
- AIX is running through the surveillance function that implements a heartbeat protocol between AIX and SystemGuard.

SystemGuard can be accessed locally or remotely from the BUMP or from the Service Console. Access is possible through SystemGuard Menus when AIX is not loaded, or through AIX Diagnostics or the AIX command line. The Stand-By Menu and the Maintenance Menu are available before AIX is loaded.

5.7 Physical and Electronic Key

The system mode (Normal, Secure, Service) can be set by turning the Physical Key or by switching the Electronic Key. The Physical Key and the Electronic Key together define a state called the *System Key*. The Electronic Key can only be set if the Physical Key is in Normal position. Otherwise, it is disabled.

Following are various Electronic and Physical Key combinations and the resultant System Key position:

Table 3. Physical Key in Normal

Physical Key	Electronic Key	System Key
normal	normal	normal
normal	secure	secure
normal	service	service

Table 4. Physical Key In Secure

Physical Key	Electronic Key	System Key
secure	not valid	secure

Table 5. Physical Key in Service

Physical Key	Electronic Key	System Key
service	not valid	service

5.8 SystemGuard Phases

Booting an SMP server is slightly different from other IBM RS/6000s because of the control of SystemGuard. During bootup and normal operation, SMP servers go through three different phases: Stand-By, Init, and Run-Time.

5.8.1 Stand-By Phase

The system is in the Stand-By phase anytime the system power is off and the BUMP is powered on.

If the BUMP is not yet powered on, the Stand-By phase is entered by plugging the unit into an electrical outlet and by turning on the power switch on the back of the J and R models. The G models do not have a switch; plugging the cord into an outlet controls power.

In the Stand-By phase, the AIX operating system is not yet loaded; the system power is not on, and the word Stand-By is displayed on the Operator Panel.

The BUMP is active, and it can receive commands from the BUMP Console or Service Console. You can enter the SystemGuard Stand-By Menu from this phase.

The Stand-By phase ends when the power button on the Operator Panel is pressed (if toggled off) and the power command is entered, causing the system to begin to boot.

5.8.2 Init Phase

Turning on power to the system unit causes the SMP to enter the Init phase.

If the System Key is in Normal, BUMP runs the built-in or resident Power-ON Tests, IPLs on the first available processor, runs the functional POST (Power-On Self Tests) on the I/O subsystem, and finally loads the AIX operating system.

If the System Key is in Service, and if certain flags are set, the system enters the SystemGuard Maintenance Menu. The qualifying conditions are: the Autoservice IPL flag is disabled, the BUMP Console is present, and there is a valid service contract.

If the System Key is in Secure, the system enters the Stall state and the LCD displays the three-digit code 200. The initialization of the system stops until the Physical Key or the Electronic Key is set to Normal or Service. This ends the Stall state and control of the system is passed to AIX.

5.8.3 Run-Time Phase

In this phase, the AIX operating system is in control of the system. The Run-Time phase is entered once the AIX operating system starts loading and LED 299 appears. AIX now controls the serial ports and consoles.

When AIX is stopped, for example by the `shutdown` command, the system goes back to the Stand-By phase. If an Uninterruptable Power Supply is attached to the SMP via S3, and if the machine is running AIX V4.2 or later, running `shutdown` with the `-p` flag will enable the system to reboot automatically when power is restored. Otherwise, the power-on command string will have to be entered at the Stand-By prompt on the BUMP Console.

If the system crashes during normal operation, it will reboot automatically unless the value for `autorestart` is false. By default, the value is true

Figure 62 on page 111 shows the SystemGuard phases.

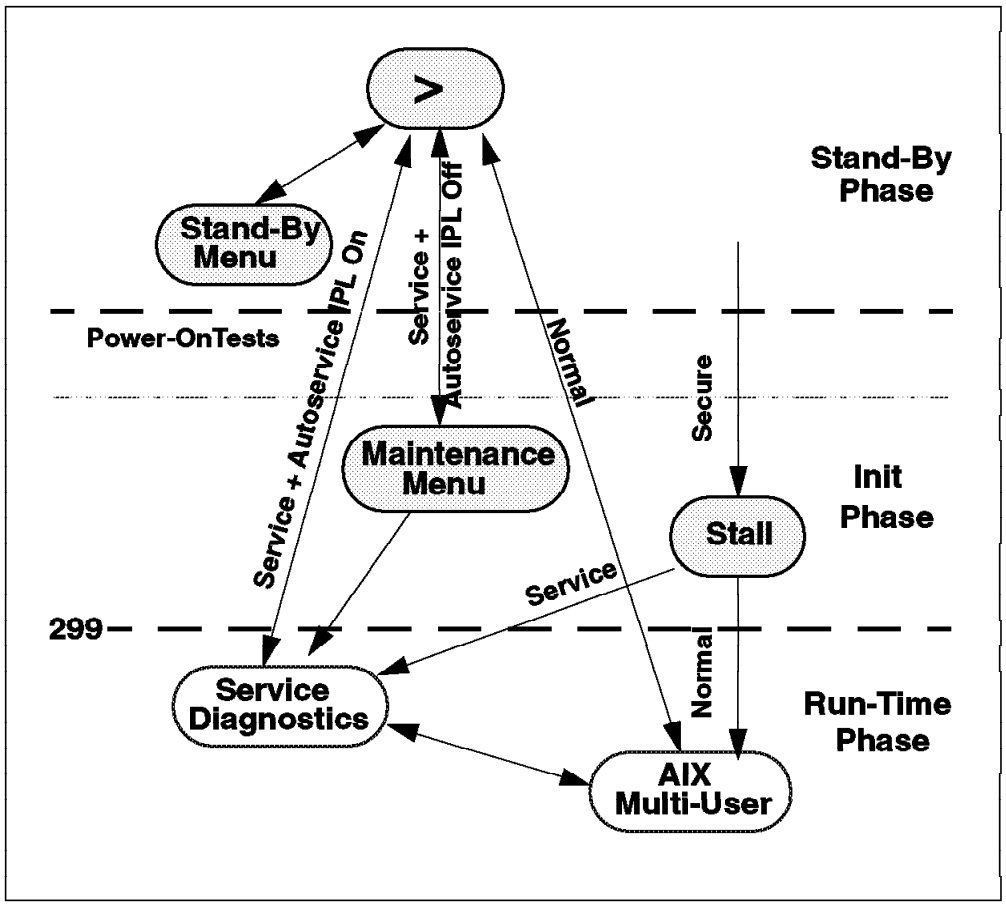


Figure 62. SystemGuard Phases

5.8.4 Phase Change (Stand-By to Init)

The phase change from Stand-By to Init is achieved either by entering power at the Stand-By prompt or, if the power switch on the front was pushed in to turn the system power off, by pushing that button once and then entering power. Note that if you type power while the power button is not pushed in, nothing happens until you press it. In this case, your power command has been received by SystemGuard, and you don't have to re-enter it.

Attention!

The word power is the default power-on command string. It can be changed by the system administrator from the Maintenance Menu, but caution should be taken to be sure the new power-on command string is not forgotten. A new string can only be set from the Maintenance Menu, and it can only be reached by issuing the power-on command string!

SystemGuard checks its own code in the EPROM, checks for a special downloadable floppy, checks the Flash EPROM, and then produces an output:

```
        BUMP FIRMWARE     - Mar 25, 1996
        ID 09.22 - POWER_ON in EPROM
#
       FLOPPY NOT READY!
DO YOU WANT TO UPDATE FLASH FROM LINE S2 {y/n} ? n

        BUMP FIRMWARE     - Jun 7, 1996
        ID 09.22 - POWER_ON in FLASH PROM
```

Figure 63. Phase Change from Stand-By to Init

The message FLOPPY NOT READY! means that there is no diskette in the diskette drive. At later firmware levels you will see a prompt asking if you'd like to download flash firmware from the S2 port. As of BUMP firmware 17.03 and flash firmware 9.23, that feature wasn't working, but was expected to in a later release.

The downloadable code referred to could contain:

- Code to be written into the Flash EPROM
- Code to change the VPD in the EEPROMs of the SMP system

Caution should be taken not to install a new release of flash firmware without checking to make sure it's compatible with the system's current BUMP firmware. BUMP firmware resides on the I/O planar and can only be updated by IBM Service. Flash firmware can be loaded by the customer after it's obtained from IBM Service or downloaded from the AIXTOOLS disk's SMP-FW package. Be sure to read the instructions that list compatible engineering change (EC) levels on key components.

Attention!

Installing flash firmware on a system with back-level BUMP firmware could cause unpredictable results. In this situation, either the BUMP firmware will have to be upgraded or the older, compatible level of flash firmware will have to be reinstalled.

5.8.5 Power-On Tests

Power-On (PON) tests are run by SystemGuard whenever the system power comes on. These tests come in two flavors:

- Minimal tests on processors, caches, memory, and related hardware which cannot be turned off.
- More extensive hardware tests, which can be turned off by setting the Fast IPL flag on from the Stand-By Menu.

Figure 64 on page 113 is an example of PON tests output when the Fast IPL flag is on.

```
        BUMP FIRMWARE   - Mar 25, 1996
        ID 09.22 - POWER_ON in EPROM


        BUMP FIRMWARE   - Jun 7, 1996
        ID 09.22 - POWER_ON in FLASH PROM

 - Low Interleaving  -
Initial test on CPU 0  - * OK !
Initial test on CPU 1  - * OK !
Initial test on CPU 2  - * OK !
Initial test on CPU 3  - * OK !
Init 1024kb L2 cache by processor 0 - * OK !
Init 1024kb L2 cache by processor 1 - * OK !
Init 1024kb L2 cache by processor 2 - * OK !
Init 1024kb L2 cache by processor 3 - * OK !
Clearing 128 Mb by processor 0 -> **** OK !

        CPU FIRMWARE    - Jun 7, 1996
        Processor 0 on IPL INIT

{{ 216 }}
{{ 220 }}
{{ 288 }}
{{ 278 }}
{{ 292 }}
{{ 286 }}
{{ 292 }}

Processor 0 on IPL Start

{{223}}
{{299}}
```

Figure 64. PON Tests Output with Fast IPL Flag On

A flashing 888 is displayed if PON tests cannot start. If PON tests hang, a three-digit code corresponding to a failed component is displayed.

Note: The system will IPL (boot) on the first available physical processor. If for any reason processor 0 is not available, the system will IPL on processor 1, and so on, until a good processor is found. If all the processors are disabled, PON tests will fail and SystemGuard will treat this as a hardware component failure and go into the Maintenance Menu in Service mode. In Normal mode, it will initiate dial-out, if enabled, and go into the Stall state afterwards. No IPL will proceed until the problem is fixed. Processors can be manually enabled again in Service mode through the Maintenance Menu. This can also be repaired locally by:

- Turning the system power off
- Moving the Physical Key into Service position
- Enabling at least one processor from the Stand-By Menu

There are other PON tests to check other system resources. These tests are a subset of the SystemGuard maintenance offline tests. They are resident within the Flash EPROM. These tests are divided into the following groups:

- BUMP Quick I/O Test Group: These tests check the accessibility and the functions of the standard and direct I/O components from the BUMP: S1, S2 and S3 lines, EEPROMs, NVRAM, Flash EPROM, and TOD (Time-Of-Day).
- JTAG (Joint Tests Action Group) Test Group: These tests check the chip connections using the JTAG features.
- Direct I/O Test Group: These tests check the accessibility of the Standard and Direct I/O components from the CPUs: IONIAN, NVRAM access, EPROM access, TOD, and the floppy disk. (IONIAN refers to the MicroChannel buses.)
- CPU Test Group: These tests, performed by all the processors, check the status of the CPU cards: processors, address translation, L1 and L2 caches.
- DCB (Data Crossbar) and Memory Test Group: These tests check the status of the system planar and memory cards, such as the data/address lines accessibility, memory components, ECC, and memory refresh (CPU checkstop).
- Interrupt Test Group: These tests collectively check the interrupt system: BUMP-CPU, CPU-CPU (CPU checkstop).
- CPU Multiprocessor Test Group: These tests check the multiprocessor mechanisms, atomic instructions, cache coherency, main memory sharing, and multiresource sharing.

The following screen is an example of PON test output when running:

```
               ***************
               *  PON TESTS  *
               ***************
.. Bump [01.01.00] DEBUG LINE TEST                              OK
.. Bump [01.02.00] S1 ASL (BUMP) TEST                           OK
.. Bump [01.03.01] S2 ASL (REM.) TEST                           OK
.. Bump [01.04.00] S3 ASL (SPE.) TEST                           OK
.. Bump [01.05.00] FLASH EP. CONTENT TEST                       OK
.. Bump [01.06.00] NVRAM CONTENT TEST                           OK
.. Bump [01.07.00] EPROM CONTENT TEST                           OK
.. Bump [01.08.00] TOD TEST                                     OK
.. Bump [01.09.00] FLOPPY-D CNT. TEST                           OK
.. Bump [01.10.00] BPP REGISTERS TEST                           OK
.. Bump [01.11.00] MISC. REGS TEST                              OK
.. Bump [06.05.00] TOD-BUMP IT TEST                             OK
..
..
```

Figure 65. PON Test Output

Note that these PON tests can be suppressed if the Fast IPL flag is enabled through SystemGuard.

5.8.6 Phase Change (Init to AIX Load and Run-Time)

Similar to the entry into the Init phase, there is a distinctive boundary when entering the Run-Time phase. At this point, SystemGuard gives up control of the system and passes it to AIX. This is indicated by the three-digit code 299 on the console and Operator Panel.

Since SystemGuard is also giving up control of the two serial lines, nothing can be displayed on the BUMP and Service consoles. The usual three-digit boot indicators are still displayed on the Operator Panel. Note that the code, 570, can take some time. The system will “walk” the SCSI buses and do several passes for each of the SCSI cards in the machine. This may take up to five minutes for each SCSI card in the SMP system.

When the boot indicators have reached c33, AIX has progressed enough to display its own boot messages on the console on S1. However, at this point, the console is the AIX system console, not the BUMP Console.

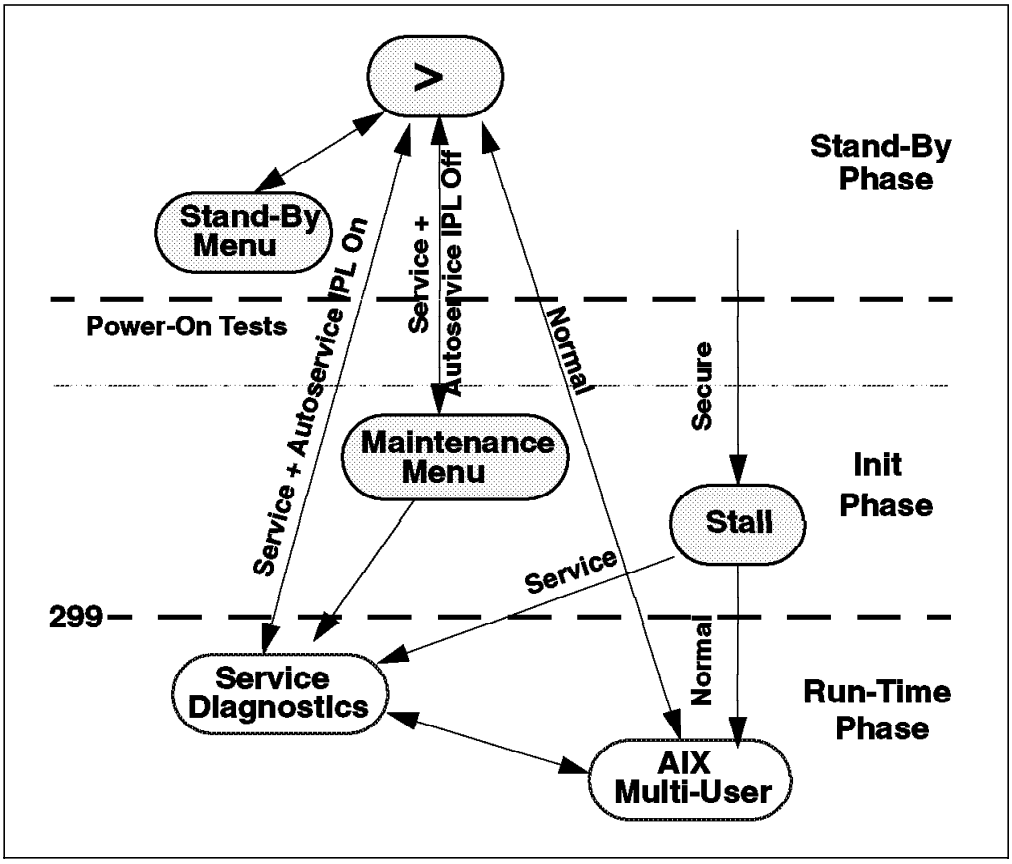


Figure 66. SystemGuard phases

Please refer to the *Service Guide* and/or *Operator Guide* for the three-digit codes and test groups.

5.9 SystemGuard Parameters and Flags

A certain number of SystemGuard parameters and flags (permissions) can be changed through the SystemGuard Menus, from AIX Diagnostics, and from the AIX command line. Basically, there are four groups of flags:

- Service Support flags: These flags enable the Service Console, maintenance usage, and determine if dial-out messages will be sent to IBM. These flags are stored in the SID (System Identification) field of the System EEPROM. They can only be updated by IBM Service.
- Diagnostics flags: These flags are used to control service, diagnostics, and maintenance functions. For example, the customer can set a flag to authorize a remote technician to change the Electronic Key setting or to enable dial-out in the event of a boot failure.
- Modem and Site Configuration flags: These flags allow the customer to customize the modem and port configuration for the Service Console.
- Phone Number flags: These fields contain the dial-in and dial-out phone numbers and the operator voice number.

5.10 Working with SystemGuard

You can change SystemGuard parameters and flags from different interfaces. They can be changed from the Stand-By Menu, the Maintenance Menu, the AIX Diagnostics Service Aids, and from the AIX command line.

The Stand-By Menu is stored in the BUMP EPROM. The Maintenance Menu is stored in the Flash EPROM.

Getting into these menus depends on the way flags are set when booting the system.

It is important to understand the flowchart in Figure 67 on page 117.

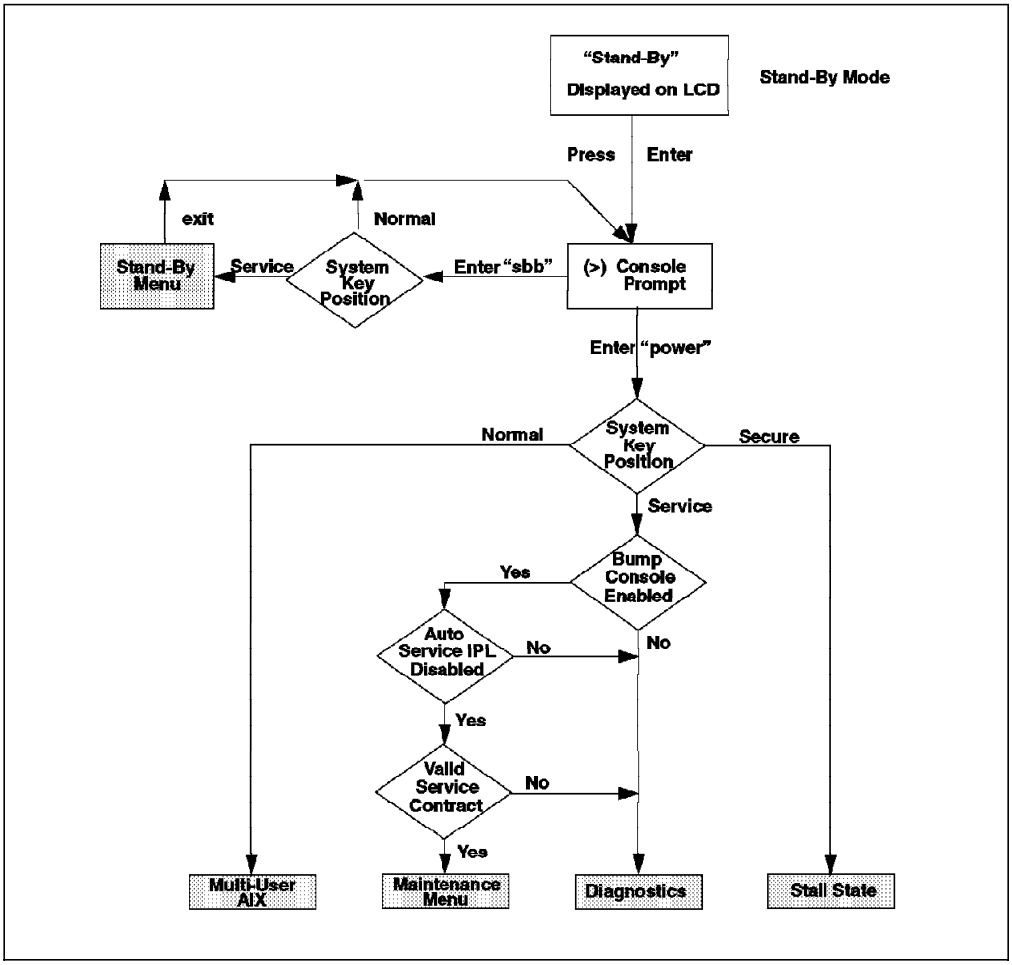


Figure 67. SystemGuard Flowchart

Basically, if the system is in Stand-By phase and the System Key is in Service, you can access the Stand-By Menu and work with SystemGuard.

If you power-on the system from Stand-By phase with the System Key in Normal position, the system will boot AIX.

If you power-on the system with the System Key in Secure, the system will stall. The LED 200 will appear. You must turn the key to Service or Normal to proceed.

If you power-on the system with the System Key in Service position, you can go to the Maintenance Menu or to AIX Diagnostics, depending on the state of three flags: BUMP Console Present, Autoservice IPL, and Service Contract Validity.

Here is what those flags mean:

- BUMP Console Present: When this flag is enabled, the LED codes and BUMP messages are displayed on the BUMP Console during the Init phase. If the flag is disabled, no codes or messages will be displayed during the Init phase. AIX messages appear when the system is nearly finished loading AIX, the same as on a RS/6000 uniprocessor. The default is for this flag to be enabled.

- The Autoservice IPL flag, if enabled, means that you want to go to AIX Diagnostics when booting with the System Key in Service or to multiuser AIX if the System Key is in Normal.
- The Service Contract flag is set by default to 32767 days, which means the contract is always valid. This enables an IBM Service Center to access the system and perform maintenance functions. If there is no Valid Service Contract, it is not possible to enter the Maintenance Menu or turn on console mirroring. You should never encounter this situation, but if you do, contact IBM Service.

5.11 SystemGuard Menus

Because little memory is available to store the SystemGuard firmware, Stand-By Menus are rather cryptic: They make extensive use of abbreviations or acronyms. You should refer to *7013 J Series Operator Guide* page x-1 for a list of these acronyms and abbreviations.

SystemGuard is menu-driven, and choices are usually numbered. Letters are sometimes used and can be entered in either lowercase or uppercase. The letter *x* is often used to exit the current menu and return to the one above it. Options are only processed after you press the **Enter** key. Until you press Enter, you can use the Backspace key to edit a selection. If you enter an option that does not match the available choices, a beep signals that an invalid selection has been made.

5.11.1 Stand-By Menu

The Stand-By Menu can only be entered when the system is in Stand-By phase, when the word Stand-By appears on the LCD display. If you don't see Stand-By, make sure that the system is plugged in.

In this mode, you will get the Stand-By prompt by pressing **Enter** on the BUMP Console. The Stand-By prompt is the greater-than (>) sign.

To enter the Stand-By Menu, you need to set the System Key in Service, either by moving the Physical Key or by setting the Electronic Key.

To set the Electronic Key to Service, perform these steps:

1. With the Physical Key in Normal, get the Stand-By prompt by pressing **Enter** on the BUMP Console.
2. Press **Enter** again to put the cursor on top of the prompt.
3. Press the **Esc** key once and then press the s key. This puts the Electronic Key in Service.
4. Press **Enter** again.
5. Enter the keyword sbb to display the Stand-By Menu.

The Stand-By Menu gives several options, as shown below:

```
  Stand-By Menu :  rev 17.03

0 Display Configuration
1 Set Flags
2 Set Unit Number
3 Set Configuration
4 SSbus Maintenance
5 I2C Maintenance









Select(x: exit): 0
```

Figure 68. Stand-By Menu

Note: It is also possible to enter the Stand-By Menu from the Service Console if the Remote Authorization flag is enabled. The Electronic Key can be set from the Service Console with the same escape sequence if permission has been granted by the customer.

For a detailed description of each option in the Menu and the meanings of each field, please refer to the following books:

- *7015 Model R30 CPU Enclosure Service Guide,* SA23-2743, Chapter 2, for the R series machines.
- *7013 J Series Operator Guide,* SA23-2724, Chapter 3, for the J series machines.
- *7012 G Series Operator Guide,* SA23-2740, Appendix B for the G series machines.

The Stand-By Menu allows the system administrator to display, in cryptic form, the physical configuration of the system (CPUs, memory, I/O, and so on) and to set flags, such as Fast IPL, to skip the second phase of PON tests. This Menu also allows the Customer Engineer to test the interconnection between the BUMP and the various hardware components through the I2C bus or the SSbus.

Attention!

Anyone not experienced with these functions should avoid these menus. It is possible to cause serious problems by changing values that are not fully understood.

5.11.2 Maintenance Menu

The Maintenance Menu also enables you to display the configuration of the system in a more easily understandable output, to perform various tests, to continue IPL, either from network, a specific SCSI device or from the bootlist, and to set various SystemGuard flags.

The Maintenance Menu can only be entered by:

1. Enabling the BUMP Console from the Stand-By Menu.
2. Setting the Autoservice IPL flag to disabled (the default value for this flag is disabled) from the Stand-By Menu.
3. Having a Valid Service Contract (this is the default).
4. Setting the System Key to Service, either by turning the Physical Key or setting the Electronic Key.
5. Powering-on the system.

The Maintenance Menu shown below should appear just after the 292 LED code is displayed on both the console and the Operator Panel LCD.

```
                        Maintenance Menu (Rev. 06.02)



              0> DISPLAY CONFIGURATION
              1> DISPLAY BUMP ERROR LOG
              2> ENABLE SERVICE CONSOLE
              3> DISABLE SERVICE CONSOLE
              4> RESET
              5> POWER OFF
              6> SYSTEM BOOT
              7> OFF-LINE TESTS
              8> SET PARAMETERS
              9> SET NATIONAL LANGUAGE



SELECT:
```

Figure 69. Maintenance Menu

5.12 SystemGuard and AIX

There are various commands available in AIX Version 4 for displaying and changing SystemGuard parameters. These commands allow the system administrator to carry out nearly all SystemGuard tasks from multiuser AIX, thereby greatly reducing or eliminating the need for a system reboot or operator intervention. This is ideal for remote support. AIX commands also allow you to perform tasks such as enabling and disabling processors and displaying and changing SystemGuard flag settings and the System Key mode.

Here are the AIX commands and examples of their output:

- `cpu_state {-l | -d Number | -e Number}`

  This command allows you to list processor status and disable or enable a processor. The `-l` option lists processors. The `-d` option disables a specific processor, and the `-e` option enables a processor. `Number` refers to the processor number, such as `proc0`, `proc1`, `proc2`, and so on. Note that disabling and enabling processors only takes effect at the next reboot. The following is an example of the `-l` option after a reboot.

```
Name       Cpu         Status         Location
proc0       0         enabled        00-0P-00-00
proc1       1         enabled        00-0P-00-01
proc2       2         enabled        00-0Q-00-00
proc3       -         disabled       00-0Q 00-01
```

- `mpcfg -d { -f -m -p -S}` for displaying flags

  `mpcfg -c { -f | -m |-p -S} <index> <value>` for changing flags

  `mpcfg { -r | -s }` for restoring or saving flag values in NVRAM

  `-r` Restores the parameters/flags to NVRAM from the file /etc/lpp/diagnostics/data/bump

  `-s` Saves parameters/flags from NVRAM into the file /etc/lpp/diagnostics/data/bump

  This is the meaning of the `mpcfg` command options:

  `-f` Indicates that the action (display or change) will be applied to the diagnostics flags.

  `-m` Indicates that the action will be applied to the modem and site configuration flags.

  `-p` Indicates that the action will be applied to the remote support phone numbers.

  `-S` Indicates that the action will be applied to the service support flags. **Note**: You will be able to display but unable to change the `-S` flag values.

- `keycfg -d` Displays the status of the System, Physical, and Electronic Keys, respectively. These keys are also called the Mode Switch, Key Mode Switch, and Electronic Mode Switch, respectively.

  `keycfg -c {service|secure|normal}` Used for changing the Electronic Key.

  Here is an example of the `keycfg -d` output:

```
Mode Switch      Key Mode Switch      Electronic Mode Switch
normal            normal              normal
```

  Note: The Physical Key overrides the Electronic Key; so the Physical Key must be in Normal if you want to set the Electronic Key.

- `mirrord`

  This daemon is used to implement the mirroring function between the BUMP Console and the Service Console. It's automatically started at boot time if the Service Contract is valid. It sleeps until mirroring is activated by putting the key in Service.

- `survd`

  This daemon implements a heartbeat protocol between SystemGuard and AIX. If SystemGuard has not received a message from AIX within the

specified delay time, SystemGuard assumes that AIX is hung and reboots the system. The survd daemon is not started automatically; only the root user has the ability to start or stop it.

```
survd -d <number of seconds>
```

This sets the heartbeat delay time. The default delay time is 60 seconds; the minimum value is 10 seconds. In a real-life situation, the delay will have to be long enough to avoid false reboots. If, for instance, the system is CPU bound and all processors are very busy, survd might not even get to the run queue. Therefore, SystemGuard would trigger a reboot.

```
survd -h
```

The flag -h issues a hardware reboot instead of a soft boot.

survd -r is the proper way to turn surveillance off.

Attention!

Issuing the following command:

```
kill -9 <survd_proc_id>
```

will result in a *reboot* of AIX since SystemGuard will stop getting messages from the daemon and assume that AIX is hung.

5.13 Processor and Memory Failure

As mentioned before, SystemGuard monitors the hardware and software and in case of failure, responds to address the failure.

In case of a processor failure, a checkstop occurs. SystemGuard takes control of the system and logs the event by saving the CPU registers image in NVRAM.

In the event of an unrecoverable memory failure, a checkstop occurs. The AIX error handler will log and report the failure:

- A dump is written to the dedicated disk area.
- The error is logged in the NVRAM.

SystemGuard then tries to recover the system. BUMP attempts to reboot the system and runs the Power-On tests.

1. If the reboot succeeds because the failure was of a transient nature, the system will come back with all resources enabled. AIX copies logout data from the NVRAM to a file and logs the event in the error log.
2. If the PON tests fail because of a solid hardware failure, SystemGuard will deconfigure the failed processor or memory bank and reboot the system.
3. If the boot fails, even in the reduced hardware configuration, the system displays an error code on the LCD and the BUMP Console. If dial-out is enabled, SystemGuard will report the problem to IBM Service. Then the system will enter an offline Maintenance Mode.

5.14 Some Common SystemGuard Tasks

The following tasks will be illustrated using the Stand-By and Maintenance Menus that are part of SystemGuard. These tasks can also be carried out from AIX Diagnostics or the AIX command line.

5.14.1 How to Set the Electronic Key

The key can be set electronically, making it possible to provide remote support without physically touching the machine. You can do this from the Stand-By or Run-Time phase.

5.14.1.1 Setting the Electronic Key from Stand-By Mode

1. Go into Stand-By phase.
2. Press **Enter** to get the prompt displayed (>).
3. Press **Enter** again to position the cursor on the prompt. The cursor (in block mode) is then superimposed on top of the prompt.
4. Press the **Esc** key and then the **s** key. This turns the Electronic Key to Service, even though the Physical Key is still in Normal position.
5. Press **Enter** again.
6. Enter the keyword, sbb (Stand-By BUMP). You should see the Stand-By Menu appear. (Hint: This is a way to check whether the System Key is in Service, since if it is not, entering sbb will not work and return to the Stand-By prompt.)
7. Exit from the Stand-By Menu.

At this step, if you want to go back to Normal, press **Enter** again to put the cursor on top of the prompt; then press the **Esc** key and the **n** key. This puts the Electronic Key in Normal position.

5.14.1.2 Setting the Electronic Key from AIX

1. While AIX is running, log in as *root*.
2. Type the following command to display the current status of the Physical, Electronic and System Keys:

   ```
   keycfg -d
   ```

3. To change the key to Service, type the following command:

   ```
   keycfg -c service
   ```

4. To change the key to Secure, type the following command:

   ```
   keycfg -c secure
   ```

5. To change the key back to Normal, type the following command:

   ```
   keycfg -c normal
   ```

5.14.2 How to Display the System Configuration

The system configuration can be displayed from either the Stand-By or Maintenance Menus.

5.14.2.1 Displaying Configuration From the Stand-By Menu

This option will display the system configuration table. This configuration can be viewed on the LCD of the Operator Panel if the console is configured. This is done by pressing the **reset** button with the Physical Key in Service.

To display the configuration, enter the Stand-By Menu and select **Display Configuration** (option 0). The first-level screen is displayed with features and devices that can be configured.

Here is an example:

```
                Display Configuration

SID TM          7013J30 45067              SID Y2               00045067
SID Y3  7fffff003935303730370000           UNIT  PAAAAAAA  40
CPU conf        CCCCAAAA                   MM conf          CCAACCAAAAAAA
FLASH_FW 0922   MM size  0080              OP_KEY  NRM      E_KEY   SRV
OPP   D78610    19H0494
SP    D78605    19H0471                    IOC   E38030     96G4400
CPU0  E59334    94H9994                    CPU1  E59334     94H9994
CPU2                                       CPU3
MC0   D78605    19H0473                    MC1   D78605     19H0473
MC2                                        MC3
SIB10 E38042    19H0310                    PS0   D29655     11H5114
SIB11                                      PS1
SIB21
SIB12                                      PS2
SIB22
SIB13                                      PS3
SIB23

Hit Return
```

Figure 70. Display Configuration Screen

Useful information can be obtained from this panel.

First, you can see in the SID TM field the type and model of the machine, 7013-J30 in our example.

Note: the model will still be a J30, even after the 604 upgrade.

The SID Y2 field contains a number that is used to build the uname of the machine. The serial number is the value in SID Y2 without the leading zeroes. Note that the number also appears as the second field in SID TM**.**

The UNIT field contains the number of units (base units and expansion units). P stands for present, while A stands for absent. In our case, we have only a base unit and no expansion unit.

The CPU conf field shows the status of the processors. In this field, C stands for configured, D for disabled, and A for absent. In our example, the system has four processors configured.

The MM conf field shows the memory configuration. This field has sixteen digits. Each digit gives the status of one memory bank. Please refer to Chapter 3,

"SMP Servers Architecture" on page 49, in this book to get more information on what constitutes a memory bank. As with the CPUs, C stands for configured, D for disabled, and A for absent. You can see in our example that we have four banks configured. Since the size of a bank depends on the type of memory cards installed on the system, you have to check the MM size field to get the amount of memory installed, and then calculate the size of one bank.

The MM size field gives you, in hexadecimal, the amount of memory installed on the system. In this case, we have 128 MB of memory (80 in hexadecimal). Since we have two 64 MB memory cards installed, the size of each bank is 32 MB.

The OP_KEY and the E_KEY shows the status of the Physical Key (Operator Panel Key) and the Electronic Key. In this example, the Physical Key is in Normal (NRM) and the Electronic Key is in Service (SRV).

The FLASH_FW field shows the level of firmware stored in the Flash EPROM.

5.14.2.2 Displaying Configuration through the Maintenance Menu

The system configuration can also be displayed through the Maintenance Menu. You will find the same kind of information displayed previously, but in a more readable fashion.

To get the system configuration from the Maintenance Menu, do the following:

1. Enter the Maintenance Menu.
2. Enter **0** to select **DISPLAY CONFIGURATION**.

The configuration display is a good picture of the SystemGuard configuration on one screen. Here is an example:

```
                  DISPLAY CONFIGURATION


MACHINE TYPE/MODEL: 7013J30 45067
FIRMWARE RELEASE:   Standby -> 1703
                    Backup eprom -> 0922
                    Flash eprom -> 0922
SERVICE CONTRACT:   Last update (yymmdd) -> 960726
                    Validity -> Unlimited contract
                    Remote service support -> Valid
                    Quick On Call service -> Not valid
AUTO DIAL:          Disabled
CONSOLES:           BUMP Consoles -> Present
                    Service Console -> Disabled - 2400 Bauds
SYSTEM ID:               00045067
NUMBER OF CPU:      4
MAIN MEMORY SIZE:   128 MByte
PRESENT UNITS:      #0



SELECT [Unit #(0-7) or x: exit]:
```

Figure 71. DISPLAY CONFIGURATION Screen

This screen is self-explanatory. An interesting feature here is that you can see the level of firmware stored in the BUMP EPROM as well as in the Flash and Backup EPROMs. Also, you can see the amount of memory without having to do a hexadecimal-to-decimal conversion in your head.

5.14.3 How to Set Fast IPL

If the Fast IPL flag is enabled, SystemGuard will skip the PON tests. By default, the Fast IPL flag is disabled; enabling it normally will only last one boot.

There are four ways to enable Fast IPL: through the Stand-By Menu, the Maintenance Menu, AIX Diagnostics Service Aids, and with AIX commands. Three methods are demonstrated here.

5.14.3.1 Setting Fast IPL from the Stand-By Menu

1. Set the System Key to the Service position.
2. Enter the Stand-By Menu by entering sbb.
3. Enter **1** to select **Set Flags**.
4. You'll note that 6 is the Fast IPL flag. It's current state is displayed at right. To toggle the value, press **6**.

The Fast IPL flag is now enabled; the PON Tests will not be run when the system boots. This will save several minutes.

```
Set Flags

0 Remote Authorization                    Disabled
1 Bump Console Present                    Enabled
2 Autoservice IPL                         Enabled
3 Extended Tests                          Disabled
4 PowerOn Tests in Trace Mode             Disabled
5 PowerOn Tests in Loop Mode              Disabled
6 Fast IPL                                Enabled
7 Set Electronic Mode Switch to Normal    SRV




Select(x:exit):
```

Figure 72. Set Flags Menu

5.14.3.2 Setting Fast IPL through the Maintenance Menu

1. Enter the Maintenance Menu.
2. Enter **8** to select the SET PARAMETERS Menu.
3. Enter **4** from the SET PARAMETERS Menu to select the MISCELLANEOUS PARAMETERS Menu.
4. Option 3 in this Menu should show the current status of the Fast IPL flag. If it is disabled, simply enter **3**, and the flag will be changed to enabled.

Fast IPL is now enabled and will last one reboot.

5.14.3.3 Setting Fast IPL through AIX

1. Log into AIX as user *root*.
2. Type in the following command to find the *index* of the Fast IPL flag and also the current flag value:

   ```
   mpcfg -df
   ```

   Following is the output of the command :

   ```
   Index   Name                                    Value
   1       Remote Authorization                    0
   2       Autoservice IPL                         0
   3       BUMP Console                            1
   4       Dial-Out Authorization                  0
   5       Set Mode to Normal When Booting         0
   6       Electronic Mode Switch from Service Line 0
   7       Boot Multi-User AIX in Service          0
   8       Extended Tests                          1
   9       Power On Tests in Trace Mode            0
   10      Power On Tests in Loop Mode             0
   11      Fast IPL                                0
   ```

3. The index is number 11, and the default value is zero (0), which means disabled.
4. Type the following command to change the status of the Fast IPL flag to enabled:

   ```
   mpcfg -cf 11 1
   ```

   Where 11 is the index and 1 the value itself; `c` is for change and `f` for diagnostics flags.

5. Type the following command just to verify the flag is changed :

   ```
   mpcfg -df
   ```

Hint!

In order to set the Fast IPL flag semi-permanently, an entry could be made at the end of /sbin/rc.boot issuing the AIX command to enable Fast IPL:

```
mpcfg -cf 11 1
```

Another possibility would be to change the flag from AIX, then issue the command `mpcfg -s` to save the flag values in NVRAM. After a boot, the flag will be set to disabled, but issuing the command `mpcfg -r` will restore the values saved in NVRAM. This could also be placed in /sbin/rc.boot.

Turning on Fast IPL should only be done during testing, when the machine needs to be rebooted often. It is not recommended that Fast IPL be turned off permanently in a production environment.

5.14.4 How to Set the Service Line Speed

The Service Line Speed should be set to 9600 Baud. This speed can be set through the SystemGuard Maintenance Menu. In order to use the Service Console properly, the terminal connected to S2 has to be set to the same speed. This speed is not necessarily the same as the speed defined in AIX for tty1. To avoid changing the speed of the terminal itself when AIX is running, it is more

convenient to have the same speed for the Service Console and tty1as defined in AIX.

5.14.4.1 Setting Line Speed through the Maintenance Menu

1. Enter the Maintenance Menu.
2. Enter **8** in this Menu to select the SET PARAMETERS Menu.

```
                          SET PARAMETERS


               0> POWER-ON COMMAND

               1> VOLTAGE MARGINS

               2> SET CONFIGURATION

               3> PHONE NUMBERS

               4> MISCELLANEOUS PARAMETERS



SELECT (x: exit):
```

Figure 73. SET PARAMETERS Menu

3. Enter **4** in this Menu to select the MISCELLANEOUS PARAMETERS Menu.

```
                       MISCELLANEOUS PARAMETERS


               0> BUMP CONSOLE -> Present
               1> AUTOSERVICE IPL -> Disabled
               2> DIAL_OUT AUTHORIZATION -> Disabled
               3> FAST IPL -> Enabled
               4> SET MODE TO NORMAL WHEN BOOTING -> Disabled
               5> BOOT MULTI-USER AIX IN SERVICE -> Disabled
               6> SERVICE LINE SPEED -> 2400 Bauds
               7> MAINTENANCE PASSWORD
               8> CUSTOMER MAINTENANCE PASSWORD
               9> ELECTRONIC MODE SWITCH FROM SERVICE LINE -> Disabled



SELECT (x: exit):
```

Figure 74. MISCELLANEOUS PARAMETERS Menu

4. Enter **6** to set this parameter. A menu showing possible line speeds is displayed.
5. Select a baud rate, and enter the corresponding menu number.
6. Exit from the menu.

5.14.4.2 Setting Line Speed through AIX

1. With AIX running, log in as *root*.
2. Type the following command to view current settings :

   ```
   mpcfg -dm
   ```

   The following is the output of the command :

   ```
   Index   Name                                          Value
   1       Modem Parameters File Name
   2       Service Line Speed
   3       Protocol Inter Data Block Delay
   4       Protocol Time Out
   5       Retry Number
   6       Customer ID
   7       Login ID
   8       Password ID
   ```

3. Type the following command to change to desired baud rate:

   ```
   mpcfg -cm 2 <line_speed>
   ```

   where `c` stands for change, `m` for modem and site configuration, `2` for the `Service Line Speed` index, and `<line_speed>` for your desired baud rate (9600, for example).

4. The line speed is changed, but will not be effective until a reboot of the system.

5.14.5 How to Authorize the Service Console

The Service Console must be authorized in order to work with SystemGuard. This allows remote support personnel to access SystemGuard. Service Console Authorization must also be activated to enable mirroring. There are three ways to do it:

5.14.5.1 Authorizing Service Console through the Stand-By Menu

1. Enter the Stand-By Menu.
2. Enter **1** from the Stand-By Menu to select the **SET FLAGS** Menu.
3. The Remote Authorization flag is 0 at the top of the menu. Its value is displayed at right. To toggle the value, press **0**.
4. Exit from the Stand-By Menu.

5.14.5.2 Authorizing Service Console through the Maintenance Menu

1. Enter the Maintenance Menu.
2. Enter **2** to select **ENABLE SERVICE CONSOLE**.
3. Exit from the Maintenance Menu.

5.14.5.3 Authorizing Service Console through AIX

1. With AIX running, log in as *root*.
2. Type the following command to view the current setting:

   ```
   mpcfg -df
   ```

3. Type the following command to enable the Service Console:

   ```
   mpcfg -cf 1 1
   ```

   Where `-cf` is for change flag; 1 is for the index. The last number, 1, is the value of the flag itself.

5.14.6 How to Set Up Console Mirroring

Console mirroring is an extremely valuable tool for both the customer and remote service personnel.

5.14.6.1 Console Mirroring Concepts

Console mirroring is a way for the customer to view what the person working remotely is doing on the system. When mirroring is active, the Service Console and the BUMP Console are logically identical, and both are tty0 (tty1 is disabled when mirroring starts).

Mirroring only works on the native serial ports, S1 and S2, and their respective ASCII consoles or terminal emulators. It does not work on graphical devices.

The BUMP Console can be either local (directly attached) or remote (through a modem connection). Remote console connection must be established via dial-in (BUMP will not dial out).

The Service Console is usually remote, connected via a Hayes-compatible modem on the S2 port. However, a local, directly attached Service Console at S2 port is also supported.

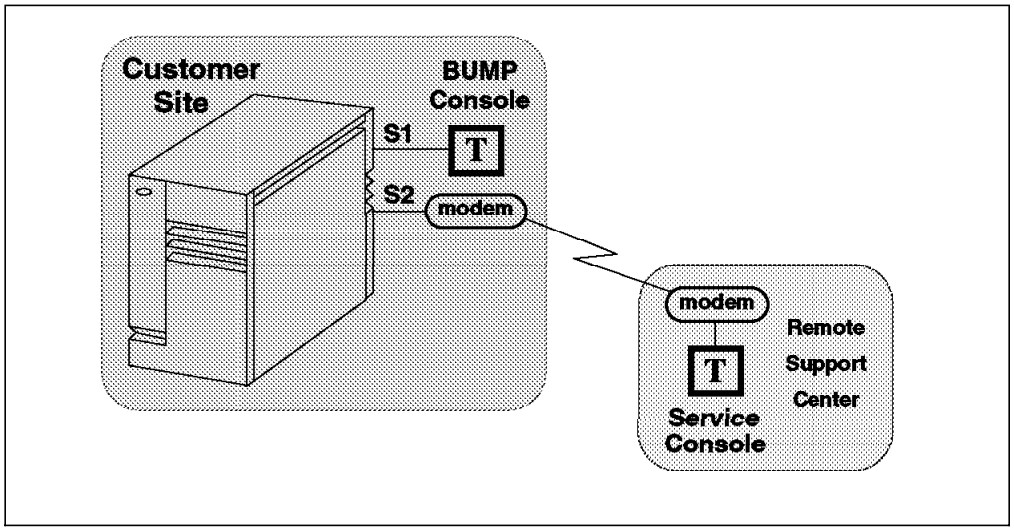


Figure 75. Console Connection

When mirroring is active, the customer on the BUMP Console and the support technician on the Service Console see the same output on their screens, and both may enter commands. One may start typing and the other finish. For example, the support person may log in as *root* on the Service Console, and the

customer can supply the root password from the BUMP Console. The remote person did not need to be told the root password.

Console mirroring is possible during all SystemGuard phases: Stand-By, Init and Run-Time.

The prerequisites for console mirroring are:

- Remote Service Support enabled (value of 1)
- Service Contract Validity 0 - 32767
- /usr/share/modems/mir_modem file present (for mirroring when AIX running)

5.14.6.2 Setting Up Console Mirroring

In order to set up console mirroring, you first need to authorize the Service Console and set the appropriate line speed. Refer to previous instructions on how to set up the service line speed and how to authorize the Service Console. Then do the following:

1. While AIX is running, log in as *root*.
2. Type the following command to ensure that Remote Service Support is enabled and the Service Contract Validity is greater than or equal to 0:

   ```
   mpcfg -dS
   ```

   The command output is:

   ```
   Index   Name                                    Value
   1       Remote Service Support                  1
   2       Quick On Call Service                   0
   3       Service Contract Validity               32767
   4       Service Support Type
   ```

   Note that mirrord cannot be started if the Service Contract Validity is -1 (No Valid Service Contract).
3. Wake up mirrord by either switching the Physical Key to Service or by typing the following command:

   ```
   keycfg -c service
   ```
4. If mirrord is awakened successfully, you should see the following messages:

   ```
   mirrord: Wait connection...
   mirrord: Remote user connected, mirroring active
   ```
5. Type the following command to verify that mirrord is running:

   ```
   ps -ef | grep mir
   ```

   The command output is the following:

   ```
   root  2308     1   0 12:08:21      -  0:00 /usr/sbin/mirrord mir_modem
   root  6212  4552   3 12:21:58      0  0:00 grep mir
   ```
6. Support personnel now should be able to work from the Service Console and the customer should be able to observe from the BUMP Console.
7. To turn mirroring off, either switch the Physical Key to Normal or type the command:

   ```
   keycfg -c normal
   ```

   You should see the following message:

   ```
   mirrord: mirroring is stopped
   ```

If the prerequisite conditions are met, the mirrord daemon is started at boot time, but goes to sleep until the System Key is changed to Service. When mirrord is awakened, it kills all processes on S2 and pushes the streams mirror module onto the S2 queue.

Since it is assumed that the Service Console is remote, mirrord requires a modem description file that specifies the characteristics of the modem. This modem file is required even if the Service Console is connected locally without a modem. Thus, a no-modem file also must be provided. The default name of the modem description file is mir_modem. Sample files are found in the /usr/share/modems directory. Please refer to Appendix A, "SystemGuard Remote Operation Configuration" on page 325 for supported modem files and information about how to initialize a modem configuration.

5.14.7 How to Enable Surveillance

Surveillance is implemented with the survd daemon. This daemon, when started, establishes a heartbeat between AIX and SystemGuard. In case of an AIX hang, SystemGuard detects it and reboots the system.

To implement the surveillance, do the following:

1. Enter: survd -d {number of seconds}

   This starts the survd daemon, thereby starting surveillance. The number of seconds determines the heartbeat delay time, where 10 seconds is the minimum and the default is 60 seconds.

2. Carry out this step if you want a hardware reboot: survd -h (-h flag sets hardware reboot).

3. To turn off the surveillance, type the following command: survd -r

Attention!

If you issue the command kill -9 <survd_proc_id>, the system will reboot because SystemGuard will think that AIX is hung, since it will no longer receives a heartbeat.

5.14.8 How to Set Up the Dial-Out Feature

The dial-out feature can be implemented through SystemGuard or AIX.

5.14.8.1 Setting Up Dial-Out from SystemGuard

The dial-out feature enables automatic transmittal of boot failure information to a remote service center. The customer must set the Dial-Out Authorization flag. When the Physical Key is in the Normal position, if a boot fails due to a PON test error, or if the boot device cannot be found, a string of data is sent to the remote service center. The data includes the Service Request Number for the failure and other pertinent information about the system.

If the data is sent to an IBM Service Center, the information is extracted and placed in a problem record. IBM Service personnel will call the customer to find out if assistance is requested.

If the customer sends the failure data to his own help desk, he must provide a "catcher" program to interpret the data stream. Sample as-is "catcher" code can be found in /usr/samples/syscatch in AIX V4.1 and AIX V4.2.

When using dial-out to contact IBM Service, be sure a valid customer number is stored in the modem and site configuration flags. These are most easily accessed via AIX using the `mpcfg` command and the `-m` option. Please refer to Appendix A, “SystemGuard Remote Operation Configuration” on page 325, for detailed instructions on setting these values.

The prerequisites for the dial-out feature are:

- Remote Service Support flag enabled (1)
- Valid Service Contract 0 to 32767 days
- Remote authorization enabled (1)
- Dial-out authorization enabled (1)

The dial-out feature uses the dial-out phone numbers listed in SystemGuard configuration. To add or change phone numbers, do the following:

1. Enter the SystemGuard Maintenance Menu.
2. Enter **8** from this Menu to select the **SET PARAMETERS** Menu.
3. Enter **3** from the SET PARAMETERS Menu to select the **PHONE NUMBERS** Menu.
4. Select your option, and enter the dial-out number or numbers. There are two Service Center and Customer Hub numbers. These relate to primary (1) and secondary (2) numbers.

The data that gets sent to the Remote Service Center is 256 bytes long and includes the following fields:

PARAMETER	SIZE
Magic Number	4
Routing Metric	4
Login-ID	12
CSS-ID	4
RETAIN Account # or customer ID	12
Password general	16
Time stamp	8
Customer system phone	20
Customer operator phone	20
Machine serial #	10
Machine device type	13
Primary error code	4
Destination (service center)	1
SRN LCD Code	64
Text problem abstract	64

5.14.8.2 Error Notification Using Console Mirroring

Problem reporting can also be implemented at the operating-system level. You can use SystemGuard, AIX, and the Object Data Manager (ODM) to develop a procedure for notifying a remote help desk in the event a predetermined problem occurs on the system.

To do this, you can use an ODM object class called errnotify. Whenever a problem occurs matching criteria established in errnotify, the system can invoke a script to turn on mirroring and send a message to a remote site via the Service Console.

Here is an example of a file, /tmp/sample.add, which is going to be used for reporting tape-drive errors to a Remote Service Center.

```
errnotify:
   en_pid = 0
   en_name = "sample"
   en_persistenceflg = 2
   en_label = ""
   en_crcid = 0
   en_class = "H"
   en_type = "PERM"
   en_alertflg = ""
   en_resource = ""
   en_rtype = ""
   en_rclass = "tape"
   en_symptom = ""
   en_method = "/u/errnotify.script"
```

Create the above file, /tmp/sample.add. Add it to the ODM by typing the following command:

```
odmadd /tmp/sample.add
```

Once this is added to the ODM, automatic error notification is running. The errnotify.script is executed as soon the tape errors are logged twice (determined by en_persistenceflg = 2).

To display the ODM object, or to delete the objects, the following commands can be used:

- `odmget -q"en_name='sample'" errnotify`
- `odmdelete -q"en_name='sample'" -o errnotify`

The /u/errnotify.script could look something like this:

```
keycfg -c service
clear
echo "tape drive problems, look at /err.log for details > /dev/tty0"
sleep 5
errpt -a -l $1 >> /err.log
keycfg -c normal
```

In this example, console mirroring is activated by turning the Electronic Key to service. Next, the script writes a message to the BUMP and Service Consoles and sleeps for five seconds. The error log entry for the tape drive is appended to the err.log file, and then console mirroring is turned off. This can be used as an example to customize individual customer environments.

5.14.9 How to Reboot AIX from the Remote Service Console

It is possible to boot AIX remotely via the Service Console.

5.14.9.1 Prerequisites

The following procedure must be carried out from the BUMP Console in order to allow someone using the Service Console to reboot the system:

1. Get into the Maintenance Menu.
2. Enter **8** in the Maintenance Menu to select the **SET PARAMETERS** Menu.

3. Enter **0** in the SET PARAMETERS Menu to select the **POWER-ON COMMAND** Menu.
4. Enter **3** and enable Service Console Power-On.
5. Enter **5** from the same menu and enter the string `power`. This string will be used for powering-on the system from the Service Console.
6. Now reboot the system in Normal mode, and log in as *root*.
7. Type the following commands:

   `mpcfg -cf 1 1` to enable Remote Authorization

   `mpcfg -cf 2 1` to enable Autoservice IPL

   `mpcfg -cf 6 1` to enable Electronic Mode Switch from S2

   `mpcfg -cf 7 1` to enable Boot Multi-User AIX in Service

   `mpcfg -cf 11 1` to enable Fast IPL
8. Type the following command to start console mirroring:

   `keycfg -c service`

Now the system is set up so that the Service Console can activate a reboot. The system can be booted either to multiuser AIX or to AIX Diagnostics for tests on the hardware.

5.14.9.2 Rebooting to AIX Multiuser

1. While AIX is running, log in as *root*.
2. Type the command `shutdown` to shut down the system. You can use the `-F` flag for a fast shutdown and/or a `-t` flag for a shutdown at a particular time.
3. Once the system has shut down, you should be able to press **Enter** and see the Stand-By prompt (>).
4. At this prompt, type `sbb` to get into the Stand-By Menu.
5. Enter a **1** at the Stand-By Menu to select the SET FLAGS Menu.
6. Enter a **7** to select **Set Electronic Mode Switch to Normal**.
7. Once the Electronic mode switch is set to Normal, exit out of the Stand-By Menu until you have your prompt (>).
8. Type `power` at this prompt; the system should reboot. You should get the AIX login prompt in about ten minutes.

Note that because the System Key was changed to Normal in step 6, the remote service technician will be disconnected temporarily while the system boots. After AIX is running, a login will appear.

5.14.9.3 Rebooting to Single-User and then to Multiuser

This allows the remote technician connected via the S2 port to shut down and reboot the system in AIX Diagnostics for hardware problem determination. After running diagnostics, the technician can reboot the system in AIX Multiuser without having to be on-site.

1. While AIX is running, type `shutdown` to bring down the system.
2. Once the system is shut down, you should see the Stand-By prompt (>). Remember, at this point the Electronic switch is in Service mode. Leave it in Service mode.

3. At the Stand-By prompt, type the power-on keyword for the Service Console (power in our example).
4. The system will reboot in AIX Diagnostics. From here, diagnostic tasks can be performed.
5. Once completed, activate a single-user boot from the Diagnostic's main menu.
6. You will be prompted for a password; enter the root password.
7. Type the following command to initialize multiuser AIX:

   ```
   init 2
   ```

8. After a few minutes, the system should have completed a multiuser initialization.
9. If S2 was configured as an AIX tty, an AIX login screen should appear on the remote Service Console.

5.14.10 How to Boot from an SCSI Device

The SMP server can be booted in Service mode from a desired SCSI device, either from the Maintenance Menu or through the bootlist.

5.14.10.1 Booting from an SCSI Device through the Maintenance Menu

When the system is powered on with the System Key in Service, it either comes up in the Maintenance Menu, starts AIX Diagnostics or boots from a SCSI device, depending upon the flag settings.

Note: To be able to boot from an SCSI device other than the boot disk, such as a tape drive, the flag BOOT MULTI-USER AIX IN SERVICE must be disabled. This can be set through the Maintenance Menu. To check or change this flag, do the following:

1. Enter the Maintenance Menu; please refer to the Maintenance Menu section in this book for details.
2. From the Maintenance Menu, enter **8** to select the SET PARAMETERS Menu.
3. From this Menu, enter **4**. to select the **M**ISCELLANEOUS PARAMETERS Menu.

```
                          MISCELLANEOUS PARAMETERS


              0> BUMP CONSOLE -> Present
              1> AUTOSERVICE IPL -> Disabled
              2> DIAL_OUT AUTHORIZATION -> Disabled
              3> FAST IPL -> Enabled
              4> SET MODE TO NORMAL WHEN BOOTING -> Disabled
              5> BOOT MULTI-USER AIX IN SERVICE -> Disabled
              6> SERVICE LINE SPEED -> 2400 Bauds
              7> MAINTENANCE PASSWORD
              8> CUSTOMER MAINTENANCE PASSWORD
              9> ELECTRONIC MODE SWITCH FROM SERVICE LINE -> Disabled



SELECT [x: exit]:
```

Figure 76. MISCELLANEOUS PARAMETERS Menu

4. Check option 5 in this Menu (BOOT MULTI-USER AIX IN SERVICE). If it is enabled, enter **5**, and the flag should be changed to disabled.
5. Have another look at the option to make sure it is disabled.

   Now we are ready to boot from a SCSI device, such as a tape drive or CD-ROM, for example.
6. Insert the bootable media in the SCSI device.
7. Exit back to the main Maintenance Menu.
8. Enter **6** in the Maintenance Menu to select the SYSTEM BOOT Menu.

```
                          SYSTEM BOOT


              0> BOOT FROM LIST

              1> BOOT FROM NETWORK

              2> BOOT FROM SCSI DEVICE









SELECT [x: exit]: 2
```

Figure 77. SYSTEM BOOT Menu

9. Enter **2** to boot from an SCSI device. The menu that appears enables you to specify the SCSI device by using the location code.
10. At this point, a BOOT FROM SCSI DEVICE screen appears. This will display the PRESENT DEVICE LOCATION CODE.

```
                          BOOT FROM SCSI DEVICE


PRESENT DEVICE LOCATION CODE:
(Drawer - Bus#/Slot# - Connector - SCSI ID/LUN) 00010060




COMMANDS: 0> CHANGE BUS#
          1> CHANGE SLOT#
          2> CHANGE SCSI ID
          3> CHANGE LUN ID
          4> CHANGE DEVICE LOCATION CODE
          5> BOOT FROM SELECTED DEVICE

SELECT [x: exit]: 5
```

Figure 78. SCSI Boot Device Location Code

If it is not the device you want, go through each option and change it to the desired BUS, SLOT, SCSI ID, and LUN ID. Option 4 allows you to change all these options at once.

Following is the description for each of these options:

- BUS: whether it is internal (0) or external (1).
- SLOT: actual physical slot number; internal bus can be 1 to 7 and external can be 1 to 8.
- SCSI ID: the SCSI address of the SCSI device.
- LUN ID: logical unit number; for an 8-bit bus this can be 0 to F, and for a 16-bit bus, this can be 00 to 1F.

11. Once the desired device is selected, enter **5** to start booting. Then the system leaves the Maintenance Menu and boots from the specified SCSI device.

In Figure 78 on page 138, the selected device is connected to an internal SCSI bus located in slot 1, and the device has the SCSI address 6 on that SCSI bus.

5.14.10.2 Booting from an SCSI Device through the Bootlist

The system can be booted from a SCSI device, such as a tape drive, without going through the Maintenance Menu. In this case, it uses the bootlist to determine the boot device while in Service mode. The bootlist can be updated through Service Aids in Diagnostics.

Following are the prerequisites for booting from an SCSI device (a tape drive for example) in Service mode:

- The Autoservice IPL flag must be enabled.
- The Service mode bootlist must be updated (this can be done in AIX Diagnostics by choosing **Service Aids** and **Display/Alter Bootlist**).
- The selected SCSI device must be supported and bootable, for example, an 8mm tape drive.
- Bootable SCSI media, such as a tape (IBM product or mksysb), must be available.

To boot from this SCSI device, do the following:

1. Switch the System Key to Service.
2. Insert the bootable media in the SCSI device.
3. Turn on the system power.

The system will then boot up from this SCSI device. If the bootup is not successful, verify that the tape is bootable, or clean the SCSI device.

5.14.11 How to Boot from the Network

The system can be booted from the network through the Maintenance Menu. Network boot allows a system to be installed via the network and also allows various maintenance tasks to be carried out on the local machine. Use the following procedure to boot from the network:

1. Enter the Maintenance Menu.
2. From the Maintenance Menu, enter 8 to select the SYSTEM BOOT Menu.

```
                          SYSTEM BOOT


               0> BOOT FROM LIST

               1> BOOT FROM NETWORK

               2> BOOT FROM SCSI DEVICE









SELECT [x: exit]: 1
```

Figure 79. SYSTEM BOOT Menu

3. From the SYSTEM BOOT Menu, enter **1** to select **BOOT FROM NETWORK**.

```
MAIN Menu


1.  Select BOOT (Startup) Device
2.  Select Language for these Menus
3.  Send Test Transmission (PING)
4.  Exit Main Menu and Start System (BOOT)









Type the number for your selection, then press "ENTER"
(Use the "Backspace" key to correct errors)
```

Figure 80. Network Boot MAIN Menu

4. From the NETWORK BOOT MAIN Menu, enter a **1** to select the **Select BOOT (Startup) Device** option.
5. The SELECT BOOT (STARTUP) DEVICE Menu appears.

```
SELECT BOOT (STARTUP) DEVICE


Select the device to BOOT (Startup) this machine.


WARNING: If you are using Token-Ring, selection of an
incorrect data rate can result in total disruption of the
Token-Ring network.
"==>" Shows the selected BOOT (startup) device

==>  1. Use Default Boot (Startup) Device
     2. Token-Ring:  Slot 3, 4 Mb data rate
     3. Token-Ring:  Slot 3, 16 Mb data rate
     4. Ethernet:  Slot 4, 15-pin connector
Page 1 of 2

88. Next Page of Select BOOT (Startup) Device Menu
99. Return to Main Menu

Type the number for your selection, then press "ENTER"
(Use the "Backspace" key to correct errors)
```

Figure 81. SELECT BOOT (STARTUP) DEVICE Menu

6. Select the device to boot from. For example, choose **3** to boot from Token-Ring: slot 3, 16 Mb data rate.

   The following screen appears:

```
SET OR CHANGE NETWORK ADDRESSES

Select an address to change
Currently selected BOOT (startup) device is:

Token-Ring:  Slot 2, 16 Mb data rate
Hardware address ..................................... 10005AC97CF1

1. Client address                                      009.003.001.027
     (address of this machine)
2. BOOTP server address                                009.003.001.008
     (address of the remote machine you boot from)
3. Gateway address                                     000.000.000.000
     (Optional, required if gateway used)

97. Return to Select BOOT (Startup) Device Menu (SAVES addresses)
99. Return to Main Menu (SAVES addresses)



Type the number for your selection, then press "ENTER"
(Use the "Backspace" key to correct errors)
```

Figure 82. SET or CHANGE NETWORK ADDRESSES Menu

7. Enter the appropriate IP addresses, and enter **99** to return to the MAIN Menu.

```
MAIN Menu


1.  Select BOOT (Startup) Device
2.  Select Language for these Menus
3.  Send Test Transmission (PING)
4.  Exit Main Menu and Start System (BOOT)









Type the number for your selection, then press "ENTER"
(Use the "Backspace" key to correct errors)
```

Figure 83. Network Boot MAIN Menu

8. Enter **4** to exit from the Menu and start system boot.
9. The following screen appears as the system boots off the network:

```
STARTING SYSTEM (BOOT)









Booting . . .  Please wait.


Token-Ring:  Slot 2, 16 Mb data rate


Hardware address ..................................... 10005AC97CF1


              Packets Sent        Packets Received

BOOTP             00000               00000
```

Figure 84. Network Boot Proceeding

To find out more information on NIM (Network Install Manager), please refer to *AIX Version 4.1 Network Installation Management Guide and Reference,* SC23-2627.

5.14.12 How to Disable and Enable Processors

In the SMP servers, it is possible to disable/enable processors. A suspected faulty processor can be disabled so that the system can run without it. The processors can be disabled/enabled through the Stand-By Menu, Maintenance Menu, Diagnostics, or through AIX commands.

5.14.12.1 Enabling/Disabling Processors through the Stand-By Menu

1. Enter the Stand-By Menu; refer to the Stand-By Menu section in this book.
2. From the Stand-By Menu, enter **3** to select the **Set Configuration** Menu. A first-level screen similar to the one below appears:

```
   Set Configuration

00 CPU0                 09 UNIT0 & dev
01 CPU1                 10 UNIT1 & dev
02 CPU2                 11 UNIT2 & dev
03 CPU3                 12 UNIT3 & dev
04 MC0                  13 UNIT4 & dev
05 MC1                  14 UNIT5 & dev
06 MC2                  15 UNIT6 & dev
07 MC3                  16 UNIT7 & dev
08 basic MCA            17 exp  MCA









Select(x: exit): 01
```

Figure 85. Set Configuration Menu

The Set Configuration screen displays the units and devices that can be configured, along with their menu index number. At this step, CPU0 stands for the CPU card 0, not processor 0.

3. Enter an index number for a CPU card to be looked at. We will select **CPU1**, (01) in this example. The following screen appears:

```
CPU1  Set          | Status


00 CPU0  C           C
01 CPU0  D
02 CPU0  T
03 CPU1  C           C
04 CPU1  D
05 CPU1  T









Select(x: exit): 04
```

Figure 86. CPU Status

4. All the CPUs and their statuses are displayed where:
   - C stands for configured.
   - D stands for disabled.
   - T stands for temporarily disabled. It means that at the next power-on or reset, the device is automatically reconfigured.
5. Enter **04** to deconfigure CPU1. You should see the status changed to D, disabled.

```
CPU1  Set          | Status

00 CPU0  C           C
01 CPU0  D
02 CPU0  T
03 CPU1  C           D
04 CPU1  D
05 CPU1  T









Select(x: exit):
```

Figure 87. CPU Status

6. Now, once the system is rebooted, it will be running without one processor on card CPU1.
7. To enable the processor, follow the same procedure as above, except choose status **C**, (03 in above example).
8. Exit the Stand-By Menu, and continue booting the machine.

5.14.12.2 Disabling Processors through the Maintenance Menu

1. Enter the Maintenance Menu. Refer to the Maintenance Menu section of this book.
2. Enter **8** to select the **SET PARAMETERS** Menu.
3. Enter **2** from the SET PARAMETERS Menu to select the **SET CONFIGURATION** Menu.

```
                          SET PARAMETERS


               0> POWER-ON COMMAND

               1> VOLTAGE MARGINS

               2> SET CONFIGURATION

               3> PHONE NUMBERS

               4> MISCELLANEOUS PARAMETERS



SELECT [x: exit]: 2
```

Figure 88. SET PARAMETERS Menu

```
                         SET CONFIGURATION


               0> CPU CARD

               1> MEMORY CARD

               2> BASIC MCA ADAPTERS









SELECT [x: exit]: 0

CPU CARD [0> CPU0| 1> CPU1 or x: exit]: 1
```

Figure 89. SET CONFIGURATION Menu

4. Enter **0** from this Menu to select the **CPU CARD** option.
5. The CPU CARD screen appears and looks similar to this:

```
                          CPU CARD - (CPU1)


PRESENT CONDITIONS: PR #0 -> Valid & Enabled
                    PR #1 -> Valid & Enabled









COMMANDS: 0> ENABLE
          1> DISABLE
          2> TEMPORARY DISABLE

SELECT [x: exit]:
```

Figure 90. CPU CARD Status

6. From this screen, you can disable or enable a particular processor on the selected CPU card. Option 0 enables a CPU; option 1 disables a CPU, and option 2 temporarily disables a CPU until the next reboot.

5.14.12.3 Enabling/Disabling Processors through AIX

The processors can be disabled/enabled through AIX as well. This is done through the cpu_state command. Following is the command with various options:

- To list the processors and view their statuses, type the following command: `cpu_state -l`

  The output should look something like this:

```
Name    Cpu      Status         Location
proc0   0        enabled        00-0P-00-00
proc1   1        enabled        00-0P-00-01
proc2   2        enabled        00-0Q-00-00
proc3   3        enabled        00-0Q-00-01
```

- To disable a CPU, proc1 for example, type the following command:

  `cpu_state -d proc1`

  Now look at the result of `cpu_state -l`:

```
Name    Cpu      Status         Location
proc0   0        enabled        00-0P-00-00
proc1   1       disabled        00-0P-00-01
proc2   2        enabled        00-0Q-00-00
proc3   3        enabled        00-0Q-00-01
```

Note: This change does not take effect until after a reboot. After a reboot, the `cpu_state -l` command will show:

```
Name    Cpu      Status         Location
proc0   0         enabled       00-0P-00-00
proc1   -        disabled       00-0P-00-01
proc2   1         enabled       00-0Q-00-00
proc3   2         enabled       00-0Q-00-01
```

The reason there is now a - beside proc1 is that it was not configured into the system when the SMP rebooted.

5.14.13 How to View the BUMP Error Log

If you are experiencing problems with firmware, IBM Service may ask you to look at the BUMP firmware error log. It is accessible from the Maintenance Menu.

To display the contents of the logging buffer:

1. Enter the Maintenance Menu.
2. Select option **1**, **Display BUMP Error Log**.
3. Use the + and - keys to scroll events on the screen forward and backward, respectively.
4. Press **x** to exit and return to the main menu.

```
                        DISPLAY BUMP ERROR LOG

EVENT #1:

aa0000000000000000bbcc0000000000000000ddee0000000000000000ff00000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000




SELECT [+:forward event or x:exit]:
```

Figure 91. Sample BUMP Error Log Buffer

5.14.14 How to Perform Offline Tests

The Power-On tests conducted during the transition from the Stand-By to the Init phase can be conducted in a controlled and interactive mode from the Off-Line Tests menu. One of these tests is for BUMP dial-out, which is described on page A.5, “Testing Dial-Out” on page 331.

To conduct the tests, follow these steps:

1. Go the Maintenance Menu and select **7**, **OFF-LINE TESTS**.
2. You can modify the option list field by entering the command corresponding to a particular option. If you want to deselect a particular option, press the

same command once again. In other words, the options are toggled by the corresponding commands. The available commands are explained as follows:

```
                                    OFF-LINE TESTS

OPTION LIST:
PROCESSOR LIST: Test parameter value

COMMANDS:   A> VERBOSE
            B> VERBOSE++
            C> HALT ON ERROR
            D> LOOP
            E> SCOPE
            F> PROCESSOR

            0> BUILD TEST LIST
            1> MODIFY/DISPLAY TEST LIST
            2> DELETE TEST LIST
            3> EXECUTE TEST LIST
            4> ERROR REPORT INSPECTION

SELECT [x:exit]:
```

Figure 92. OFF-LINE TESTS Menu

A Set the verbose option, which displays test execution messages. Without this option, only test titles and results are displayed.

B Sets the extra verbose option, which displays the verbose messages, plus detailed test execution messages. If this option is selected, the verbose option is redundant and ignored.

C Sets the halt on error option, which forces test execution to stop when the first error is found. In this case, the error message and status are displayed.

D Sets the loop option, which continuously runs selected tests in a loop while displaying the test identifier and loop step counter. To stop the tests, press the Break key.

E Sets the scope option, which continuously runs the test (in the list of selected tests) in a loop, without displaying messages. To stop the tests, press the Break key.

0 Displays the Build Test menu, which enables you to specify the test list.

1 Displays and enables modification of the tests in the build list.

2 Deletes the tests in the build list, after operator confirmation. Attempting to delete tests from an empty list causes an error message to be displayed.

3 Runs selected tests.

4 Displays an error report.

x Exits this menu and returns to the Maintenance Menu

3. After selecting **VERBOSE** or **Verbose++**, select **0** to go to the BUILD TEST LIST menu.

```
                                BUILD TEST LIST

GROUP        DESCRIPTION                        GROUP        DESCRIPTION

01           BUMP QUICK IO                      03           DIRECT IO
04           CPU                                05           DCB AND MEMORY
06           INTERRUPT                          10           MCA
20           MULTIPROCESSOR                     50           JTAG


COMMANDS: nn> SELECT GROUP
           *> ADD ALL AVAILABLE TESTS TO LIST

SELECT [x:exit]:
```

Figure 93. BUILD TEST LIST Menu

4. From the BUILD TEST LIST menu, select the Test Groups you wish to use to build your test list. Tests will be run in the sequence they are chosen. The test groups are chosen by their Group Number. If you want to run tests from more than one group, separate the Group Numbers with a slash character (/).
5. After a Group Number has been selected, a list of tests will be displayed. Select the test you want to run by its number. You can use an asterisk to select all tests in a Group. If you want to select more than one test, but not all, use a slash character (/) to separate test numbers.

```
                           BUILD TEST LIST

GROUP 01 BUMP QUICK IO

TEST     DESCRIPTION                    TEST     DESCRIPTION

01       DEBUG LINE                     02       S1 LINE (BUMP)
03       S2 LINE (Remote)               04       S3 LINE (SPECIFIC)
05       FLASH EPROM                    06       NVRAM
07       EPROM                          08       TOD
09       FLOPPY DISK                    10       BPP
11       MISCELLANEOUS REGISTERS        12       CPU ACCESSIBILITY
50       ASYNC LINES ACCESSIBILITY      51       BPP LINES
52       PRINTER

COMMANDS: nn> SELECT TEST
           *> ADD ALL TESTS TO TEST LIST
SELECT [x:exit]:
```

Figure 94. Sample BUILD TEST LIST Menu

6. After building your list, press **x** to get back to the Off-Line Tests menu.
7. Select **3** to execute your test list. If errors occur, select **4** to examine the error report. Appendix A in the Service Guide of each respective SMP system provides information on test group, error codes, and their interpretation.

Chapter 6. Service Director for RS/6000

Service Director provides the customer with tools to effectively manage hardware errors on the RS/6000. When used properly, it can play an important role in increasing system availability and improving customer satisfaction. This is particularly important in an SMP server environment. The application is supplied at no charge to customers whose machines are under warranty or covered under an IBM Maintenance Agreement.

6.1 Introduction

The Service Director application automatically reports hardware-related problems to IBM Service via a modem. It performs analysis of those problems before reporting them to IBM.

Calls can be placed without customer intervention, or they can be sent as e-mail to customer personnel who decide whether to place a service call. Service Director aids IBM Customer Engineers in problem determination by providing analysis and an event log that contains snapshots of the system error report, a history of service calls, and other valuable information.

6.1.1 Service Director Availability

Service Director is available in the United States, Canada, Austria, Belgium, Denmark, Finland, France, Germany, Israel, Italy, Japan, the Netherlands, Norway, Slovenia, South Africa, Spain, Sweden, Switzerland, and the United Kingdom. Other countries that plan to offer Service Director in the near future are Bulgaria, Hungary, Poland, Russia, and Singapore.

Each country has a phone number that customers' machines use to report problems. The number in the United States and Canada is 1-800-830-1041. Contact local support representatives for the number in other geographies.

6.2 Service Director Overview

You implement Service Director by installing the program, registering machines, setting up configuration files, and customizing the application.

6.2.1 Service Director Components

Service Director for RS/6000 has three major components:

- **Product Support Application (PSA)** The PSA determines what to do about specific errors and captures and passes information required to resolve the problem. The diagnostics code provided by IBM is the default PSA.
- **Analysis Routine** The Analysis routine within Service Director schedules execution of the PSAs. The PSAs may be configured to run at specific intervals. When the Analysis routine runs, it monitors errors or events that are identified by the PSAs.
- **User Interface** The User Interface for Service Director provides a structured view of problem management information, such as the status of recent hardware events logged by Service Director, the history of hardware events, and statistics on problems managed by the application.

6.2.2 Automatic Problem Reporting

Service Director can automatically create problem records on IBM's Problem Management System, RETAIN. These records contain information collected by Service Director. The customer may determine whether to automatically call IBM Service and/or to forward information about an event to a system administrator first. The customer may also set a threshold number of events that must occur before Service Director places a service call. Electronic mail generated by errors is sent from the failing host to the designated people.

The customer may designate one machine as the Service Director gateway for multiple RS/6000s. Errors will be forwarded to the gateway machine, which will then notify IBM.

The customer may also block calls on specific adapters or devices. This might be done, for example, on a third-party peripheral device that is not covered under an IBM Maintenance Agreement. You should contact an IBM service representative to see if a third-party device may be covered under the IBM Maintenance Agreement.

6.2.3 Problem Reporting and Notification

The Notify option in Service Director lets the customer establish a list of people who will receive electronic mail from the application. This option is especially useful for customers who prefer to have a help desk or support group manage hardware service calls.

6.3 Service Director Function

Service Director works on a timed cycle. At preset intervals, it checks errors. Between checks, Service Director sleeps. It wakes up, however, if an error occurs and causes an entry to be placed in the system error log.

The customer may determine the frequency of Service Director's inspection of the error log. The default is once an hour. Checks cannot be made more frequently than that, but they can be scheduled less frequently.

The Service Director Event Log records calls placed to IBM and their status, provided all communication with IBM was completed and the problem record (PMR) was opened successfully. Once the call has been placed, Service Director will go back to sleep and wake up at the predetermined time or when another hardware error occurs.

To prevent sending too many calls, Service Director will only open one call in a 24 hour period for a specific error number on a specific machine. Service Director will track additional instances of the same error, but it will not attempt to place another service call to IBM if one is already open.

When the PMR is received at IBM, Support Center staff will analyze it and, if applicable, tell the Customer Engineer the most likely Field Replaceable Unit (FRU) causing the problem.

Service Director sends electronic mail to the customer telling him/her when the machine will be going off warranty or when the maintenance agreement will expire. Notices are sent 90, 60 and 30 days prior to warranty or maintenance

agreement expiration. This automatic feature is activated when Service Director registers customer machines with IBM.

6.3.1 Errors Reported by Service Director

Service Director incorporates independent program modules called Product Support Applications. Each PSA is responsible for gathering error information and determining if a reportable failure has occurred. When Service Director detects a reportable error, it begins the process of placing a service call.

Service Director can accommodate additional PSAs as they are developed for new products. Here are the PSAs currently provided with Service Director:

- The IBM PSA. It relies on concurrent diagnostics to determine if a reportable failure has occurred. Concurrent diagnostics is limited in the errors that it can detect.
- The IBMELH PSA. It reviews the system error log for problems associated with “SP2” or the “9076” model type. When an event of this type occurs, the PSA checks its internal threshold for the specific error and notifies Service Director if appropriate.

6.3.2 Service Director Limitations

The following types of errors are ones for which Service Director is able to place a service call:

- Any PSA-generated reportable error, providing a Hold flag hasn’t been set to prevent the PSA from reporting the problem. Please refer to 6.5.2.2, “PSA Hold Files” on page 170.
- Any error detected by a third-party PSA for a third-party device or peripheral that interfaces with IBM diagnostics software routines.

The following are types of errors for which Service Director is **not** able to place a service call:

- A machine check (888) that locks up the system and requires a reboot. No service call will be placed unless a valid Service Request Number (SRN) is produced during diagnostics that run at boot time. If the machine cannot reboot, Service Director cannot run and no service call will be placed. However, on an SMP server, SystemGuard can be configured to dial-out and report a boot failure to IBM.
- Errors that diagnostics determines are not reportable failures.
- Errors on a resource for which no valid PSA is available.
- Errors on a resource on which a Hold flag has been set.

6.4 Installation of Service Director

This section outlines the steps necessary to complete a successful installation of Service Director. The default settings and the screens on which you can change them are shown.

6.4.1 Service Director Prerequisites

Service Director for RS/6000 will run on AIX V4.1 or V4.2, which are the supported releases for the SMP servers.

You should have approximately 5 MB of free disk space for the installation and use of Service Director.

Service Director requires a Hayes-compatible modem and an analog phone line. Ideally, the modem would be one that has already been tested to work with SystemGuard, such as the IBM 7851, IBM 7852, or USRobotics Sportster, so that the same modem can be used by both programs. Before Service Director can be configured, the modem must be installed, tested, and working properly.

AIX Diagnostics must be installed on the hard disk and error logging and error analysis must be turned on.

6.4.2 Obtaining Service Director

IBM now preinstalls Service Director for RS/6000 on all diagnostics-capable systems, including SMP servers. However, the code installed at the factory may not be the latest available. Normally, the customer is responsible for obtaining the latest version of Service Director and customizing the application. IBM could provide installation and customization, possibly on a chargeable basis. Please contact you local IBM service representitive for more details on this option.

There are now two ways to obtain Service Director:

- **Internet** Customers who have access to the Internet may use anonymous ftp to get the code from IBM. In the United States, detailed instructions on downloading Service Director are available in document #1715 in 1-800-IBM-4FAX. To transfer Service Director in this manner, follow these instructions:

  1. Log in or su to root.
  2. Enter `cd /tmp`
  3. Enter `ftp aix.boulder.ibm.com` or `ftp 198.17.57.66`
  4. Login as *anonymous* and use your e-mail address as your password.
  5. Enter `bin`
  6. Enter `cd /aix/servdir_client_code`
  7. Enter `get servdir.installp.Z`
  8. Enter `quit`

  After downloading the compressed file, uncompress it with the following command:

  ```
  uncompress servdir.installp.Z
  ```

  The installation file, servdir.installp, is included in the compressed file. Once you have uncompressed it, follow the procedures in 6.4.3, “Installation Procedure” on page 155.

- **AIXTOOLS** IBM Customer Engineers can obtain the code from the AIXTOOLS disk by issuing the following command in VM:

  ```
  TOOLS SENDTO LEXVMIC0 TOOLMGR3 AIXTOOLS GET SD6K PACKAGE *
  ```

The CE should then download the package and follow the instructions in the README file to place the installable image on disk. Then the procedures outlined in 6.4.3, “Installation Procedure” on page 155 can be followed.

In the past, Service Director could be obtained on diskettes by sending a FAX request to IBM. Because so few customers have used this method, the option will be discontinued.

6.4.3 Installation Procedure

Service Director code must be installed on every host that will be reporting errors to IBM. Machines either report their own errors or forward them to another host, called a gateway, which makes the call to IBM. Machines that place their own calls via modem or forward them to another machine via TCP/IP are referred to as Reporting Only Clients. A system that serves as a gateway to IBM for itself and other systems is called a Forwarding Server Client.

If you are installing multiple systems and using a Forwarding Server Client, you should install that machine first and do the registration of your other systems from that host. You can then use Service Director utilities to install the necessary software keys, setup files, and Service Director code on the remote clients from the server.

You can install Service Director either from the command line, using the `installp` command, or from SMIT.

In order to install Service Director from the command line, do the following:

1. Log in as *root* on the host to be registered.
2. Enter `installp -aqXd <path>/servdir.installp all`
3. Go the /usr/lpp/servdir directory and follow the setup instructions found in the README.SD file

In order to install Service Director from SMIT, do the following:

1. Login as *root* on the host to be registered.
2. Enter `smit` or `smitty`
3. Go to the “Software Installation and Maintenance” menus:
   - “Install/Update Software”
   - “Install/Update Selectable Software (Custom Install)”
   - “Install Software Products at Latest Available Level”
4. Enter the full Service Director path and filename for the source, servdir.installp.
5. Select **Do**
6. The “Installp Summary” message will indicate whether installation was successful.

During the installation, various messages will appear on the screen ending with the Installp Summary message. If you get errors, refer to Service Director documentation for assistance in problem determination.

This completes the installation procedure. Your next step will be to register Service Director.

6.5 Registering Service Director

In order for Service Director to execute, the machine on which it is installed must be registered. Registration is accomplished by executing a registration program.

In a networked environment, you may run the registration program on the gateway host and register all of the systems for which it will act as a server.

After Service Director is installed, SMIT will have a registration option on the "Problem Determination/Service Director/Register Machine" screen. The SMIT fast path is `smitty servdir_reg`. Also you can execute the registration program from the command line by entering:

```
/usr/lpp/servdir/servdir.reg
```

The purpose of registration is to generate unique software keys for each host that will be using Service Director. Until a software key has been generated and distributed, Service Director will not execute. Registration menu selections 4 and 5 let you distribute the Service Director software keys and other required files to all participating hosts.

If Service Director came preinstalled on your RS/6000, you have been provided a temporary key that will expire 13 months from the date your software was installed. This temporary key will allow you to use Service Director in a "demo" mode, which will allow you to monitor the system but not place calls to IBM Service. At any time during the warranty period, you can complete the registration process and get a full-function key.

The software key used by Service Director is unique to each system. Before starting the registration process, you should know the customer contact, phone number and address, the four-digit model type, the `uname -m` output, and the last five digits of the serial number of each machine you wish to register.

The registration process includes these steps:

1. Introduction to the registration program
2. Building a list of machines to register
3. Registering the machines
4. Establishing a gateway machine (optional)
5. Distributing the keys and enabling files to all registered systems
6. Distribute client Service Director code, if needed

6.5.1 The Registration Program

After Service Director has been installed, you'll see four screens that spell out the Service Director licensing agreement. On the last screen, you can indicate whether you accept the agreement. You must accept the agreement before you can use the application.

During the registration process, data for each machine will be stored locally in the /usr/lpp/servdir directory in the form:

./SERDIR_REGISTRATION_DATA/ttttsssssssuuuuuuuuuuuu

where tttt is the model type, sssssss is the seven-digit serial number, and uuuuuuuuuuuu is the output of `uname -m`.

If at a later time you want to copy the enabling files yourself, here's where they are located:

- ProbrepAt is stored in /usr/lib/ras.
- callhomehost (if a gateway is configured) goes in /usr/lib/ras.
- msglog.index goes under /usr/lib/servdir.
- customer.dat goes under /usr/lib/servdir.

The files listed above are the ones transferred to the clients during step 4.

6.5.1.1 Build/Update List of Machines for Registration

Following are the Registration Menu screens as you'd see them and some sample selections:

```
SERVICE DIRECTOR REGISTRATION 2.0
---------------------------------

1. Build/Update List of Machines for Registration
2. Register Machines with IBM
3. Build Reporting Topology Files
4. Distribute Registration Files
5. Service Director code distribution

d. Delete/remove Service Director code from Client
p. Print current machine list
r. Review introduction help screens
x. EXIT

Please enter a Step number or a menu selection:
```

Figure 95. Registration Menu

Select **1** to **Build/Update List of Machines for Registration.**

```
BUILD LIST OF MACHINES FOR REGISTRATION                        (Build1)
---------------------------------------

You may register one or more machines with Service Director at a time.
The following selections allow you to CREATE, ADD, UPDATE, or DELETE the
machine list which is used to register Service Director for each
configured host.

1. CREATE/SAVE machine list
2. ADD machine(s) to the current machine list
3. UPDATE machine(s) in the current machine list
4. DELETE a machine from the current machine list
5. ADD machine(s) from an input list file

r. RETURN to parent menu

Please enter menu selection:
```

Figure 96. Service Director Registration - Build Menu

Enter **1**, **CREATE/SAVE machine list**. The following screen is displayed when this option is selected. This screen can be utilized to create, backup, or restore machine lists as well as to save the local Service Director directory structure to a tar file.

```
BUILD LIST OF MACHINES FOR REGISTRATION                          (Build8)
---------------------------------------

NOTE: It is recommended that selection 4 be selected after a new
      machine has been created.

1. Create NEW machine list
2. BACKUP current machine list
3. RESTORE backed up machine list
4. SAVE Service Director image to I/O device

h. HELP
r. RETURN to parent menu
Messages:

Please enter menu selection
```

Figure 97. Registration - Build/Backup Process Screen

Select **1**, **Create a NEW machine list**. The build process validates the machine configuration in order to properly define the Service Director server/client reporting structure. All machines that use Service Director are clients of the application. Hosts that generate their own calls for service are referred to as Reporting Only Clients. Those that place their own calls or forward calls to IBM from networked machines are called Forwarding Server Clients.

```
BUILD LIST OF MACHINES FOR REGISTRATION                       (Build2)
---------------------------------------

1. uname -m:                                        00009895A300
2. Type:                                            7013
3. Serial:                                          0009895
4. Hostname:                                        itsorus
5. Is this a Reporting server? (y/n):               y
6. Contact:                                         Terry
7. Voice phone:                                     512-838-1234
8. Data phone:
9. Company name:                                    IBM
10. Street:                                         11400 Burnet Rd.
11. City:                                           Austin
12. State:                                          TX
13. Zip:                                            78758
14. Country:                                        UNITED STATES

i. INSERT this machine's hostname and uname number
o. OVERRIDE default configuration data for this machine
s. SAVE this machine's data, move to next machine
c. CANCEL this entry, do not save it, return to calling menu

Please enter menu selection:
```

Figure 98. Registration - Build Screen

Enter data for this host. Select **o** to **OVERRIDE default configuration data for this machine**. You'll then see this screen:

```
BUILD LIST OF MACHINES FOR REGISTRATION                    (Build3)

For Machine with Type: 7013 Serial: 0009895

1. Host reports problems? (N/Y)                Y
2. Host forwards problems (N/Y)                Y
3. Comm method (RPC or TTY)                    TTY
4. File containing RPC target:                 /usr/lib/ras/callhomehost
5. TTY device                                  tty1
6. Parent TTY adapter:
7. Port number of parent adapter:
8. AIXServ login id:
9. AIXServ password:
10. AIXServ customer number:
11. Forwarding metric:                         5
12. Notification mail address:                 joe@systemname

d. SCROLL DOWN for other values
s. SAVE these values (may replace default configuration, too)
c. CANCEL, discard any changed values
h. SHOW explanations for data fields (uses pg command)

Please enter menu selection:
```

Figure 99. Service Director Registration - Build Screen

Fill in data for the specific host, then enter **d**, **SCROLL DOWN for other values**, to enter additional data.

```
BUILD LIST OF MACHINES FOR REGISTRATION                    (Build4)

For Machine with Type: 7013 Serial: 0009895

13. File containing RFC clients:           /usr/lib/ras/probrephosts
14. Modem file name:
15. Remote latency delay:                  0
16. Remote timeout                         300
17. Retry count:                           2
18. Response timeout:                      600
19. TTY retry timeout:                     86400
20. BAUD rate:                             9600
21. Dial-out phone number:                 1-800-830-1041
22. Phone number to call back to:
23. Local operator voice phone:
24. Hub phone:
25: Duplicate problem timeout:
26. Dial Tone (P)ulse (T)one:              T

u. SCROLL up, for more values
s. SAVE these values (may replace default configuration, too)
c. CANCEL, disregard any changed values
h. SHOW explanations for data fields (uses pg command)

Please enter menu selection:
```

Figure 100. Service Director Registration - Build Screen

Select **s**, **SAVE these values** from this and the previous screen.

```
You may use the configuration data just entered as the default
values, meaning future CPU's added to the registration machine list
will use this configuration. The default configuration does not change
existing CPU data already in the registration machine list.

If you answer N (no) at this point, the data just entered will be
used only for the machine that is currently being added or
updated in the machine list.

Do you wish to overwrite the default configuration? (Y/N) ?
```

Figure 101. Registration - Build Screen

If you wish to overwrite the default configuration with the data you have just entered, select **y**.

```
BUILD LIST OF MACHINES FOR REGISTRATION                    (Build2)
---------------------------------------

1. uname -m:                            00009895A300
2. Type:                                7013
3. Serial                               0009895
4. Hostname:                            itsorus
5. Is this a Reporting Server? (Y/N)?   Y
6. Contact:                             Terry
7. Voice Phone:                         512-838-1234
8. Data phone:
9. Company Name:                        IBM
10. Street:                             11400 Burnet Rd.
11. City:                               Austin
12. State:                              TX
13. Zip:                                78758
14. Country:                            UNITED STATES

i. INSERT this machine's hostname and uname number
o. OVERRIDE default configuration data for this machine
s. SAVE this machine's data, move to the next machine
c. CANCEL this entry, do not save it, and return to calling menu

Please enter menu selection:
```

Figure 102. Registration - Build Screen

Enter **s**, **SAVE this machine's data, move to the next machine** until all of the machines have been entered. When the list is complete, and a new blank screen appears. Enter **c**, **CANCEL this entry, do not save it, and return to the calling menu**.

```
BUILD LIST OF MACHINES FOR REGISTRATION                          (Build1)

You may register one or more machines with Service Director at a time.
The following selections allow you to CREATE, ADD, or UPDATE the
machine list which is used to register Service Director for each configured
host.

1. CREATE a new machine list
2. ADD machine(s) to the current machine list
3. UPDATE machine(s) in the current machine list
4. DELETE a machine from the current machine list

r. RETURN to parent menu

Please enter menu selection:
```

Figure 103. Registration - Build Screen

This screen will allow you to return to the Service Director Registration Menu and continue the registration program. If you are through adding machines, select **r, RETURN to the parent menu**.

6.5.1.2 Register Machines with IBM

```
SERVICE DIRECTOR REGISTRATION 2.0
---------------------------------

1. Build/Update List of Machines for Registration
2. Register Machines with IBM
3. Build Reporting Topology Files
4. Distribute Registration Files
5. Service Director code distribution

d. Delete/remove Service Director code from Client
p. Print current machine list
r. Review introduction help screens

Please enter a Step number or a menu selection
```

Figure 104. Registration - Main Menu

From this screen you are ready to start registration with IBM. In this example, select **2**, **Register Machines with IBM**.

```
REGISTER MACHINES WITH IBM:                                    (Register1)
--------------------------

At this point, you should have finished creating the Service Director
registration files. If this is not correct, please choose Return and
select the Build/Update selection to complete the list.

Otherwise, make sure your connection to IBM is
operational, then choose the appropriate Register action.

There may be a fairly long delay while sending and receiving data.
Please do not interrupt this program unless absolutely necessary.

e. REGISTER ONE machine at a time
a. REGISTER ALL machines in the current machine list
t. PLACE PROBLEM TEST CALL for registered Reporting Server
r. RETURN to parent menu

Please enter menu selection:
```

Figure 105. Registration - Register Menu

In this example, assume you are only registering one machine. In that case, you would enter **e**, **REGISTER ONE machine at a time.** For multiple machines, you'd enter **a**, **REGISTER ALL machines in the current machine list**.

```
REGISTER MACHINES WITH IBM                                  (Register1a)
--------------------------

You asked to register machines singularly. Please give the number of the entry
you wish to register (only give one number at a time.)

Rec                   Processor         R=Full key K=Key sent
#    Hostname         ID Number         T=Temp key C=Client code sent

1. <hostname>

r. RETURN to parent menu

Please enter menu selection:
```

Figure 106. Registration Menu

The registration screen shown above will display a series of codes next to each host if the process associated with the key has been executed on that host. For example, if the R K C codes appear beside a hostname, the host has been registered, and the access key and Service Director client code have been distributed. The T flag denotes a 30-day demo key.

The registration process may be lengthy, depending on the speed of the customer's modem and the number of machines to be registered. IBM validates the warranty and maintenance agreement status of each host. The enabling keys will be generated and now need to be distributed to participating machines.

To prepare to distribute the key(s), select **r**, **RETURN to parent menu.**

```
SERVICE DIRECTOR REGISTRATION 2.0
---------------------------------

1. Build/Update List of Machines for Registration
2. Register Machines with IBM
3. Build Reporting Topology Files
4. Distribute Registration Files
5. Service Director code distribution

d. Delete/remove Service Director code from Client
p. Print current machine list
r. Review introduction help screens
x. EXIT

Please enter a Step number or a menu selection:
```

Figure 107. Registration - Main Menu

6.5.1.3 Build Reporting Topology Files

Now that the list of machines has been registered, the next step is to develop the reporting topology for this customer's environment. Specifically, this is the path that the individual machines will take to report their hardware problems and the gateway machine that will forward them to IBM.

Select **3** from this menu to **Build Topology Files**.

```
BUILD REPORTING TOPOLOGY                        (Gateway1)
-----------------------

When using RPC connections between machines, the call forwarding
feature needs to know the machine name of the partner node(s).
This step creates the required name list files and stores them
locally for eventual distribution to the correct machines.

It also verifies that the Remote timeout values on the partner
nodes are set to (3 X Remote timeout X Retry Count)
values found on the Reporting Server host.
Adjustment of this Server to Remote timeout ratio may be required
depending upon your network system load.

Select Continue to build the files, Return to return to the previous
menu.

a. CONTINUE
r. RETURN to parent menu

Please enter menu selection:
```

Figure 108. Registration - Build Report Topology

Now that all pertinent information has been collected for all machines, the gateway system needs to build the files necessary for reporting problems. Once **CONTINUE** is selected, the files will be generated that will allow Remote Procedure Call (RPC) connectivity. Service Director uses this mechanism to manage networked systems.

```
This Reporting Server (itsorum.austin.ibm.com) can forward
problem reports from Client host calls to "callhome" process.
If you would like this function, the /usr/etc/portmap will
be turned on, an entry will be made in the /etc/inittab file and the
Callhome forwarder will be started. Would you like this function? (y/n)
y
The following is the current Service Director crontab entry:

0 * * * * /usr/lpp/servdir/servdir.timer 1>/dev/null 2>/dev/null

Do you wish to add or change the current Service Director crontab entry?
Enter (y)es to change entry:
n
THE BUILD REPORTING TOPOLOGY FILES STEP HAS BEEN COMPLETED!

The "RPC Target" file that will reside on the clients and is usually
known as /usr/lib/ras/callhomehost contains the following:
<IP Address>

The "RPC Client" file that will reside on the Server and is usually
known as /usr/lib/ras/probrephost contains the following:

<Hostname>, <Hostname>, ...

Verify all Client hostnames are listed in the "RPC Client" file and
the Server IP address and/or hostname are listed in the "RPC Target"
file.
If not, verify that the "Comm method" for the missing host is set to "RPC"
and/or your local server has hostname resolution capabilities.

Press Any key to return to the previous menu.
```

Figure 109. Registration - Build Report Topology

On the above screen, note that the first question relates to using and starting the Callhome forwarder in connection with a gateway host. The second question, relating to the crontab entry, gives you the opportunity to make a local change that can be propagated easily to all of the client machines in the next step, distributing registration files.

6.5.1.4 Distribute Registration Files

At this point, you will be returned to the Service Director Registration menu. The next step is to distribute files to the individual machines and to ensure that connectivity exists via the Service Director reporting and enabling process. From the Registration menu, select **4**, **Distribute Registration Files.**

```
DISTRIBUTE FILES                              (Distrib1)
---------------

At this point, you should have finished registering your machines with IBM
and creating the reporting topology files. If this is not correct, please
choose Return to return to the appropriate step.

Otherwise, make sure you have TCP/IP connectivity to the remote
machine(s) that need the Service Director enabling files sent to them,
then choose the appropriate Distribute action.

There may be some delay while sending and receiving data.
Please do not interrupt this program unless absolutely necessary.

e. DISTRIBUTE files to ONE machine at a time.
a. DISTRIBUTE files to ALL machines in the current machine list.
r. RETURN to parent menu.

Please enter menu selection:
```

Figure 110. Registration - Distribute Files

For purposes of illustration, select **e** to **DISTRIBUTE files to ONE machine at a time.**

```
DISTRIBUTE FILES                             (Distribute2)
---------------

You asked to distribute files to machines singularly.
Please give the number of the entry you wish to distribute to
(only give one number at a time.)

Rec                      Processor        R=Registered K=Key sent
 #   Hostname            ID Number           C=Client code sent
1 - <Hostname>

r. RETURN to parent menu

Please enter menu selection:

Is the root user-ID different than "root"? Enter (y) to change
-------------------------------------------------------------
DISTRIBUTION STATUS                           Key        (Distribute6)
------------------

Last host=<Hostname> <uname -m #>

DISTRIBUTION OF FILES, ALL TRANSFERRED OK TO CPU)s):

Press Enter to continue
```

Figure 111. Registration - Distribute Menu

After successful completion, enter **r** to **RETURN to parent menu**.

6.5.1.5 Service Director Code Distribution

In addition to distributing setup files, Service Director also can distribute its client code. This saves you having to manually install each host. You have the choice of distributing code to a client that forwards problems to a gateway machine or to a server that can generate its own calls and those sent to it by other hosts. Code distribution is separate from the distribution of key files.

From the Service Director Registration Menu, select option **5**, **Service Director code distribution**.

```
SERVICE DIRECTOR CODE DISTRIBUTION                          (Distribute3)
---------------------------------

This selection will distribute the Service Director execution code,
"Notify User List", crontab entry, and other setup items on this host
to the client machine(s) in the current machine list. Once distributed,
further installation and basic setup procedures will not be required on
the client. In order to insure the clients are properly set up you must
complete the following steps prior to making this selection:

1. Completed steps 1, 2, and 3 of the Main Menu.
2. Distribute the registration files to this local host (Main Menu 4).
3. Exit this registration program and "Invoke Service Director Session"
   Setup the "Notify User List", crontab, and any other customized
   features you require of Service Director. You MUST set up the "Notify User
   List" so remote clients will be able to notify the system administrator
   when a problem is reported. EXIT Service Director session.
4. Make sure you have TCP/IP connectivity to the remote machine(s).
5. Re-invoke the registration program and continue with the distribution options.

Please enter "r" to return, or enter "c" to continue.
```

Figure 112. Registration - Service Director Code Distribution

Attention!

Pay careful attention to the list of things that must be done prior to distributing code. Note that the Notify User List should be set up **before** code is distributed. Please refer to 6.5.2.1, “Notify User List” on page 169. The crontab entry must also be correct. Once you have verified that you have complied with steps 1 through 5, you can press **c** to continue.

When you continue with code distribution, the following screen will appear.

```
SERVICE DIRECTOR CODE DISTRIBUTION                          (Distribute4)
---------------------------------

For proper distribution, the previous 5 steps should have been completed.

If you have not done so please invoke "SETUP Service Director Distribution
Files", selection "s". This selection will allow you to determine which
type of client you wish to distribute to, a "Reporting Only Client" or a
"Forwarding Server Client."

     - "Reporting Only Client" Service Director session code only
     - "Forwarding Server Client" Service Director and Registration session
       code that can also forward calls from his clients

s. SETUP Service Director Distribution Files
r. RETURN to parent menu
Messages:

Please enter the menu selection:
```

Figure 113. Registration - Service Director Code Distribution

If this is the first time you've displayed this screen, or if you wish to change which type of Service Director host you want to create, select option **s**, **SETUP Service Director Distribution Files**. If **s** is selected, the following screen is displayed.

```
SERVICE DIRECTOR CODE DISTRIBUTION                          (Distribute5)
---------------------------------

When the option to "BUILD Client" is selected, the associated files
are selected and a remote exec is built to set up the remote client(s).
This remote exec will execute after the Service Director file transfer
has completed successfully. Service Director should be active and if
the registration files have been distributed, fully executable.

Files other than those defined for the BUILD Client Code may be sent to
/usr/lpp/servdir director of the machines in the current machine list
by doing either of the following:

 1. Manually copy files into SERVDIR_UPDATE subdirectory
 2. By entering an ASCII copy list file name at menu prompt

c. BUILD Reporting Only Client
s. BUILD Forwarding Server Client
r. RETURN to parent menu
Messages: Please enter the full path for the copy file list
```

Figure 114. Registration - Service Director Code Distribution

You can customize file distribution by entering your own file name that contains a list of files to distribute. Otherwise, select which type of client you want to build and press **Enter**. This will build a tar format file containing the appropriate code, plus the notify list and crontab file you've created. A message should appear on the screen saying:

Client tar file built, please return and select distribution option

When you see this message, select **r, RETURN to parent menu**, to actually distribute the code. The following screen is displayed.

```
SERVICE DIRECTOR CODE DISTRIBUTION                          (Distribute4)
---------------------------------

For proper distribution the previous 5 steps should have been completed.

If you have not done so please invoke "SETUP Service Director Distribution
files" selection "s". This selection will allow you to determine which
type of client you wish to distribute to, a "Reporting Only Client" or a
"Forwarding Server Client."

   - "Reporting Only Client" Service Director session code only
   - "Forwarding Server Client" Service Director and Registration session code
      that can also forward calls from his clients


s. SETUP Service Director Distribution files
e. DISTRIBUTE files to ONE machine at a time
a. DISTRIBUTE files to ALL machines in the current machine list
r. RETURN to parent menu
Messages:


Please enter the menu selection:
```

Figure 115. Registration - Service Director Code Distribution

Select the type of distribution you want and press **Enter**. In this example, press **e DISTRIBUTE files to ONE machine at a time**. Upon completion, the following message should be displayed:

```
"DISTRIBUTION OF FILES, ALL TRANSFERRED OK TO CPU(s):
Press Enter to continue"
```

6.5.1.6 Delete/Remove Service Director Code from Client

Service Director provides the ability to remove or deinstall the application code from a client. On the main menu, there's an option to **Remove Service Director Code from Client**. If that option is selected, the following screen appears.

```
REMOVE SERVICE DIRECTOR CODE                              (Remove1)
----------------------------

You have selected the option to remove Service Director Code from
remote Clients. This selection will use remote exec function to
execute "servdir.remove" executable to remove all Service Director
code and files from the remote machine.

e. REMOVE Code from ONE machine at a time
a. REMOVE Code from ALL machines in the current machine list
r. RETURN to parent menu

Please enter menu selection:
```

Figure 116. Registration - Remove Service Director code from Client

Select **r**, **RETURN to parent menu**.

6.5.1.7 Print Current Machine List

Service Director provides the customer with the option of printing the current machine list. This option can be utilized at any time from the Registration menu screen. When you select that option, a Registration Print Screen appears that asks you whether you want to print to a device other than the default system printer. If so, you can enter the queue name and press **Enter**.

Selecting **x** for **EXIT** from the main menu causes Service Director to leave the Registration menu. Once the various machines have been registered and the keys distributed, exit the program and invoke Service Director from the SMIT interface.

6.5.2 Service Director Setup Verification

After registration, you should enter Setup from the main Service Director menu to check the customer profile and verify the User Notify List. You'll recall that notification was one of the things you set up before distributing files; in the examples that follow, we'll use the Setup functions to make sure the information is correct and was distributed to all machines. You should verify on **each** host, not just the one serving as a gateway.

Other options besides Setup appear on the Service Director main menu. They allow further customizing, but they are not required for basic Service Director function. For additional information, please refer to /usr/lpp/servdir/README.SD or to the *Service Director User's Guide*.

```
SERVICE DIRECTOR - MAIN MENU

   LOCAL EVENTS
   REMOTE EVENTS
   UTILITIES
   SETUP
   LOCK FILES
   README FIRST




F1=Help   F2=Refresh   F3=Cancel/End
F10=Exit  ENTER=DO
```

Figure 117. Service Director Main Menu

6.5.2.1 Notify User List

Select **SETUP.** Then, on the SETUP Menu, select **NOTIFY USER LIST**. The following screen will appear.

```
                  SERVICE DIRECTOR SETUP SCREEN

<name>@<address>
EVENTS-Y CALLS-Y PROBLEMS-Y VERSION-Y CALL HELD-Y

<name>@<address>
EVENTS-Y CALLS-Y PROBLEMS-Y VERSION-Y CALL HELD-Y

<name>@<address>
EVENTS-Y CALLS-Y PROBLEMS-Y VERSION-Y CALL HELD-Y

F1=Help   F2=Refresh   F3=Cancel/End
F10=Exit  ENTER=DO
```

Figure 118. Notify User Screen

On this screen, you can provide the e-mail address of as many as three people who will be notified under the circumstances you specify. You can customize the notification for each person. Here are the definitions of the keywords appearing here:

EVENTS Notify whenever an event is entered in the event log.

CALLS Notify whenever a CALL is submitted to the IBM Problem Management Reporting system.

PROBLEMS Notify whenever a Service Director technical problem occurs, such as running out of memory.

VERSION Notify whenever a new version of Service Director is available.

CALL HELD Notify whenever a Call problem was created, but is held.

6.5.2.2 PSA Hold Files

One other option on the Setup menu that should be pointed out is the PSA HOLD Files selection. If there are non-supported resources on the host and you want Service Director to ignore them, you should set up a hold file to prevent an unwanted call from being placed to IBM Service.

Here's how to create hold files:

1. From the Setup Menu, select **PSA HOLD FILES**.
2. Select the Product Support Application (PSA) that would be reporting errors from the resource in question. The default PSA is IBM, which uses the system error log and diagnostics to verify errors logged. If the nonsupported resource reports to the error log, select **IBM**.
3. You will see a **SETUP PSA HOLD FILES** screen. Press the **Enter** key to move the cursor over the field that is titled **IGNORE RESOURCE.**
4. Once there, enter the **exact name of the resource to be ignored**, then press the **Enter** key to go to the next line.
5. After making all entries, press **F3** to get out and enter **(Y)es** to save the entries.

Attention!

If there are OEM components that are not supported under the customer's IBM Maintenance Agreement, Service Director should be instructed to ignore them. If a call is placed to IBM on an unsupported component, the customer may be liable for a per-call service charge. You can contact an IBM service representative to determine if the OEM component can be added to your IBM Maintenance Agreement.

6.6 Using Service Director

This section is not intended to be a comprehensive guide to using Service Director. For that, you should refer to the *Service Director User's Guide*. In this section, you'll be introduced to the types of functions that are available in Service Director.

You can bring up the Service Director Main Menu by going into SMIT Problem Determination/Service Director/Invoke Service Director User Interface. The screen looks like this:

```
                    SERVICE DIRECTOR - MAIN MENU



              LOCAL EVENTS
              REMOTE EVENTS
              UTILITIES
              SETUP
              LOCK FILES
              README FIRST




F1=Help     F2=Refresh    F3=Cancel/End
F10=Exit    ENTER=DO
```

Figure 119. Service Director - Main Menu

The LOCAL EVENTS and REMOTE EVENTS options are self-explanatory. The UTILITIES Menu provides tools for testing Service Director communications and user notification lists, as well as changing the crontab time and displaying the current system setup. We've already visited the SETUP Menu, which contains the Notify User List, Service Director PSA List and PSA Hold Files. There are some additional options available on that screen that we'll discuss shortly. The LOCK FILES Menu allows you to create, verify and remove Service Director lock files. These files prevent system errors from triggering more than one Service Director "control engine" at a time. README FIRST contains the Service Director User's Guide in ASCII format.

6.6.1 Service Director Local Events Menu

The Service Director Events Menu, reached by selecting **LOCAL EVENTS** on the Main Menu, lets you examine all events or just those recorded by a PSA. Remember that a PSA will not log an event that does not require a service call, and all error log events are viewable through this menu. By default, only the IBM PSA appears on this screen, but additional PSAs will appear here as they are added.

The Events Menu will display a one-line summary of events, including:

- A record number for events generated by a PSA
- A date and time stamp
- The resource name
- The type of error, either Temporary, Permanent, Software or Other
- A Failing Symptom Code (FSC) created by a PSA
- A PMR number, if a service call was opened
- The current status of a service call
- A brief description of the error

Function keys allow you to print the report, look at detailed data from a specific event, or obtain the call data from the transmission to IBM Service. You can also delete an event from the log.

6.6.2 Service Director Remote Events Menu

Select **REMOTE EVENTS** to review the log on a remote host or the log file of a Forwarding Server Client host. AIX rcp and rsh commands are used to access remote files. Because of security requirements, root access permissions must be set up before using these options.

Two options on the Remote Events Menu are REMOTE EVENTS LOG and SERVER/FORWARDER LOG. To view the events on a remote host, select the first option. You will then be prompted for the hostname, user ID and path associated with the remote host and event log you wish to access. There's an additional entry called RCP PATHNAME that defines where the rcp and rsh commands are located on the remote host. By default, the user ID is root, and the default path is /usr/lpp/servdir. The event log filename is not specified in this entry. The default rcp and rsh command path is /usr/bin.

Once you have accessed the Remote Events Log, it will look like the local version we've already discussed.

If you select Server/Forwarder Log, you'll be prompted for the hostname, user ID, and path information of the remote host, just as when you were accessing a Remote Events Log on a remote, nonforwarding host. The same defaults apply. Although this screen is similar to the Remote Events Log, you must supply the complete path to the forwarder log and the filename. There is no default forwarder log. The forwarder log name was defined when the forwarder or callhome process was started. Please refer to Client/Server Call Forwarding in the *Service Director User's Guide* for more information.

When dealing with remote logs, the F5 function key initiates a file transfer, and the requested log is placed in the /tmp directory and displayed.

The output of a Server/Forwarder Log contains the following column headings:

MSG-TYPE Indicates the type of transmission, such as CALL and TEST. DELETE indicates the record is marked for deletion.

TRANSMISSION-DATE Date and time stamp of when the transmission was logged.

CLIENT-REC A unique record number received from a client call.

ERROR The Failing Symptom Code generated by the PSA.

PMR The problem management record in RETAIN.

TYPE The CPU model type on which the error occurred.

SERIAL The CPU serial number of the machine on which the error occurred.

6.6.3 Service Director Utility Menu

The options on this menu are generally used to manage or test the Service Director application. Communication paths to specified customer e-mail accounts and to IBM can be tested. The AIX crontab time and current system configuration can be viewed or changed from options selected on the Utility Menu.

6.6.4 Service Director Setup Menu

The following submenus appear under Setup.

CUSTOMER PROFILE A customer profile should be completed before executing Service Director. This contains the customer contact information.

NOTIFY USER LIST This selection allows you to send e-mail to up to three users when selected types of events occur.

CONFIGURATION This refers to the configuration of Service Director and includes such things as maximum file sizes and working paths.

SD PSA LIST This is the list of Service Director PSAs.

COMMUNICATIONS This allows you to change the ProbrepAT configuration file used when a gateway host has been selected. This submenu also deals with the communications protocols being used, RPC, or TTY.

PSA HOLD FILES This allows you to customize a defined PSA to exclude particular resources or set thresholds for error reporting.

6.6.5 Service Director Lock Files Menu

The options on this screen include checking for lock files, creating them, removing them and restarting Service Director.

If a lock exists, Service Director will not execute. Locks are created to prevent creation of multiple Service Director control processes and to synchronize access to application resources.

If you want to temporarily stop Service Director from executing, you can create a lock file. Once the lock file has been removed, Service Director will resume normal operation. It will also pick up any errors that have occurred since the last time it was executed. In order to prevent reporting invalid errors that occurred after a lock was manually created, you should reset Service Director prior to removing the lock. This can be done by restarting the application. The Service Director time stamps will be reset to the current time. The time stamps are used by Service Director and the PSAs to determine the last time an error occurred. By resetting the time stamps, all errors occurring before the current time will be ignored.

6.6.6 Service Director README First

This option allows you to display the *Service Director User's Guide* as a flat ASCII file. The file is /usr/lpp/servdir/servdir.UG.PS.

Chapter 7. Cluster Power Controller

The Cluster Power Controller (CPC) is a device that allows connection of several CPUs (possibly within a rack but not required) and peripheral drawers or stand-alone units that support the Power Control Interface (PCI). It has a feature code 6175.

It allows a single console to be used for several CPUs. The console can be an ASCII terminal or a PC running DOS. Provided with the CPC control program is a DOS-based terminal emulator which allows you to use a desktop or a laptop PC as a console. A CPC enhances the SMP power control architecture by eliminating multiple IBM 3151 terminals and telephone line connections.

It is possible to daisy-chain several CPC units. The CPC offers power control and connectivity to the following RS/6000 models and peripherals:

- 7012-G30, G40
- 7013-J30, J40
- 7015-R10, R20, R21, R24, R30, R40, 950, 970, 990
- 7013-570, 57F, 580, 58H, 590, 59H, 591 (Note)
- Disk units: 9333, 9334, 7133, 7134, 7135

Note: The power-on feature is not available on the 5XX deskside models. If a 5XX system has been shut down, it must be physically powered back on using the power switch on the front panel.

7.1 CPC Features

The Cluster Power Controller (CPC) provides the following features and functions:

1. Power Control
   - Power-on/off rack-mounted UP/SMP systems and peripherals
   - Power-on/off rack-mounted shared peripherals
   - Power-on/off deskside UP/SMP systems and peripherals
2. Single Console
   - Connect a single console to multiple rack-mounted UP/SMP systems
   - Connect a single console to multiple deskside UP/SMP systems
   - Connect a single console to multiple daisy-chained CPCs
3. Single Modem Connection
   - Share a single modem between multiple rack-mounted UP/SMP system units that are connected to one or more CPCs
   - Share a single modem between multiple deskside UP/SMP systems that are connected to one or more CPCs
4. Update CPC Microcode
   - Update the CPC microcode from an AIX system
   - Update the CPC microcode from a PC system

5. Scheduled and remote power control
6. Mirroring between the TTY and the modem ports
7. Support for the SystemGuard dial-out feature using the single modem connection on the CPC

7.1.1 CPC Connectors

The following figure shows the connectors and features that are available on the front panel of the Cluster Power Controller.

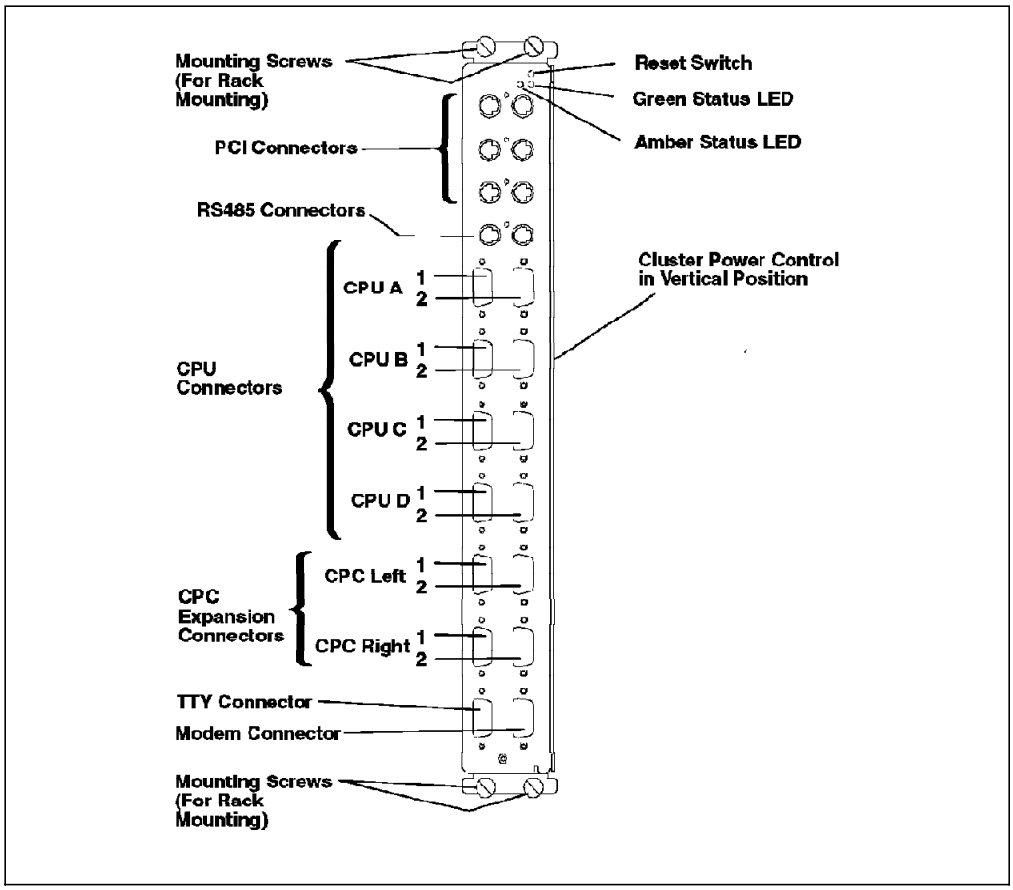


Figure 120. CPC Front Panel

- *Reset Switch*:

  This is a recessed reset button that can be used to perform a hard reboot of the CPC. This is useful for the rack-mounted CPC as the power cord is not accessible after installation in the rack.

  Note: The power status of all the attached system units and disk drawers are not affected when recycling power to the CPC. The system units and disk drawers will not be powered off.

- *Amber LED*:

  The amber LED is used to indicate an error condition.

- *Green LED*:

  The green LED is used to indicate the CPC is ready.

- *Disk Drive Connectors (PCI)*:

The six PCI connectors are used to control the power for the rack-mounted uniprocessor system units (R10, R20 and R24) and/or the disk subsystems. The ports are marked 4-1, 4-2, 4-3, 4-4, 4-5, and 4-6.

- *RS485 Connectors*:

  These connectors are reserved for future use. The ports are marked 6-1 and 6-2.

- *CPU Connectors*:

  These connectors are used to connect the S1 port of all the attached system units to a single TTY device or console. They are also used to connect the S2 port of all the attached system units to the modem port of the CPC.

- *CPC Expansion Connectors*:

  These connectors are used for connecting multiple CPCs in a daisy-chain fashion. Please refer to *Figure 144 on page 200* for the correct cabling diagram.

- *TTY Connector*:

  An ASCII terminal is attached to this port and can be switched to the S1 port of each system unit connected to the CPU ports (A-1, B-1, C-1, D-1) on the CPC. For the SMP systems, this device becomes the BUMP console.

- *Modem Connector*:

  A modem (or another ASCII terminal) can be attached to this port and can be switched to the S2 port of each system unit connected to the CPU ports (A-2, B-2, C-2, D-2) on the CPC. For SMP systems, this port is used as the service console.

Please refer to the *7015 Model R00 Rack Installation and Service Guide,* pages 1-8 or the *Cluster Power Control Operator and Service Guide*, pages 1-2 for more information.

7.1.2 CPC Port Connections

The following figure is a conceptual diagram of a typical CPC-to-CPU connection. Here, we have two G40 servers and one J40 server connected to one BUMP console and to one remote console through the CPC.

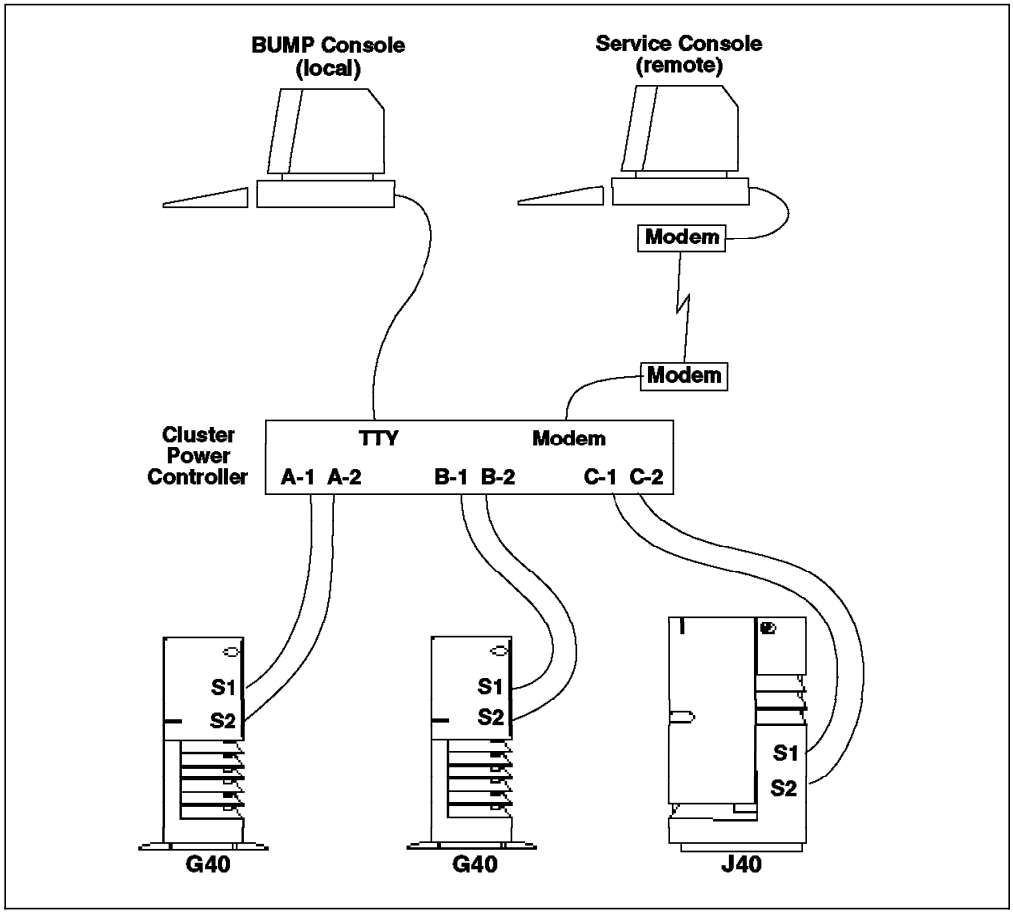


Figure 121. Typical CPC-to-CPU Connection

S1 and S2 of the first G40 are connected to CPU-A Port 1 and Port 2, respectively. S1 and S2 of the second G40 are connected to CPU-B Port 1 and Port 2, respectively. S1 and S2 of the J40 are connected to CPU-C Port 1 and Port 2. The BUMP console user can connect to any of the S1 ports on the three SMP systems. The service console user can connect to any of the S2 ports on the three systems (assuming that the CPC modem port has been enabled).

This example only shows SMP systems attached to the CPC. In reality, a customer may want a combination of systems and devices attached to the CPC.

The following table shows the possible connections between the CPC ports and the various devices.

Qty	Type	CPC A1	CPC A2	CPC B1	CPC B2	CPC C1	CPC C2	CPC D1	CPC D2	4-1	4-2	4-3	4-4	4-5	4-6
1	SMP	K	K	A	A	A	A	A	A	A	A	A	A	A	A
2	SMP	K	K	K	K	A	A	A	A	A	A	A	A	A	A
3	SMP	K	K	K	K	K	K	A	A	A	A	A	A	A	A
4	SMP	K	K	K	K	K	K	K	K	A	A	A	A	A	A
1	UP	K	K	A	A	A	A	A	A	B	A	A	A	A	A
2	UP	K	K	K	K	A	A	A	A	B	B	A	A	A	A
3	UP	K	K	K	K	K	K	A	A	B	B	B	A	A	A
4	UP	K	K	K	K	K	K	K	K	B	B	B	B	A	A
6	933x	A	A	A	A	A	A	A	A	B	B	B	B	B	B
6	713x	A	A	A	A	A	A	A	A	B	B	B	B	B	B

Note:

A	Available or open connection
B	Power control for UP (excluding 5XX) or Disk Subsystems
K	Serial port cables
SMP	G30, G40, J30, J40, R30 or R40 SMP server units
UP	570, 580, 58H, 59H, R10, R20, R21, R24 (UP) system units
933x	9333, 9334 disk subsystems
713x	7133, 7134, 7135 disk subsystems

Table 6. CPC Port Connections

The maximum number of devices that can be attached to a single CPC using a combination of device types is:

- 4 x SMP + 6 x 933x (or 713x)
- 2 x UP + 2 x SMP + 4 x 933x (or 713x)
- 4 x UP + 2 x 933x (or 713x)

Other combinations are possible, but you get an idea of the mixture. 713x refers to 7133, 7134 and 7135.

7.1.3 CPC Cables

The following diagram illustrates the cables required to connect a CPC to multiple CPUs (G40, J40 or 5XX) and a single disk unit.

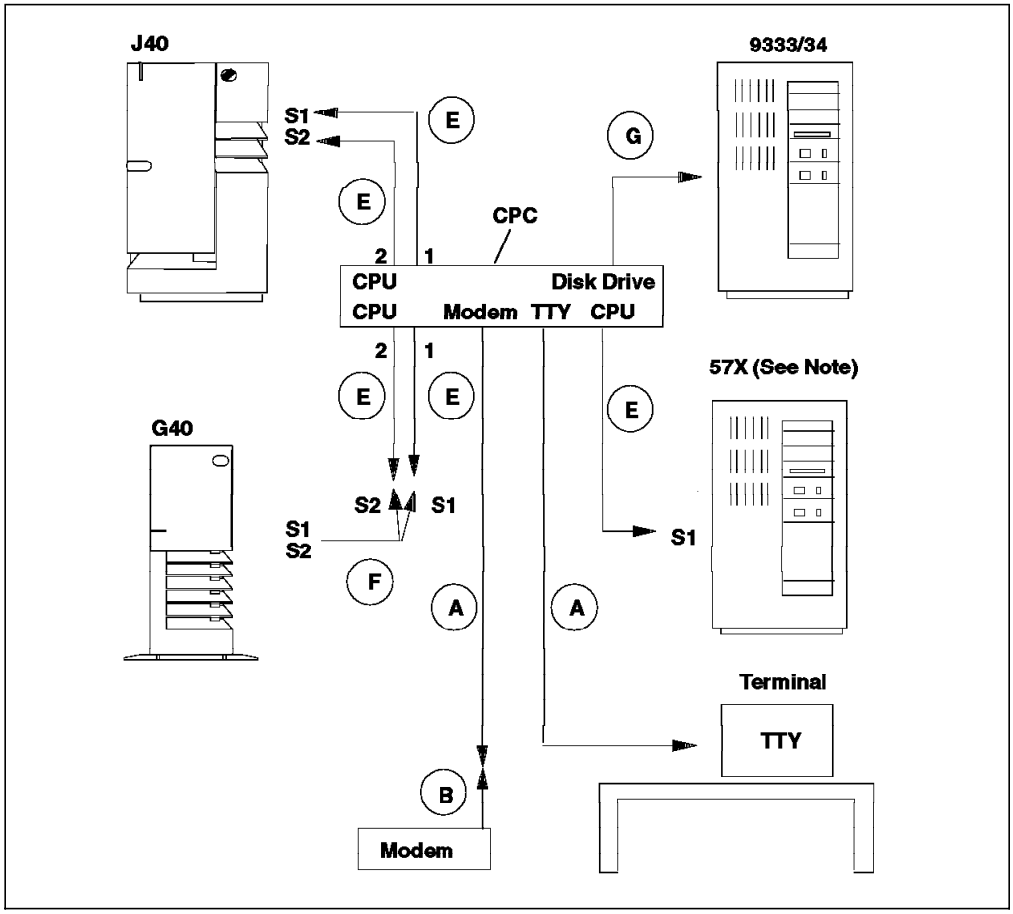


Figure 122. Multiple CPUs connected to a single CPC

The following table gives a description of the cables that can be used to connect the CPC to the various devices.

Cable	IBM P/N	Description
A	6298963, FC 6176	CPC(9F) - TTY(25M) Null modem cable, 10 ft.
B (Note 4)	58F2861	Null modem adaptor or terminal/printer interposer (25F) - (25M)
C (Note 1)	6450242 (kit)	R40(9F) - CPC(25M) AT serial adaptor connector cable, 10 in
D	11H7336, FC 6177	CPC(9F) - CPC(9F) Null modem CPC-CPC interconnect cable, 25 ft.
E	11H7337, FC 6178	CPC(9F) - System(25F) Null modem serial port cable, 10 ft.
F (Note 2)	11H3834	S1/S2 G40(25F) - S1(25M) and S2(25M) 1 to 2 Y-cable, 1 ft.
G (Note 3)	42F6839	CPC(4P) - 933x/713x(4P) Power control interface cable, 10 ft.
K	12H1605, FC 6180	CPC(4P) - R10, R20, R24 (5P) Power control interface cable, 10 ft.

Note:
1. Supplied with R40 to convert 9F SIB ports to 25M
2. Supplied with G40 to convert S1/S2 25M SIB to 2 x 25M
3. Supplied with peripherals. 713x refers to 7133, 7134 and 7135
4. Cables A and B can be replaced with a 9F - 25M straight cable if used with a modem device, part number for this cable is 40H6328

Legend
25M - 25-pin male connector
25F - 25-pin female connector
9F - 9-pin female connector
5P - 5-pin PCI connector
4P - 4-pin PCI connector

Table 7. CPC Connection Cables

7.1.4 CPC Configuration Rules

The following rules should be followed when connecting to and configuring the CPC.

- Connect and configure the UP system units first. For R10, R20 and R24 system units, connect the serial port 1 (S1) to the CPU (A-D) Port 1 and connect the power control port (R10/R20-J1, R24-J220) to the first available disk drive PCI connector (4-1, 4-2, 4-3, 4-4). For the deskside UP system units, connect the serial port 1 (S1) to the CPU (A-D) Port 1. Connect the serial port 2 (S2) of the UP system unit to the corresponding CPU (A-D) Port 2, this is optional and will allow a (remote) terminal to connect to the modem port of the CPC and be able to connect to the S2 port of the (UP) system unit.
- After connecting and configuring all the UP system units, connect and configure the SMP systems - S1 to CPU (A-D) Port 1 and S2 to corresponding CPU (A-D) Port 2.
- After configuring all the system units, connect the disk subsystems power control port (auxiliary port J19 or J20) to the available disk drive PCI connectors (4-1, 4-2, 4-3, 4-4, 4-5 or 4-6). There should not be anything connected to the main ports (J17 or J18) on the disk subsystem units.
- When one or more disk subsystems are connected to multiple system units, group the shared disks as a separate system so that the power can be controlled separately from the system units.
- The system unit key switch should be in the Normal position for remote power-on.

7.2 CPC Installation

The CPC can be connected to the SMP servers G30, G40, J30, J40, R30 and the R40 in different ways. For example, one CPC to one CPU, one CPC to multiple CPUs or multiple CPCs to multiple CPUs. The CPC also allows connection of disk drawers or deskside units. In the desktop (G30/G40) and the deskside (J30/J40) models, the CPC is physically a separate unit, external to the servers. Rubber feet are placed on the bottom of the CPC so that it can be placed on a shelf or rest on top of the server.

For rack-mounted units, the CPC is physically installed in a 7015-R00 rack. Two CPCs can be mounted in a single R00 rack if a customer requires dual power supplies. One of the CPCs will connect to one of the power distribution units and control half the devices in the rack. The other CPC will connect to other power distribution unit and control the remainder of the devices in the rack.

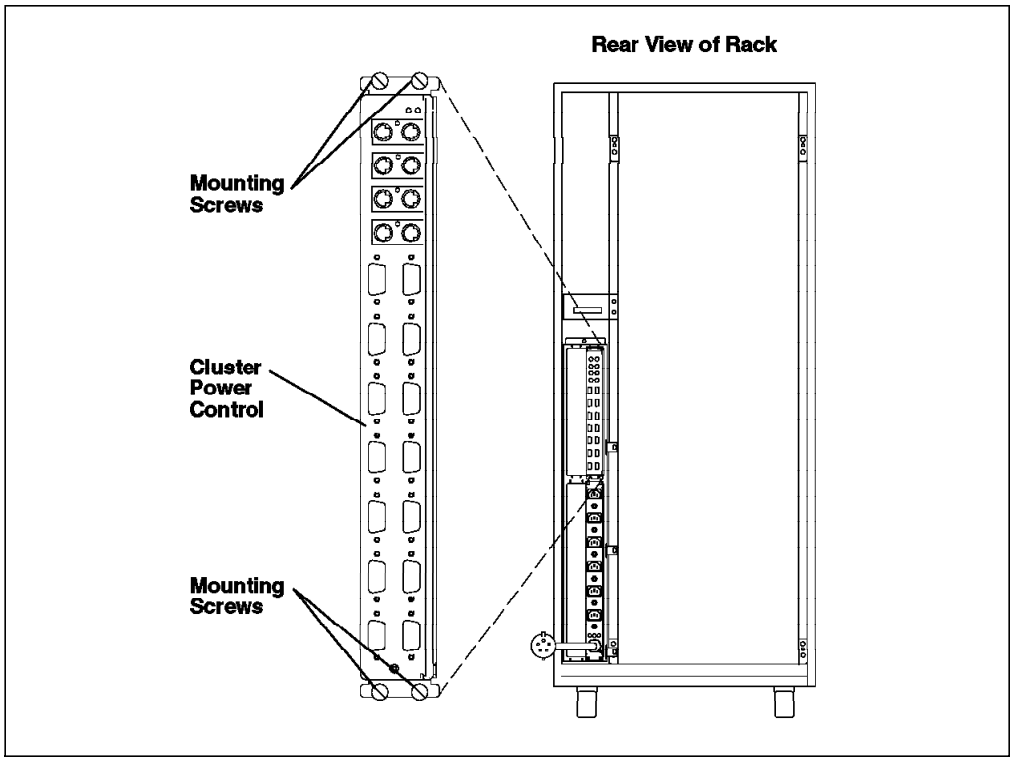


Figure 123. Rack-Mounted CPC

7.2.1 Prerequisites

Following are the prerequisites for CPC installation:

1. If installing the CPC (feature code 6175) in a 7015-R00 rack, verify at least one of the following feature codes is installed in the rack: feature codes 9171, 9173, 9174, 6171, 6173 or 6174.
2. An IBM 3151 ASCII terminal is available or a PC with terminal emulation. The microcode diskette comes with cpcterm, which is an IBM 3151 terminal emulation program.
3. You must have the microcode installation diskette and instructions, available in FBM 11H7663. The diskette is labelled Cluster Power Control Microcode

and should be Version 1.0 or later. The part number for this diskette is 11H7664.

4. AIX V3.2.5 or later for UP systems, AIX V4.1.4 with 604 SMP updates or AIX V4.2 for SMP servers.

7.2.2 General Installation Steps

Following are the basic installation steps.

1. Position the CPC near the system components or in a rack.
2. Connect the power and serial signal cables.
3. Attach a terminal to the tty port.
4. Attach a modem to the modem port (optional).
5. Power-on the CPC by plugging in the power cord.
6. Customize the CPC; that is, configure it and save the configuration.

7.2.3 CPC Power-On

The CPC does not have a power switch, so the power cord provides this function. When the power cord is plugged in and the electric outlet power is on, the CPC is powered on. Once powered on, the status of the CPC hardware is indicated by the lights (green and amber lights). The following is the description of the light status progress:

1. During the first 10 seconds, the POST runs and the lights stay off.
2. After 20 seconds, if the tests complete successfully, the green light will stay on. This indicates a console has been connected to the CPC.
3. If after 20 seconds, both lights start flashing, it means either:
   - There is no console plugged in the tty port of the CPC.

     or

   - The Secondary CPC in a multiple CPC or daisy-chained configuration is not connected correctly to the Primary CPC (the CPC with a terminal attached). See Section 7.4.7, “Daisy-Chaining CPCs” on page 199 for more information.
4. If after 20 seconds, the tests fail, only the amber light will stay on. The green light will be off. This indicates the CPC is defective and should be replaced.

Use the following table as a guide:

Green LED	Amber LED	Meaning
Off	Off	POST running or powered off
On	Off	CPC OK! Code running, main menu on TTY
Off	On	CPC faulty
On	On	CPC OK! and connected. This is a secondary CPC
Flashing	Flashing	POST successful but no console connected or secondary CPCs not connected correctly, or TTY - CPC or CPC - CPC cable failure

Table 8. Cluster Power Control Light Status Indicator

7.3 System Customization

Once the CPC is connected and successfully powered on, some basic configuration parameters need to be set. These parameters can be changed by accessing the Main menu of the CPC program which runs within the CPC. The Main menu appears on the attached console whenever the CPC is powered on. Connections to the various CPUs can be made using the menus in the CPC program. The Main menu can also be accessed while connected to a system unit by using the hot-key sequence.

To bring up the CPC Main menu:

1. The CPC should be powered on.
2. A terminal should be connected to the TTY port of the CPC.
3. The terminal setup parameters should be 9600 baud, 8 data bits, 1 stop bit with no parity. These are the default settings. These values can be changed using the Set Parameters menu of the CPC program.
4. Once the POST has run and the green LED is on, the CPC Main menu should appear on the console.

The CPC Main menu looks like:

```
CPC Microcode - Version 1.0  (03/14/95)









MAIN MENU - Console CPC:
================================

[0]  TTY
[1]  Modem

Select an Option: _
```

Figure 124. CPC Main Menu

The following figure shows the various CPC menu options and the paths required to access these options:

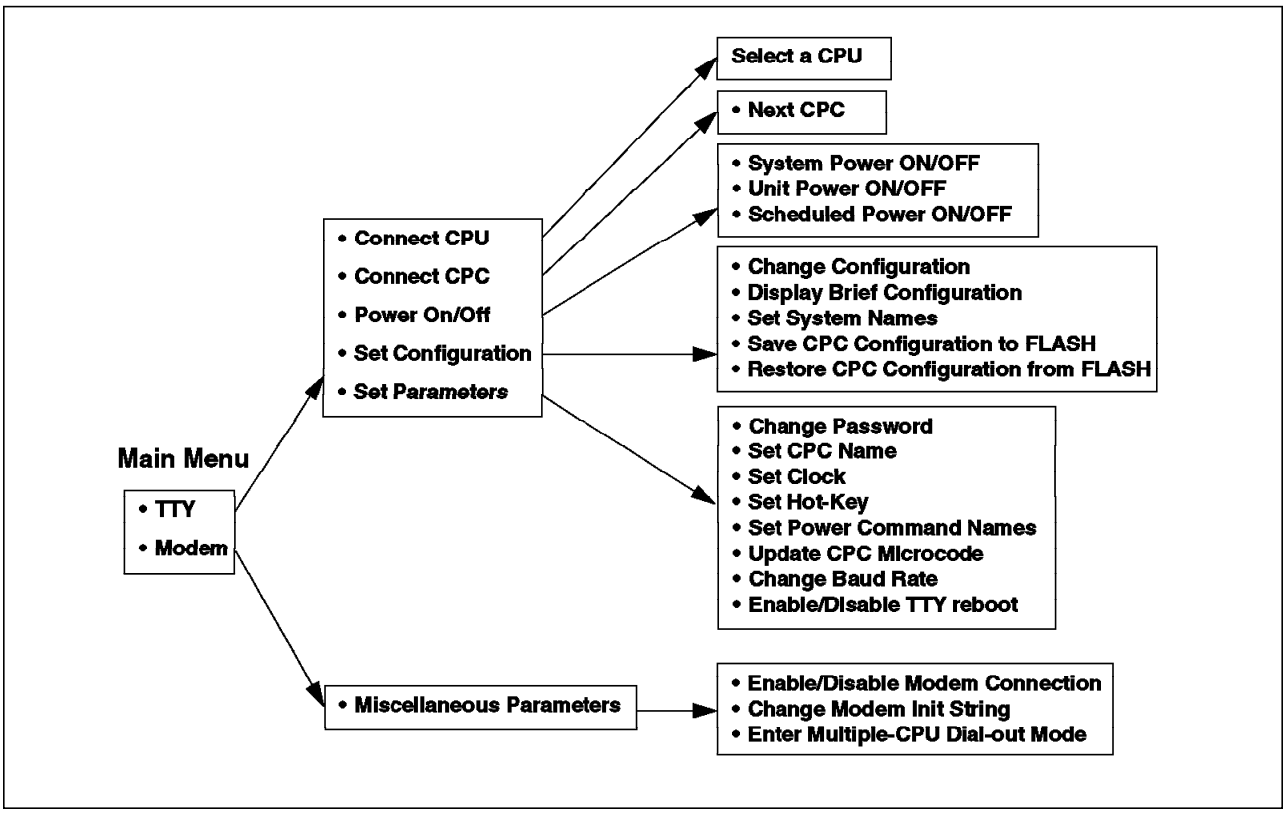


Figure 125. CPC Menu Options

After accessing the Main menu, the following steps should be performed to use the CPC.

1. Configure the CPC
2. Configure the CPUs
3. Configure the peripherals
4. Install the poweroff user

7.3.1 Configure the CPC

The following operations are performed to configure the CPC:

- Set the CPC name
- Set a password for the CPC
- Set the date and the time for the CPC

From the main menu, perform the following steps to configure the CPC:

1. Enter **0** to select the **TTY** option

   The following screen will appear:

```
CPC Microcode - Version 1.0  (03/14/95)









TTY MENU - Console CPC:
===============================
[0]  Connect CPU
[1]  Connect CPC
[2]  Power On/Off
[3]  Set Configuration
[4]  Set Parameters

Select an Option (x to exit):
```

Figure 126. TTY Menu

2. Enter **4** to select the **Set Parameters** option.

   The following screen will appear:

```
SET PARAMETERS
--------------
[0]  Change Password
[1]  Set CPC Name
[2]  Set Clock
[3]  Set Hot-key
[4]  Set Power Command Names
[5]  Update CPC Microcode
[6]  Change Baud Rate
[7]  Enable/Disable TTY Re-boot  (Currently: ENABLED)

Select an Option (x to exit): _
```

Figure 127. SET PARAMETERS Menu

3. Enter **1** to select **Set CPC Name**. The following line appears:

   ```
   This CPC NAME is currently set to:
   Enter this CPC's Name (up to 8 characters):
   ```

   Enter a unique name of the customer's choice, for example, cpc1. Once the name is entered, it gets saved and the screen goes back to the Set Parameters menu.

```
CPC NAME changed to: cpc1

Configuration saved - OK.
```

Figure 128. Set CPC Name

Note: For multiple CPCs daisy-chained together, it is important to give each CPC a unique name. This will be the only way to differentiate the port connections between the CPCs.

4. Enter **0** to set the password. By default, there is no password set on the CPC.

   This password allows usage of the CPC menus. The password is required once to enter the CPC Main menu. To be secure, the user should always exit the CPC Main menu and logout from any connected CPUs.

5. Enter **2** to select the **Set Clock** menu option.
6. Enter **0** from this menu to set the time.
7. Enter **1** from this menu to set the date.
8. Exit back to the TTY menu using **x**.

CPC configuration is complete; we are now ready to configure the system units (CPU).

7.3.2 Configure a CPU

Follow these steps to configure the CPU.

1. From the CPC Main menu, enter **0** to select the **TTY** menu.
2. Enter **3** to select the **Set Configuration** option.

   The following menu will appear:

```
SET CONFIGURATION
-----------------
[0]  Change Configuration
[1]  Display Brief Configuration
[2]  Set System Names
[3]  Save CPC Configuration to FLASH
[4]  Restore CPC Configuration from FLASH

Select an Option (x to exit): _
```

Figure 129. SET CONFIGURATION Menu

3. Enter **0** to select the **Change Configuration** option. After a few seconds, the following screen will appear:

```
CHANGE CONFIGURATION - SELECT UNIT
----------------------------------

     Conn   Unit Name         Type Description  PwrStat  Sys  System Name
     -----  ----------------  ----------------  -------  ---  -----------
[0]  CPU A                    empty             off       0
[1]  CPU B                    empty             off       0
[2]  CPU C                    empty             off       0
[3]  CPU D                    empty             off       0
[4]  4-1                      empty             off       0
[5]  4-2                      empty             off       0
[6]  4-3                      empty             off       0
[7]  4-4                      empty             off       0
[8]  4-5                      empty             off       0
[9]  4-6                      empty             off       0

Select an Option (x to exit): _
```

Figure 130. CHANGE CONFIGURATION - SELECT UNIT Menu

4. Enter **0** to select **CPU A**. The following screen will appear:

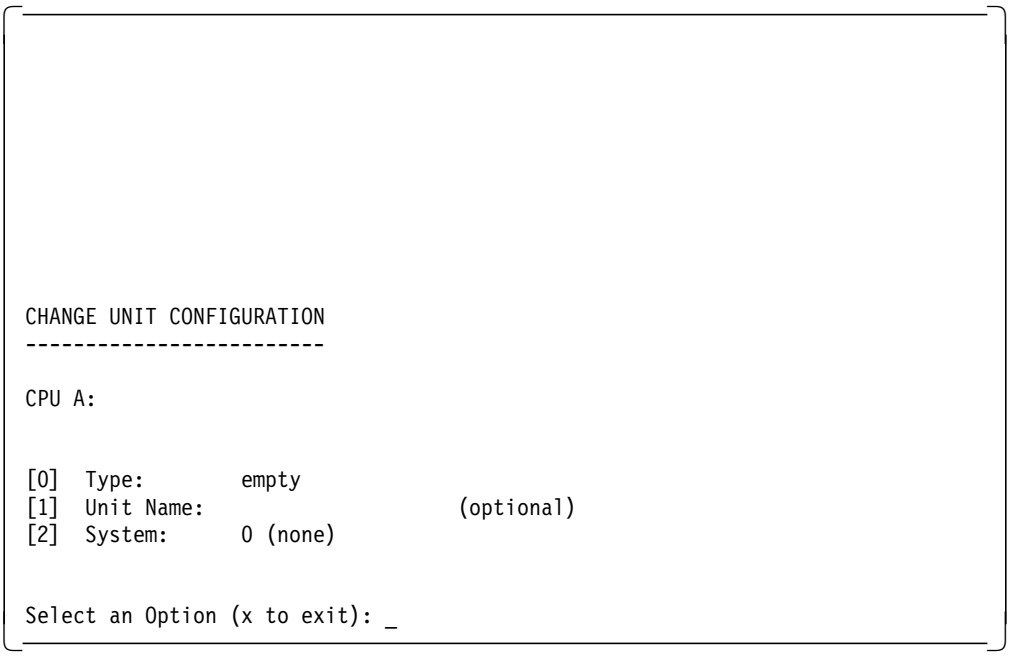


Figure 131. CHANGE UNIT CONFIGURATION

5. Enter **0** to set the CPU **Type**. Enter your CPU type, and press **Enter**. The CPU type is R = RS/6000 CPU (UP), P = PowerPC CPU (SMP) or . = empty (d=peripheral is used when configuring the 4-X ports).

   Note: All rack-mounted UP system units require a power control cable connection to the corresponding 4-X port to allow a remote power-on from the CPC program. CPU A uses 4-1 port, CPU B uses port 4-2, CPU C uses port 4-3 and CPU D uses port 4-4. This is the reason why the UP system units need to be connected and configured before the SMP system units and the peripherals.

   We configured a UP system on the CPU A ports, so the CPU type was set to R.

6. Enter **1** in the Change Unit Configuration menu, enter a unit name and then press **Enter**. This could be the serial number of CPU A or anything that the customer can use to easily identify the system unit. For this example, we used up1.

7. *(This step is optional)* Enter **2**, enter the System Number and press **Enter**. The system number is used to group system units and disk drawers for power (on/off) operations. The system number can be set to a value in the range of 1-39. For this example, we used 1.

8. *(This step is optional)* If a system number was set in the previous step, an additional field for the system name will appear. If the system number has already been used by another system unit, the existing system name will be displayed. Select **3**, and enter the system name (if you wish to set the system name or change the existing one) and then press **Enter**. The system name is used to identify the new system group; it is good to be as descriptive as possible. For this example, we used Backup Server for the system name.

   Here is an example of the screen output after the configuration:

```
CHANGE UNIT CONFIGURATION
-------------------------

CPU B: smp1

[0]  Type:        RS/6000 non-SMP
[1]  Unit Name:   up1              (optional)
[2]  System:      1
[3]  System Name: Backup Server    (optional)

Select an Option (x to exit): _
```

Figure 132. CHANGE UNIT CONFIGURATION

9. Enter **x** to exit back to the CHANGE CONFIGURATION - SELECT UNIT menu.

   Repeat this procedure for configuring additional system units. We configured an SMP system on the CPU B ports as System 2.

   The CHANGE CONFIGURATION - SELECT UNIT menu now looks like:

```
CHANGE CONFIGURATION - SELECT UNIT
----------------------------------

     Conn   Unit Name         Type Description  PwrStat  Sys  System Name
     -----  ----------------  ----------------  -------  ---  -----------
[0]  CPU A  up1               RS/6000 non-SMP   ON        1   Backup Server
[1]  CPU B  smp1              PowerPC-SMP       ON        2   Primary Server
[2]  CPU C                    empty             off       0
[3]  CPU D                    empty             off       0
[*]  4-1  (power control for CPU A)
[5]  4-2                      empty             off       0
[6]  4-3                      empty             off       0
[7]  4-4                      empty             off       0
[8]  4-5                      empty             off       0
[9]  4-6                      empty             off       0


Select an Option (x to exit): _
```

Figure 133. CHANGE CONFIGURATION - SELECT UNIT Menu

10. Enter **x** to exit back to the TTY menu.

7.3.3 Configure a Peripheral

Use the following procedure after configuring all the system units to configure the disk drawers or deskside units.

1. From the CPC Main menu, enter **0** to select the TTY menu.
2. Enter **3** to select the **Set Configuration** option.

   The following screen will appear:

```
SET CONFIGURATION
-----------------
[0]  Change Configuration
[1]  Display Brief Configuration
[2]  Set System Names
[3]  Save CPC Configuration to FLASH
[4]  Restore CPC Configuration from FLASH

Select an Option (x to exit): _
```

Figure 134. SET CONFIGURATION Menu

3. Enter **0** to select the **Change Configuration** option. After a few seconds, the following screen will appear:

```
CHANGE CONFIGURATION - SELECT UNIT
----------------------------------

     Conn   Unit Name         Type Description  PwrStat  Sys  System Name
     -----  ----------------  ----------------  -------  ---  -----------
[0]  CPU A  up1               RS/6000 non-SMP   ON        1   Backup Server
[1]  CPU B  smp1              PowerPC-SMP       ON        2   Primary Server
[2]  CPU C                    empty             off       0
[3]  CPU D                    empty             off       0
[*]  4-1  (power control for CPU A)
[5]  4-2                      empty             off       0
[6]  4-3                      empty             off       0
[7]  4-4                      empty             off       0
[8]  4-5                      empty             off       0
[9]  4-6                      empty             off       0

Select an Option (x to exit): _
```

Figure 135. CHANGE CONFIGURATION - SELECT UNIT Menu

4. Enter **5** to select the **4-2** port or the first available PCI port since the UP system units may have used some of these ports. There will be an asterisk (*) in place of the option number to select the port (4-X) if the PCI port is being reserved for power control for a UP system unit.

   The following screen will appear:

```
CHANGE UNIT CONFIGURATION
-------------------------

4-2:

[0]  Type:        empty
[1]  Unit Name:                    (optional)
[2]  System:      0  (none)

Select an Option (x to exit): _
```

Figure 136. CHANGE UNIT CONFIGURATION Menu

5. Enter **0** to select the **Type** option. The following line appears:

```
Set Peripheral Type (d = peripheral, . = empty):
```

Enter **d** for the peripheral type or **.** if it's empty.

6. (*This is optional*) Enter **1** to select the **Unit Name** option. The unit name should be something that the customer can use to uniquely identify that particular disk subsystem. For this example, we used diskunit1 for the Unit Name.
7. (*This is optional*) Enter **2** to select the **System** option. This can be a new unique system number, or it can be an existing system number that was created when configuring the system units earlier. The power for the disks and system units can then be controlled as a group. If the disks are shared between system units, they should be configured as a separate group or not configured in any group (System - 0 for none, so that the power will only be controlled on a per unit basis). For this example, we used 3 for the system number.

   Once the system number is entered, an additional field for the system name appears.
8. Enter **3** to select the **System Name** option. Enter a unique system name. Once again, you need to enter something that is easily identified by the customer. For this example, we used Shared 9334 for the system name.

```
CHANGE UNIT CONFIGURATION
-------------------------

4-2: diskunit1

[0]  Type:        Peripheral
[1]  Unit Name:   diskunit1         (optional)
[2]  System:      3
[3]  System Name: Shared 9334       (optional)

Select an Option (x to exit): _
```

Figure 137. CHANGE UNIT CONFIGURATION Menu

9. Enter **x** to return to the Change Configuration menu, and configure the remaining disk subsystems.

```
CHANGE CONFIGURATION - SELECT UNIT
----------------------------------

     Conn   Unit Name         Type Description  PwrStat  Sys  System Name
     -----  ----------------  ----------------  -------  ---  -----------
[0]  CPU A  up1               RS/6000 non-SMP   ON        1   Backup Server
[1]  CPU B  smp1              PowerPC-SMP       ON        2   Primary Server
[2]  CPU C                    empty             off       0
[3]  CPU D                    empty             off       0
[*]  4-1  (power control for CPU A)
[5]  4-2    diskunit1         Peripheral        ON        3   Shared 9334
[6]  4-3                      empty             off       0
[7]  4-4                      empty             off       0
[8]  4-5                      empty             off       0
[9]  4-6                      empty             off       0

Select an Option (x to exit): _
```

Figure 138. CHANGE CONFIGURATION - SELECT UNIT Menu

10. Enter **x** to exit to the Main menu.

The configuration will be saved when exiting the menus, so if the CPC reboots, it will be reconfigured using the updated parameters in the flash memory.

7.3.4 Installation of Poweroff User

For an orderly shutdown of the RS/6000, the CPC logs into the system using a user ID called ″poweroff″ with a default password. This user ID issues the shutdown -F command. The poweroff user can be installed using the script supplied with the CPC microcode diskette, part number 11H7664, or by manually entering all the commands.

Note: The dosread command is required for this procedure; so check that the *bos.dosutil* fileset is installed on each system unit.

Repeat the following procedure on each system unit.

1. Insert the CPC microcode diskette into the diskette drive on the RS/6000.
2. Log in as root and enter the following:

   ```
   # dosread -a /aix/powinst.aix /tmp/powerpasswd.install
   ```

3. Run the installation script.

   ```
   # sh /tmp/powerpasswd.install
   ```

```
# sh /tmp/powerpasswd.install
This adds a user named poweroff with the default poweroff password.
The /etc/passwd file is backed up as /etc/passwdbackup.
The /etc/security/passwd file is backed up as /etc/security/passwdbackup.
You can change the password for the poweroff using passwd.
#
```

Figure 139. Output of powerpasswd.install script

The figure above shows the output on the screen after you have run the powerpasswd.install script. Take note that the original password files are backed up accordingly.

4. Check that the poweroff user ID has been appended to the end of the /etc/passwd file.

   ```
   # grep poweroff /etc/passwd
   ```

7.3.5 Setting up Power-On/Off Command on the CPC

In order for Power-On/Off on the CPC to work successfully, you have to configure the parameters correctly. Below are the steps to configure them.

1. From the CPC Main menu, enter **0** to select the **TTY** menu.
2. Enter **4** to select **Set Parameters**.
   a. Enter **4** to **Set Power Command Names.** The available options are displayed.

```
Set Power On/Off Command Parameters
-----------------------------------

[0] Set Power-ON Command String
[1] Set Power-OFF Command UserID
[2] Set Power-OFF Command Password

Select an Option (x to exit): _
```

Figure 140. Set Power Command Names

   b. Enter the power-on command string and power-off command ID and password.

Ensure that the Power-OFF userid and password is valid and found in AIX. This is the default ID and password which you will get after the installation of `poweroff` user as mentioned in 7.3.4, “Installation of Poweroff User” on page 194. You can change the ID and password, but it has to match both the AIX and CPC menus.

Also ensure that the Power-ON string is set according to the power on command as used in the SystemGuard menu. The default string is `power`.

For the CPC power-on to work on the SMP servers, the server has to be in stand-by mode, and the system key has to be in Normal mode.

7.4 CPC Operations

After the CPC has been installed and customized, a number of different operations are now possible.

7.4.1 How to Connect and Log Into the CPUs

Before the terminal can be connected to the CPU, make sure that the cables are connected, the CPC has powered up with no errors, the CPC program is running with the Main menu displayed on the tty and the CPC has been customized.

To connect to a CPU:

1. From the CPC Main Menu, enter **0** to select the **tty** option. The following screen appears:

```
CPC Microcode - Version 1.0  (03/14/95)









[0]  Connect CPU
[1]  Connect CPC
[2]  Power On/Off
[3]  Set Configuration
[4]  Set Parameters

Select an Option (x to exit): _
```

Figure 141. CPC Program Menu

2. From this menu, enter **0** to select the **Connect CPU** option. A list of all the CPU ports will appear.

```
CONNECT TO CPU
--------------------

[0]  CPU A  name: up1                system: 1  system name: Backup Server
[1]  CPU B  name: smp1               system: 2  system name: Primary Server
[2]  CPU C  name:                    system: 0  system name:
[3]  CPU D  name:                    system: 0  system name:

Select an Option (x to exit): _
```

Figure 142. CONNECT TO CPU Menu

3. Select the corresponding number for the CPU you want to connect to. For example, enter 1 to connect to CPU B. The CPC connects to CPU B, and the following message will appear:

```
Connecting to CPU B (name: smp1)...
Hit Ctrl-T to EXIT back to CPC menus
CONNECTED.
```

Figure 143. Connecting to CPU B

After this connecting message, the login prompt will appear. If there is no login or command line prompt (if you were already logged on), press **Enter**. If there is still no prompt, the tty port might not be configured or enabled.

7.4.2 How to Power-On/Off Systems From the CPC

The power on/off feature allows you to control the power of the systems and peripheral drawers. This feature can be used through the CPC program menus. The following procedure details how to control a system or a unit's power:

To power-on or power-off a system:

1. From the CPC Main menu, enter **0** to select the **TTY** menu.
2. Enter **2** to select the **Power-On/Off** option.
   a. Enter **0** to select **System Power-On/Off**. The available systems are displayed. Select the corresponding number of the system you would like to perform the power-on/off function on.

b. The next menu will allow the power-on/off operation. Following are input commands in this menu:

```
System power (n = ON, f = OFF, x to exit):
```

OR

a. Enter 1 to select **Unit Power-On/Off**. A screen with all the CPUs and peripheral units will appear. Enter the corresponding unit number to select the unit you would like to power-on/off.

3. Once completed, exit to the previous menu by entering **x**.

7.4.3 How to Enable SystemGuard Dial-Out

To be able to utilize the dial-out feature offered by SystemGuard, you need to enable a parameter in the CPC program. This is done by:

1. From the CPC Main menu, enter **1** to select the **Modem** option.
2. From the Modem menu, enter **0** to select the **Miscellaneous Parameters** menu.
3. Enter **2** to enable the **Enter Multiple-CPU Dial-out Mode** feature.
4. Enter **x** to exit back to the previous menu.

This allows the modem to be shared by multiple SMP systems. The assumption is that only one system will be dialing out to report a hardware problem at any given time.

7.4.4 How to Enable the CPC Modem Connection

The CPC switches the TTY port and the modem between the various system units. The TTY port will be connected to the S1 port while the modem port will be connected to the S2 port of the same system. When either user (TTY or modem) accesses the CPC menus, the output is displayed on both ports. Input is accepted from both ports, so it is like mirroring at the CPC level. Once connected to a CPU, the sessions are independent, unless mirroring is activated on an SMP system unit between the S1 and S2 ports.

To activate the modem connection:

1. From the CPC Main menu, enter **0** to select the **Modem** menu.
2. Enter **0** to select the **Miscellaneous Parameters** menu.
3. Enter **0** from this menu to **Enable Modem Connection**. The status is displayed as part of the menu selection.

   ```
   [0] Enable/Disable Modem Connection (currently: ENABLED)
   ```

4. Enter **x** to exit from this menu.

7.4.5 How to disable TTY Reboot

The TTY Reboot capability is enabled when the CPC is shipped to the customer. This feature reboots the CPC if the terminal is turned off and then turn back on again. Some terminals do not have a power-saving function, or customers may want to turn the terminal off to save power. They may want to disable this feature so that the CPC is not rebooted every time the terminal is turned off and on again.

To disable the TTY Reboot feature:

1. Enter **0** to select the **TTY** menu from the CPC Main Menu.
2. Enter **4** to select the **Set Parameters option**
3. Enter **7** to toggle the TTY Reboot setting.
4. Exit back to the Main menu by entering **x**.

7.4.6 Microcode Update

The microcode for the CPC may have to be updated at some time to provide additional features. The microcode can be updated using a PC attached to the tty port or from one of the RS/6000s attached to one of the CPU ports. The xmodem protocol is used to download the microcode to the CPC from either source.

The following procedure copies the microcode to a RS/6000 running AIX and then downloads the microcode to the CPC.

1. Insert the CPC Microcode diskette into the diskette drive on the RS/6000.
2. Log in as root and enter the following:

   ```
   # dosread -a /aix/dos2aix.aix /tmp/dos2aix.aix
   ```

3. Run the script to copy all the diskette files to the /tmp directory.

   ```
   # sh /tmp/dos2aix.aix
   ```

4. Now, switch over to the CPC menu using the hot-key and select the terminal attached to the tty port on the CPC by selecting the **TTY** menu from the CPC Main menu.
5. Enter **4** to select the **Set Parameters** option.
6. Enter **5** to select the **Update CPC Microcode** option.
7. Enter **0** to select the **Set Download Source** option, and enter the number for the port that is connected to the RS/6000 that has the updated CPC flash code (for example, 1 for CPU-B).
8. Enter **1** to **Connect to Download Source**, and enter

   ```
   #/tmp/xmodem -sbKc /tmp/flash.bin
   ```

   and then enter **Ctrl-T** or the hot-key to get to the CPC menu.
9. Enter **2** to **Start Download of New Microcode**.
10. After a successful download of the microcode, enter **3** to **Update FLASH** with the downloaded microcode image. Do not power-off the CPC until it has completed the update.
11. Enter **4** to select **Re-boot CPC**, and enter **y** in response to the question. The CPC will reboot and the version number and the date of the updated microcode will appear at the top the CPC Main menu.

7.4.7 Daisy-Chaining CPCs

The following diagram illustrates three CPCs in a daisy-chained configuration.

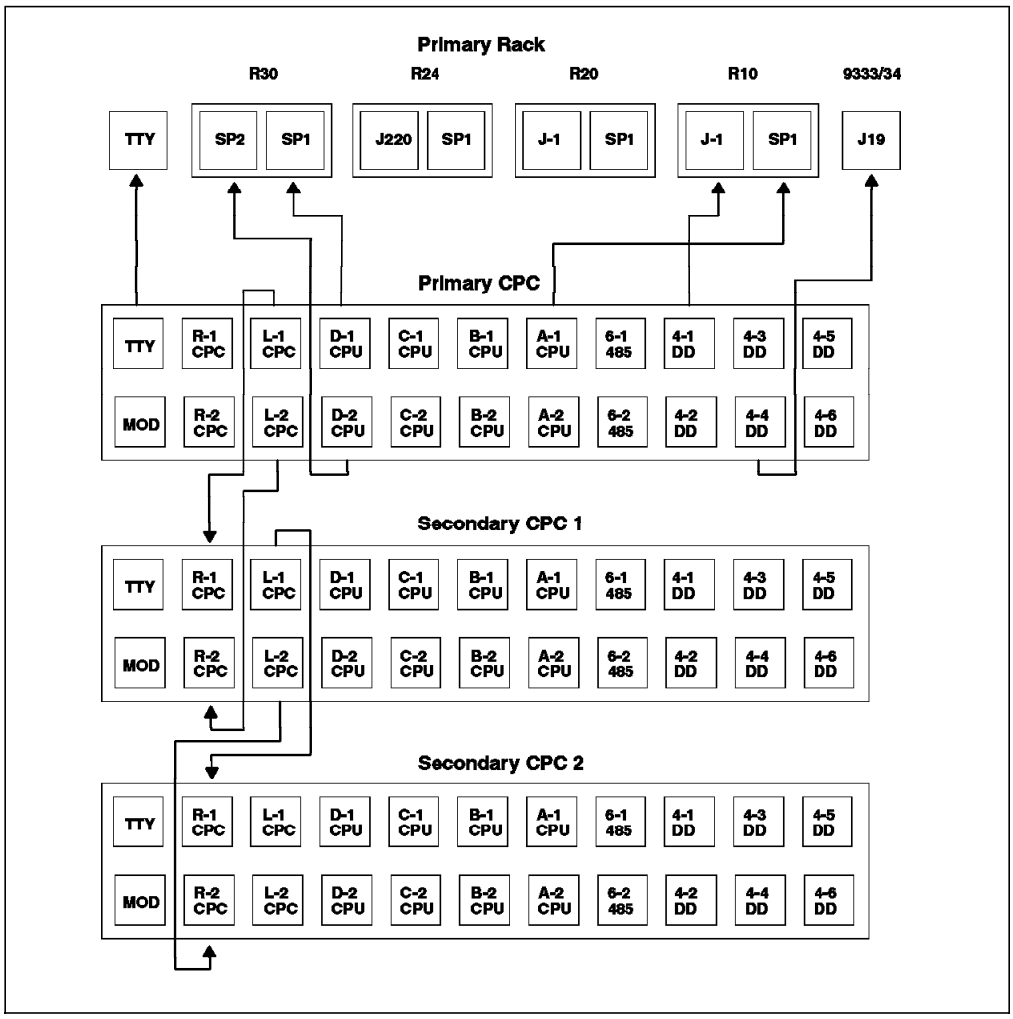


Figure 144. Daisy-Chaining CPCs

In this diagram, there are three CPCs. The connection between the CPCs uses the CPC left and CPC right connectors.

1. The CPC L-1 port of the first CPC (primary CPC) gets connected to the CPC R-1 port of the second CPC (secondary CPC1), the CPC L-2 port of the first CPC (primary CPC) gets connected to the CPC R-2 port of the second CPC (secondary CPC1)
2. The secondary CPC1 gets connected to the secondary CPC2 by connecting CPC L-1 port of the second CPC to CPC R-1 port of the third CPC (secondary CPC2) and connecting the CPC L-2 port of the second CPC to the CPC R-2 port of the third CPC.

7.4.8 How to Connect to a Secondary CPC

In a multiple CPC configuration, for example in a daisy-chained configuration, we need to be able to connect through to the first, second and third CPCs, and so on. This is can be carried out using the following procedure:

1. From the main menu, select the **TTY** option or modem option if connecting via the modem port.
2. The Console CPC menu appears. From this menu, enter **1** to select the **Connect CPC** option. The following screen appears:

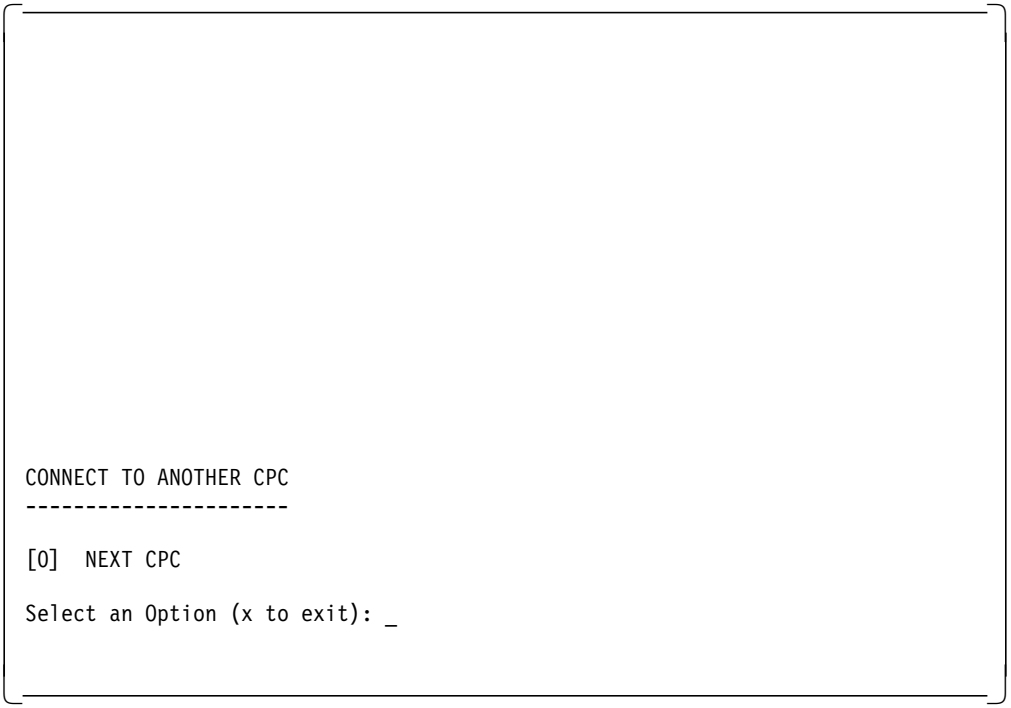


Figure 145. CPC Connect Menu

3. Enter a **0** from this menu to connect to the **Next CPC**.
4. Use the CPC menus to configure and use the Secondary CPC.

   Note: Set the hot-key for the Secondary CPC(s) to something other than Ctrl-T so that you can return to the Secondary CPC menus after connecting to a CPU on the Secondary CPC.
5. Enter **Ctrl-T** to return to the Primary CPC menus.

Chapter 8. Installing an SMP Server

The objective of this chapter is to describe how to install an SMP system with AIX V4.1.4 or AIX V4.2. Installing an SMP server is not much different from installing a UP system. However, there are some specifics to the SMP that we would like to highlight in this chapter.

Note

The 604 SMP Servers require at least AIX V4.1.4 with 604 SMP updates or AIX V4.2. The 604 SMP updates, necessary to upgrade a 601 SMP Server to 604 processors, can be ordered via APAR IX57164.

You will see details about these updates in the section 8.2.1, “What is AIX V4.1.4 with 604 SMP Updates?” on page 216.

The intention of this chapter is not to cover all of the AIX V4.1.4 features and enhancements, or those of AIX V4.2, but to make you more comfortable with the items that relate to installing and managing your SMP server.

The areas covered are organized into three main parts:

1. AIX V4 - General Considerations
2. Installing an SMP with AIX V4.1.4
3. Installing an SMP with AIX V4.2

8.1 AIX V4 - General Considerations

The purpose of this first section is to explain how AIX V4 is packaged, highlight SMP specifics and give an overview of the software maintenance mechanism.

8.1.1 AIX V4 Packaging

AIX Version 4 represents the most significant enhancement to AIX since its initial introduction. One of the most interesting characteristics is the way AIX V4 is packaged.

8.1.1.1 Packaging Terminology

It is very important to understand the terminology of *fileset* and *package* as they relate to the packaging of AIX Version 4 and associated products.

Fileset is the POSIX term for the smallest installable unit within a product. A fileset is part of a package. For example, bos.net.tcp.client is a fileset for the bos.net package. This packaging allows installation of only what is necessary, (one or several specific filesets) generally requiring less disk space.

A **package** is a collection of filesets that are built together to form one installable image file, which is in a Backup File Format (BFF). For example, bos.net is a package.

There are also some additional packaging terms that might be useful to know.

A **Licensed Program Products** (LPPs) is a purchasable product. It can be a collection of packages, or it can be a single package. For example, items such as BOS, X11, and SNA, are all LPPs. An LPP does not have to be contained in a single Backup File Format image.

A **Fileset Update** is an update that corrects or enhances functions in a previously installed fileset. Since each fileset can be serviced separately, fixes for AIX V4 (fileset updates) are smaller and more localized.

A **Bundle** is a file that contains a number of filesets for installation. It can be thought of as an installation profile. A number of bundles are supplied with AIX Version 4 for various environments, such as client, server, and application development. A system administrator can create his or her own bundles if the default bundles are not suitable.

An **Update Bundle** is a collection of fixes and enhancements that updates the installed software to the latest level available on the media.

A **Maintenance Bundle** is a predetermined level of fixes and enhancements for the Base Operating System (BOS).

A **Product Offering** is a selected set of packages (or LPPs) which are shipped together on the same physical media.

8.1.1.2 Fileset Names

The following conventions are used in AIX V4 for naming a software package and its filesets for AIX V4 :

- All the fileset names must be of the form Package_Name.Option where Option is unique for the package, and Package_Name is the name of the package or LPP name. For example, *bos* or *X11*.
- The packages for a given product should begin with the product name followed by a dot (.).
- If a package has only one installable fileset, then the fileset name may be the same as the product name.
- All package names must be unique. Two packages with the same name is not allowed.
- All fileset names must be made up of ASCII characters.
- Fileset names must be greater than one character in length and must begin with a letter or an underscore (_). Subsequent characters must be a letter, a digit, an underscore, or a dot (.).
- The maximum length for a fileset name is 144 bytes.

8.1.1.3 Standard Fileset Names

Standard fileset name extensions exist for certain types of filesets to help identify their usage. There is no requirement that any of these names be used in the name of a fileset.

.rte	Runtime or minimum set of commands and libraries
.adt	Application Development Toolkit
.data	/usr/share portion of a product
.fnt	Font portion of a product
.diag	Diagnostics for a product
.ucode	Microcode for a product

.smit	SMIT tools and dialogues for a product
.mp	SMP unique code
.up	Uniprocessor-unique code
.compat	Compatibility code (to be removed in a future release)
.loc	Locale files for a specific fileset
.msg.(lang)	Message catalogues for a specific fileset
.info.(lang)	InfoExplorer Databases for a specific fileset
.help.(lang)	Help Dialogs for a specific fileset

8.1.1.4 Device Driver Packaging

The AIX V4 Configuration Manager (`cfgmgr`) will automatically install software support for detectable devices, where the naming convention for device driver packaging is *devices.bus_type.card_id* and where the *bus_type* is:

mca	for MicroChannel
pci	for PCI Bus
isa	for ISA Bus
sys	for RSC Bus
buc	for 601 Bus
sio	for system planar
pcmcia	for PCMCIA devices

card_id is the unique hexadecimal card identifier.

For example, the package name for FDDI device support is `devices.mca.8ef4`, and the package name for parallel printer port support is `devices.sio.ppa`.

Each adapter to the system has three parts: the configuration method and device driver, diagnostics, and microcode. For example, the filesets for the FDDI adapter will be called:

devices.mca.8ef4.rte	Device driver code and config methods
devices.mca.8ef4.diag	Diagnostic code for FDDI
devices.mca.8ef4.ucode	Microcode for FDDI

sys is the RISC Single Chip (RSC) bus that is used to connect the graphic adapters in the RS/6000 Models 220 and 230. **buc** is the PowerPC 601 bus that is used to connect the graphic adapters in the RS/6000 Models 25x and 41x. **sio** is used for the devices that are connected to the system planar, such as diskette drives, serial and parallel ports, keyboard, and mouse.

8.1.1.5 Message Catalog Packaging

The messages, locales, convertors, help, and info databases are all shipped by language. The language in a fileset name is commonly indicated by the *lang* wildcard since most language-dependent filesets are translated into more than one language.

The naming convention for all message packages shipped in AIX V4 is:
LPP.msg.(LANG)(.Fileset_Name)

where *LANG* is any of the supported languages. Fileset_Name is optional since there may be only one message fileset for the LPP.

Some examples :

- bos.net.nfs.client messages will be packaged in bos.msg.(lang).net.nfs.client.

- bos.msg.en_US.net.tcp.client is TCP/IP Client Support Messages for US English.

AIX V4 will automatically install the message filesets for the Primary Language for filesets that use this naming convention. The Primary Language can be specified during BOS installation. Additional language filesets can be installed after the system has been installed.

This naming convention is also used for the other language-dependent filesets, such as InfoExplorer databases, COSE help files, and language converters.

8.1.1.6 Package Installation Database

A fileset called pkg_gd (Package Guide) is shipped on the installation media to provide the current information on the various products available for AIX V4. It is not installed automatically; so you have to manually install it from the installation media. It contains information about:

- Package and Fileset Names for LPPs available on AIX V4.1
- Approximate Disk Space Required
- Requisite Software
- Special Installation Notes
- Special Migration Notes

To view the database, you need to issue the command `info -l lp_info`.

Note

The pkg_gd fileset was not provided in the original AIX V4.1.4, but on August 16th 1996, the AIX V4.1.4 G5 server installation media was updated to include the 604 SMP updates and other filesets as the pkg_gd fileset. However, it is possible to order it from FixDist (APAR IX55799), and it is provided in AIX V4.2. For further informations about AIX 4.1.4 updated, please refer to section 8.2.1, “What is AIX V4.1.4 with 604 SMP Updates?” on page 216.

8.1.2 SMP Specifics

This section introduces some AIX V4 specifics to the SMP servers.

8.1.2.1 AIX V4 Levels for 604 SMP Servers

A number of levels of AIX V4 are already available, and additional support will be added over time. The following list shows the AIX V4 levels that are supported on the SMP servers with 604 processors :

- AIX V4.1.4 with 604 SMP updates or later
- AIX V4.2

8.1.2.2 MP Kernel

AIX V4 is shipped with two kernels: an MP kernel and a UP kernel. The two kernels share common code, but the locking required for an SMP environment is eliminated for the UP kernel. There are three key filesets in this regard.

- In AIX V4.1 :

bos.rte This fileset is used to ship all files that are common to both the UP and the MP.

bos.rte.up This fileset contains files specific to a UP (like the UP kernel).

bos.rte.mp This fileset contains files specifics to an SMP (like the MP kernel)

- In AIX V4.2 :

  bos.rte This fileset is used to ship all files that are common to both the UP and the MP.
  bos.up This fileset contains files specific to a UP (like the UP kernel).
  bos.mp This fileset contains files specifics to an SMP (like the MP kernel)

The BOS Runtime fileset, bos.rte, becomes a number of smaller filesets after installation. These filesets are bos.rte.* and are used for updates only.

When installing a system, the correct kernel will be installed, depending on the type of system you are dealing with.

In fact, as part of the BOS installation, the command `bootinfo -z` determines whether the system is MP-capable or not. Then the BOS installation process installs the appropriate fileset and links /unix to the correct kernel.

This link is only used by the `bosboot` command when rebuilding the boot image.

You can install both the UP and the MP filesets on the same system. This is used when setting up a Network Install server for AIX V4.

Attention !

Note that **the MP kernel will work on a UP system**, if, when creating the boot image, you specified the corresponding UP platform type or used the default one.

However, a **UP kernel will not work on an MP system** because there is some MP hardware-specific support that is not included in the UP kernel. You will get an LED code of 911 if you try to boot an SMP server with a UP boot image.

Note: When booting from the product CD-ROM or tape, the MP kernel is used. This eliminates the need to build different versions of the installation media for the two systems.

In AIX V4.2, the bos.mp fileset contains the following files:

```
# lslpp -f bos.mp
         Fileset               File
  ----------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.mp 4.2.0.0        /usr/sbin/open_door
                        /usr/sbin/cpu_state
                        /usr/sbin/bindprocessor
                        /usr/lib/boot/unix_mp

Path: /etc/objrepos
  bos.mp 4.2.0.0        NONE
```

The file /usr/lib/boot/unix_mp is the MP kernel. The /usr/sbin/cpu_state and /usr/sbin/bindprocessor files are discussed in Chapter 10, “SMP Performance Tools” on page 293.

8.1.2.3 Platform Types

The system device type is an abstraction that allows machines to be grouped according to fundamental configuration characteristics, such as number of processors and I/O bus structure.

The system device, sys0, is the highest-level device in the system node, which consists of all physical devices in the system.

Machines with different system device types have basic differences in the way their devices are dynamically configured at boot time.

rs6k applies to all of the uniprocessor RS/6000 that have a MicroChannel bus.

rs6ksmp applies to symmetric multiprocessor models.

rspc applies to the PReP-compliant systems that have a PCI and ISA bus, such as the RS/6000 40P, 43P, E20, E30, F30, F40.

In AIX V3.2.5, the prototype files used by the bosboot command to build boot images were dependent on the boot device. This is still true in AIX V4, but in addition, the prototype files are dependent on the system device type (sys0) of the machine for which the boot image is built.

This is reflected in the names of the prototype files:

/usr/lib/boot/rs6k.cd.proto

/usr/lib/boot/rs6k.disk.proto

/usr/lib/boot/rs6k.tape.proto

/usr/lib/boot/rs6ksmp.cd.proto

/usr/lib/boot/rs6ksmp.disk.proto

/usr/lib/boot/rs6ksmp.tape.proto

/usr/lib/boot/rspc.cd.proto

/usr/lib/boot/rspc.disk.proto

/usr/lib/boot/rspc.tape.proto

/usr/lib/boot/network/rs6k.ent.proto

/usr/lib/boot/network/rs6k.fddi.proto

/usr/lib/boot/network/rs6k.tok.proto

/usr/lib/boot/network/rs6ksmp.ent.proto

/usr/lib/boot/network/rs6ksmp.tok.proto

/usr/lib/boot/network/rspc.ent.proto

/usr/lib/boot/network/rspc.tok.proto

These files, in addition to the configuration methods, are contained in the following filesets:

devices.base.*

devices.rs6ksmp.base.*

devices.rspc.base.*

Attention !

Beginning with AIX V4.2, the rs6ksmp platform is represented in NIM by the rs6k platform with a netboot_kernel type of mp. All references to the rs6ksmp platform, in NIM, will be removed.

8.1.2.4 Determining the Platform Type

The bootinfo command is used during the boot and BOS install phases to gather and display information.

The bootinfo command can be used to determine the type of platform you are using. Also this command can help you in determining the current boot device, default install disk, and a variety of other boot information.

These are some of the options that we found to be of most interest to the SMP servers.

`bootinfo -z`	will return a numeric:	
	0	machine is not MP-capable
	1	machine is MP-capable
`bootinfo -T`	will return one of the following:	
	rs6k	means RS/6000 UP
	rs6ksmp	means RS/6000 SMP
	rspc	means PowerPC PReP System
`bootinfo -r`	will display the amount of real memory in kilobytes	
`bootinfo -t`	will list the type of boot, and the following are the responses:	
	1	Disk boot
	3	CD-ROM boot
	4	Tape boot
	5	Network boot
`bootinfo -k`	will display the key position:	
	1	Secure position
	2	Service position
	3	Normal position

8.1.2.5 Creating an MP Boot Image

The bosboot command is used to create a boot image. It uses the prototype files listed in 8.1.2.3, “Platform Types” on page 208.

However, any boot image will only support a single platform type and a single boot device type. The supported boot devices are token-ring, Ethernet, FDDI, CD-ROM, disk, or tape.

The command you use to create an MP boot image is:

```
# bosboot -a -d hdisk0 -k /usr/lib/boot/unix_mp
```

where

-a Creates a complete boot image and device.
-d Specifies the boot device. This flag is optional for the hard disk.
-k Specifies the full path to the kernel. This is optional since bosboot uses the /unix link in order to build the correct boot image (UP or MP).

Three other flags could be of interest :

-L Enables lock instrumentation for MP servers for use with the lockstat command. This flag has no effect on systems that are not using the MP kernel, and it does create some overhead when used on a production SMP server.
-T Specifies the platform for the boot image. This is optional since the platform type is taken from the system where the bosboot command is run.
-U Creates an uncompressed boot image. The boot image is compressed by default. If you wish to use this flag, ensure that the boot logical volume is large enough for the uncompressed boot image.

8.1.2.6 SMP CPU-ID

On traditional RS/6000 UP systems, the output of the uname -m command is tied to the serial number of the CPU card.

Since an SMP server has several processors, the CPU-ID used for product licenses cannot be tied to a processor and must be unique.

Therefore, for SMP servers, a unique value for the CPU-ID will be created for each chassis, and this value will be maintained through upgrades (that is, adding new processors or changing the processor technology).

The CPU-ID is part of the VPD (Vital Product Data) of the system. The SID Y2 field in the System EEPROM is used to build the CPU-ID of the system.

uname -m will give an output similar to xxyyyyyymmss (for example - 00645067A000) where:

xx and ss are always 00

yyyyyy is the CPU-ID

mm is the model ID depending on what server you are running on, and if it is an upgrade or not.

Note

An x30 system upgraded to 604 processors will keep the same model ID. You might find in some documentation the name x30* (star) to refer to an x30 machine upgraded to the 604.

To see what processors are installed in the system, you can use the following command:

`# lsattr -El proc<n>` - where `<n>` is the processor ID you want to check

The output will display either PowerPC_601 or PowerPC_604.

The tables below show the model ID for the different SMP systems:

J30, G30 and R30 upgraded to 604 Processors			
RS/6000	7013-G30 *	7012-J30*	7015-R30 *
Model ID	A6	A0	A3

J40, G40 and R40			
RS/6000	7013-G40	7012-J40	7015-R40
Model ID	A7	A1	A4

Note: For SMP servers built in Austin, the CPU-ID is the serial number of the machine. This is not the case for machines built in EMEA.

8.1.2.7 Specific SMP Devices

If you issue the following command:

```
# lsdev -C
```

you will see a number of devices that are not present in UP systems.

```
cabinet0      Available 00-00        Cabinet
op_panel0     Available 00-00        Operator Panel
mcaplanar0    Available 00-00        MCA Planar
sif0          Available 00-00        Power Supply Interface
power_supply0 Available 00-00        Power Supply
cpucard0      Available 00-0P        CPU card
L2cache0      Available 00-0P-00-0L  L2 Cache
proc0         Available 00-0P-00-00  Processor
proc1         Available 00-0P-00-01  Processor
cpucard1      Available 00-0Q        CPU card
L2cache1      Available 00-0Q-00-0L  L2 Cache
proc2         Available 00-0Q-00-00  Processor
proc3         Available 00-0Q-00-01  Processor
```

In this system, you can see two CPU cards (cpucard0 and cpucard1). There are two processors on each card (proc0 and proc1 for cpucard0) with a single device for the level 2 cache (L2cache0), even though each processor has its own dedicated level 2 cache. The J40 and R40 will also show two microchannel busses (bus0 and bus1).

8.1.3 AIX V4 Software Maintenance

In AIX V4, the software maintenance levels are identified by Version Release Modification Fix (VRMF) levels. The VRMF levels can be displayed by using the `lslpp` command. You will no longer see any Program Temporary Fix (PTF) numbers, for instance U4xxxxx. For example, the AIX V4.2 bos.mp fileset has a VRMF of 4.2.0.0, indicating Version 4, Release 2, Modification level 0, and Fix level 0.

The *Version* number is incremented to indicate a new product or the repackaging of an existing product. Versions include major functional enhancements and typically come two or more years apart.

The *Release* number is incremented to indicate new enhancements or new functions. Releases of AIX V4.2 will come approximately one year apart.

The *Modification* number is incremented whenever an accumulation of maintenance is added to a fileset. A modification level can also include support for new processors or devices where this support does not affect the behavior of the product on existing systems. Whenever the modification level is adjusted, the fix level is reset to zero. The modification levels for AIX V4.2 will come 3-6 months apart.

The *Fix* level is incremented whenever a fix is added to the fileset.

Each fileset in AIX V4 can be serviced separately. Fixes are delivered in fileset packages. Changes to filesets are cumulative, meaning that each new level of a fileset contains all the previous changes.

Maintenance and fixes should not change application programming interfaces so that applications that are written to the documented programming interfaces will function identically on different maintenance and fix levels. This is important for software vendors or customers for certification of their applications on AIX V4.

8.1.3.1 Fileset VRMF Numbering

The following diagram is used to show that the various filesets will show a different VRMF for a particular level of AIX V4.1. For an AIX V4.1.4 update, only some of the filesets will be at level 4.1.4.0.

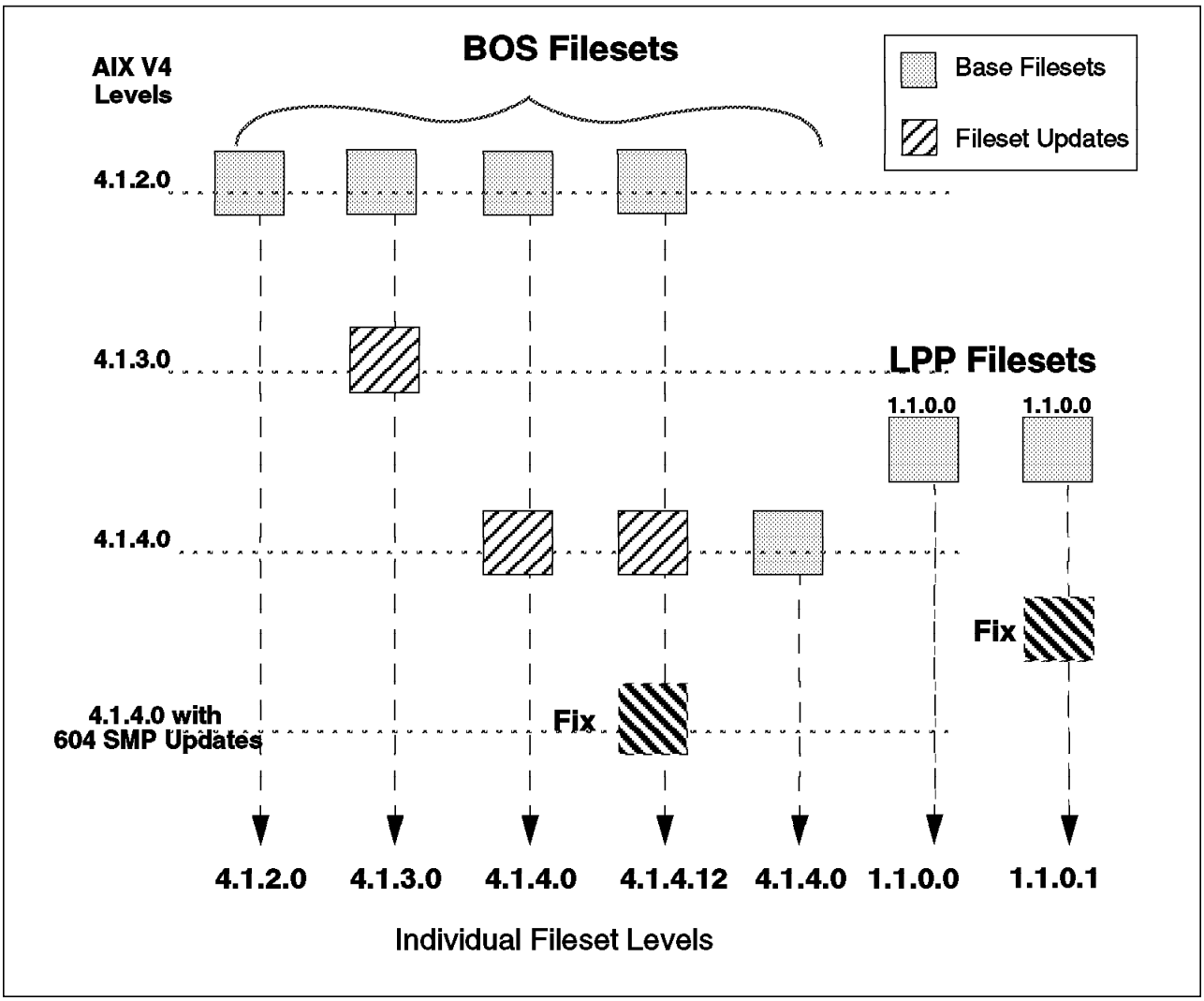


Figure 146. Fileset Numbering Examples

If the fileset for the base operating system was shipped in 4.1.2 and if there were absolutely no code changes to that fileset since 4.1.2, then, the VRMF is unchanged.

If the fileset for the base operating system was shipped in 4.1.2 and if there were changes to the code for 4.1.3 or 4.1.4, the modification level for the VRMF will be adjusted to be in line with the operating-system level. For example, the level of bos.rte.mp was 4.1.3.0 for AIX V4.1.3 and 4.1.4.0 for AIX V4.1.4. The content of the fileset was modified for both levels of AIX V4.1.

If a fix is added to a fileset, then, the fix level is incremented. For example, to install an SMP with AIX V4.1.4 with 604 SMP updates, the bos.rte.mp fileset has to be at the 4.1.4.12 level; so you need to add the corresponding fix.

If there is a new (additional) fileset for the base operating system or an LPP, the VRMF will be the level of the operating system or the LPP that it ships with.

If the LPP is new for V4.1, the VRMF level can really be set to any version or release value (but probably 1.1.0.0). If the LPP filesets were shipped on AIX V3.2, the version or release need only be bumped (+1), but the modification should

be reset to 0. For example, AIXLink/X.25 LPP was new for AIX V4.1.2, and the VRMF for all the sx25 filesets was set to 1.1.0.0.

8.1.3.2 oslevel Command

The oslevel command has been retained from AIX V3.2.4. Its behavior is slightly different due to the new VRMF format (and it is quicker!).

```
# oslevel -?
Usage: oslevel ( -l <level> | -g | -q )
     -l : List filesets at levels earlier than maintenance level
          specified by the <level> parameter
     -g : List filesets at levels later than most recent
          complete maintenance level
     -q : List names of known maintenance levels which may be
          specified with the -l flag

  Output indicates that base system software is entirely at
  or above a particular maintenance level. Corresponding output
  would be 4.1.1.0 first AIX V4.1 maintenance level.

  The additional options may be specified to determine which
  filesets differ from the maintenance level
```

To check the level of the operating system, enter:

```
# oslevel
4.2.0.0
```

To list the names of the known Maintenance Levels, enter:

```
# oslevel -q
Known Maintenance Levels
------------------------
4.2.0.0
```

The VRMF information that the oslevel command uses for all the base operating system filesets is now stored in an ODM database. There is a specific entry in the /usr/lib/objrepos/fix database for each Maintenance Level. It looks like this:

```
# ODMDIR=/usr/lib/objrepos odmget fix | pg
fix:
        name = "4.2.0.0_AIX_ML"
        abstract = "AIX V4.2.0.0 Maintenance Level"
        type = "p"
        filesets = "bos.acct:4.2.0.0
bos.adt.base:4.2.0.0
bos.adt.debug:4.2.0.0
bos.adt.graphics:4.2.0.0
bos.adt.include:4.2.0.0
 ...
```

8.1.3.3 Instfix Command

The `instfix` command, which has been introduced with AIX V4.1, has a number of options to :

- List the contents of an update media
- Search for a fix number on a media
- Search in ODM for installed fixes
- Search for a key word
- Give fix abstracts

Following are some examples:

To list the entire fixes Table of Contents on the media, enter:

```
# instfix -T -d /dev/rmt0.1
```

To install all filesets associated with fix IX38794 from the tape mounted on /dev/rmt0.1, enter:

```
# instfix -k IX38794 -d /dev/rmt0.1
```

To install all fixes on the media in the tape drive, enter:

```
# instfix -T -d /dev/rmt0.1 | instfix -d /dev/rmt0.1 -f
```

The first part of this command lists the fixes on the media, and the second part of this command uses the list as input.

To list all entries on the tape using a keyword search string of SCSI, enter:

```
# instfix -s SCSI -d /dev/rmt0.1
```

To inform the user on whether fixes IX38794 and IX48523 are installed, enter:

```
# instfix -i -k "IX38794 IX48523"
```

8.2 Installing an SMP Server with AIX V4.1.4

AIX Version 4.1.4 is a modification level of AIX Version 4.1 operating system. This modification level includes :

- New AIX Connections functions
- Extended hardware support
- Additional TCP/IP functions
- License under the new International Program License Agreement (IPLA)
- Additional languages for the Program Integrated Information (PII)

For more details about AIX V4.1.4 additional features, please refer to the IBM site:

`http://www.austin.ibm.com/software/OS/aix414.html`

You can also refer to the following site, which is only available for IBMers:

`http://w3.austin.ibm.com/afs/austin/depts/service/public_html/releases/`

8.2.1 What is AIX V4.1.4 with 604 SMP Updates?

604 SMP servers need at least AIX V4.1.4 with several filesets at a required level or higher. These following filesets can be ordered via APAR IX57164:

bos.diag.rte	4.1.4.4
bos.rte.mp	4.1.4.12
devices.isa_sio.km.diag	4.1.4.2
devices.mca.8f95.diag	4.1.4.1
devices.mca.8f95.rte	4.1.4.3
devices.mca.edd0.com	4.1.4.7
devices.mca.f6f8.diag	4.1.3.1
devices.scsi.disk.diag.com	4.1.4.1
devices.scsi.disk.diag.rte	4.1.4.2
devices.sio.ktma.diag	4.1.4.2
X11.base.rte	4.1.4.6

APAR IX57164 is only a packaging APAR. It will not appear in the list of APARs on the Install Fileset Updates by Fix panel in SMIT, nor will you be able to see this APAR installed after installing the 604 SMP updates using:

```
instfix -ik IX57164
```

Since August 16, 1996, the AIX V4.1.4 G5 server installation media is updated to include the 604 SMP updates. That means that the first AIX V4.1.4 medias did not have these required levels, but these levels are now included in the updated G5 Server media, labeled :

5765-393	AIX 4.1.4 for Server for G5 Processors (Includes SMP Support)

8.2.2 Updating AIX V4 to AIX V4.1.4 with 604 SMP Updates

The following are the steps to carry out when you want to update your 601 SMP server to AIX V4.1.4 with 604 SMP updates in order to support a 604 upgrade:

- Read carefully the *AIX 4.1.4 with 604 SMP Updates Release Notes.*
- Log in as root.
- Insert the AIX V4.1.4 media containing the 604 SMP updates into the appropriate drive.
- Bring-up the appropriate SMIT menu for the ASCII version, as shown in Figure 147 on page 217.

  ```
  # smitty update_all
  ```

```
Update Installed Software to Latest Level (Update All)

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* INPUT device / directory for software               /dev/cd0
* SOFTWARE to update                                  _update_all
  PREVIEW only? (update operation will NOT occur)     no +
  COMMIT software updates?                            yes +
  SAVE replaced files?                                no +
  AUTOMATICALLY install requisite software?           yes +
  EXTEND file systems if space needed?                yes +
  VERIFY install and check file sizes?                no +
  DETAILED output?                                    no +


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
```

Figure 147. ASCII Version - Update All Menu

or bring-up the appropriate VSM menu for the GUI version, as shown in Figure 148 on page 218

```
# xinstallm &
```

- Press the **Enter** key for the ASCII version or click on the **Install/Update Action** for the GUI version.

  At this point, the system will check filesets and install all the fixes, and the new filesets, that are available on the installation media.

  Attention!

  After installing the 604 SMP updates, the system will need to be rebooted immediately. Until the system is rebooted, users will not be able to login to the system.

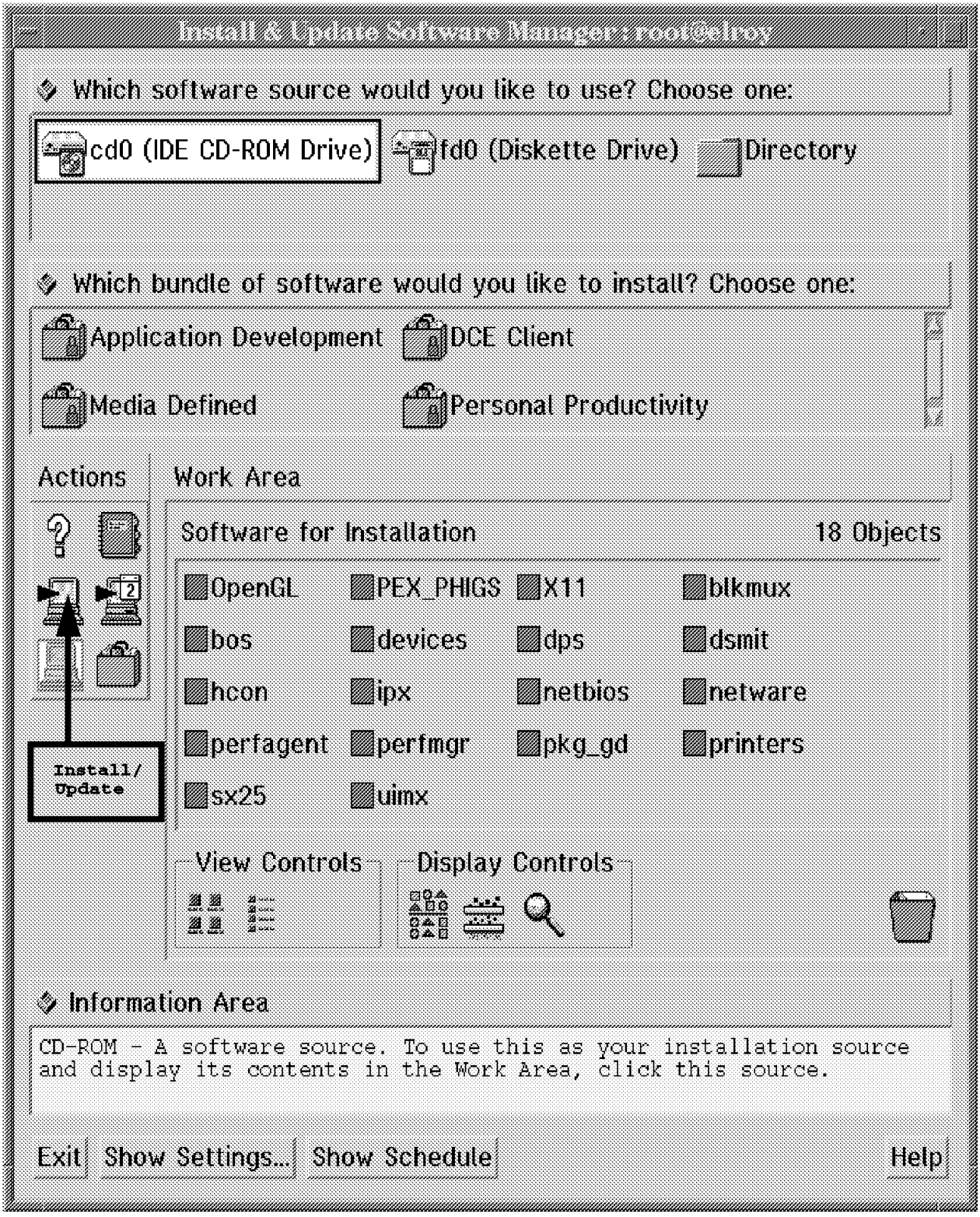


Figure 148. GUI Version - Install/Update Menu

When AIX has been updated, you have to be sure that the filesets for the 604 SMP are at the required levels or higher. In order to check this, you can use the following command, and look at the levels of the filesets listed in the section 8.2.1, “What is AIX V4.1.4 with 604 SMP Updates?” on page 216.

```
# oslevel -g
```

```
# oslevel -g
Fileset                                  Actual Level        Maintenance Level
-----------------------------------------------------------------------------
bos.diag.rte                             4.1.4.4             4.1.4.0
bos.net.tcp.client                       4.1.4.12            4.1.4.0
bos.rte.install                          4.1.4.1             4.1.4.0
bos.rte.libc                             4.1.4.11            4.1.4.0
bos.rte.libcfg                           4.1.4.1             4.1.4.0
bos.rte.libs                             4.1.4.9             4.1.4.0
bos.rte.mp                               4.1.4.12            4.1.4.0
bos.rte.security                         4.1.4.2             4.1.4.0
bos.sysmgt.serv_aid                      4.1.4.5             4.1.4.0
devices.mca.edd0.com                     4.1.4.7             4.1.4.0
devices.scsi.disk.diag.com               4.1.4.1             4.1.4.0
devices.scsi.disk.diag.rte               4.1.4.2             4.1.4.0
```

or you can use:

```
# lslpp -L | grep bos.rte.mp
```

```
# lslpp -L | grep bos.rte.mp
  bos.rte.mp                4.1.4.12    C    Base Operating System
```

If the filesets are at the correct level, or higher, as shown above, you are ready to perform a 601 to 604 hardware upgrade. So, please refer to 4.8, “Upgrading G30, J30 and R30/R3U from 601 to 604 Processors” on page 99 for more information.

8.2.3 AIX V4.1.4 Packaging

AIX V4.1.4 is proposed in five packages:

1. AIX Version 4.1.4 for Clients
2. AIX Connections Version 4.1.4
3. AIX Version 4.1.4 for Servers, 1-16 Users
4. AIX Version 4.1.4 for Servers, Unlimited Users
5. AIX Version 4.1.4 for Servers, 1-2 Users

The actual contents of the five packages are slightly different, the client package being a subset of the server. Although AIX V4.1.4 for Clients can be ordered on any system, most of the SMP servers will be used as multiuser servers; so AIX Version 4.1.4 for Servers or later version is likely to be ordered for these servers.

8.2.4 AIX V4.1.4 Bundles

The AIX V4.1.4 supplied bundle files are stored in the /usr/sys/inst.data/sys_bundles directory, and the files are:

```
# ls /usr/sys/inst.data/sys_bundles
ASCII.autoi        Client.bnd         GOS.autoi          Server.bnd
App-Dev.bnd        Client.def         Hdwr-Diag.def      Server.def
App-Dev.def        DCE-Client.bnd     Media-Defined.bnd
BOS.autoi          DCE-Client.def     Pers-Prod.bnd
```

Additional bundles can be created by the system administrator using the Install and Update Software Manager option of Visual System Management (VSM); these files are created in the /usr/sys/inst.data/user_bundles directory.

Client

This bundle includes a set of BOS packages deemed to provide the most common client functionality. This bundle has a base component (Client.bnd) and a graphical component.

DCE Client

This bundle contains the software required to be a client in a Distributed Computing Environment (DCE) network. This bundle has a base component (DCE-Client.bnd).

Server

This bundle installs packages and options that provide a more robust, full-function server. It essentially installs commonly used AIX server functionality as well as enhanced RAS (Reliability, Availability, Serviceability) functionality. This bundle also has a base component (Server.bnd) and a graphical component.

Personal Productivity

This bundle, Pers-Prod.bnd, installs packages and options that provide an enhanced *Personal Productivity* environment for AIX V4.1.4 users. It essentially includes the same type of functionality as the client bundle with the addition of CDE (COSE Desktop) and Common Open Software Environment (COSE) applets. This environment is only available if the X11 run-time is installed.

Application Development

This bundle App-Dev.bnd is essentially the same as the client with the addition of the filesets for application development and debugging.

Media Defined

It allows bundles to be defined by the installation media. This allows Independent Software Vendors (ISVs) to provide bundled installation of their software products.

Other

Update, Maintenance Level, and All can be used as *bundles* for installation.

8.2.5 AIX V4.1.4 Installation Methods

This section outlines the various means of installing AIX Version 4.1.4 on your SMP server. In AIX V4, you can install a system:

- By using the Network Installation Management (NIM)
- By using a CD-ROM or tape

Whichever method you choose, you also have the following options for the type of installation you wish to carry out:

- New or Complete Overwrite Installation
  This will completely overwrite an existing system or install a new system with AIX.
- Migration or Preservation Installation
  Both these options can be used to upgrade an existing AIX system. Please see the sections below for more information.
- Mksysb Installation
  A mksysb is an AIX system image backup. Installing using this method will completely overwrite the contents of the root volume group on the target system.

These options are outlined below:

New or Complete Overwrite Installation: This method should be used to install AIX V4 on a new system or on a system on which you want to overwrite the existing version of AIX.

The main advantage of this method is that your resulting AIX V4 installation is perfectly clean. On the other hand, the entire system will need to be configured or reconfigured if it was previously installed. The amount of work depends on the complexity of your installation; this work must be evaluated before choosing such a method.

User-defined volume groups (nonroot volume groups) will be preserved and will remain there after the new installation. You will have to import these volume groups and mount the file systems created on them.

Migration Installation
A migration install allows you to upgrade your previous version of the operating system to AIX V4.1.4, while keeping customized configuration information and optional, installed software.

Any configuration files that cannot be migrated are saved in a specific directory under /tmp. System messages will inform you of the location of the saved files. Information is also stored in the system installation log in /var/adm/ras/devinst.log.

The Migration Install will preserve all file systems including /, /usr, and /var.

A migration installation will preserve logical volumes (including dump and paging), system configuration files, user data, and all file systems. It removes all the files in the /tmp file system.

In order to perform a Migration Install, you have to boot the system to get to the Installation and Maintenance menu. The procedure is the same as described in section *"Installation Flow" on page 222*.

Following is a figure describing the main steps of a migration install.

Preservation Installation
With the Preservation Install method, the contents of /usr, /, /var and /tmp will be deleted.

On the other hand, the Preservation Install will save the previous paging space and dump device, /home, and other user-created file systems in rootvg.

User-defined volume groups (nonroot volume groups) will be preserved and activated automatically after the Preservation Install. Before mounting the file systems belonging to these user-defined volume groups, you might need to recreate the file systems' mounting directories that were removed during the installation (directories under / or /usr, for example).

Configuration files, with the exception of /etc/filesystems, will be deleted. If you want any additional configuration files to be saved during the Preservation Install, you must edit /etc/preserve.list file on your AIX V4.1 system, and add the full path names of the configuration files you want to save. The system needs sufficient disk space in the /tmp file system to store the files listed in the /etc/preserve.list file.

In order to perform a Preservation Install, you have to boot the system to get to the Installation and Maintenance menu. The procedure is the same as described in the section below titled "Installation Flow".

mksysb Installation
The mksysb method of installation restores a previous system image backup, ideally created on the same machine. This completely erases the contents of the root volume group. All configuration files will be restored to the time of the backup. This method will not necessarily overwrite any other volume groups in the system because you may specify the disk which you wish to overwrite. If you wish to restore a mksysb onto a different system from the one on which it was created, some additional steps are necessary. Please see 8.3.7, "Cloning" on page 257.

8.2.6 Installing or Overwriting an SMP Server

The intent of this section is to describe the main steps in performing a new installation or reinstallation.

Installation Flow: In order to boot a system to get to the Installation and Maintenance menu, the procedure is the same as for previous versions of AIX; that is, turn the key to the Service position and switch on the system with the installation media in the appropriate device, which will either be the tape drive or the CD-ROM drive.

Attention !

If your SMP system boots in diagnostics, but you wanted it to boot from media, there are two likely explanations:

- First, the Service mode bootlist is probably not set correctly. In order to change the Service mode bootlist, you can enter the following command at a shell prompt:

  ```
  # bootlist -m service <device> (for example, cd0)
  ```

  or you can follow these main steps in the diagnostics menus:

  ```
  Task Selection (Diagnostics, Advanced Diagnostics, Service Aids, etc.)
  -> Display or Change Bootlist
     -> Service mode bootlist
        -> Alter Current bootlist
  ```

- Second, if it still does not boot on the media, you have to ensure that the flag, called “Boot Multi-user AIX in Service”, is disabled. In order to disable it, you can follow these main steps in the diagnostics menus :

  ```
  Task Selection (Diagnostics, Advanced Diagnostics, Service Aids, etc.)
  -> Display or Change BUMP Configuration
     -> Display or Change Flags and Configuration
        -> Change Diagnostic Flags
  ```

  You can refer to *Chapter 5, “SystemGuard” on page 105* for more information about this flag.

Note: Remember that a diskette can no longer be used as boot media for AIX V4. Only CD-ROM, tape, and Network adapters can be used for BOS install.

The initial question that will be asked is to identify the console device. In addition, you have to choose a language from a selection of the eight European languages, which in our case was English. Bear in mind that this language selection is for the installation process only. The Primary Language (if you require it to be different) will be offered later. At this point, the Installation and Maintenance menu appears, and you can now choose the type of installation that you require.

Hint

You can choose the option **Start Install Now with Default Settings** if you are confident it will give you what you want. We generally found it better to choose the option **Change/Show Installation Settings** rather than Install Now with Default Setting because the default settings can vary, depending on the level of AIX you are coming from. Choosing **Change/Show Installation Settings** removes the likelihood of any surprises

The following diagram shows the steps for installing AIX V4.

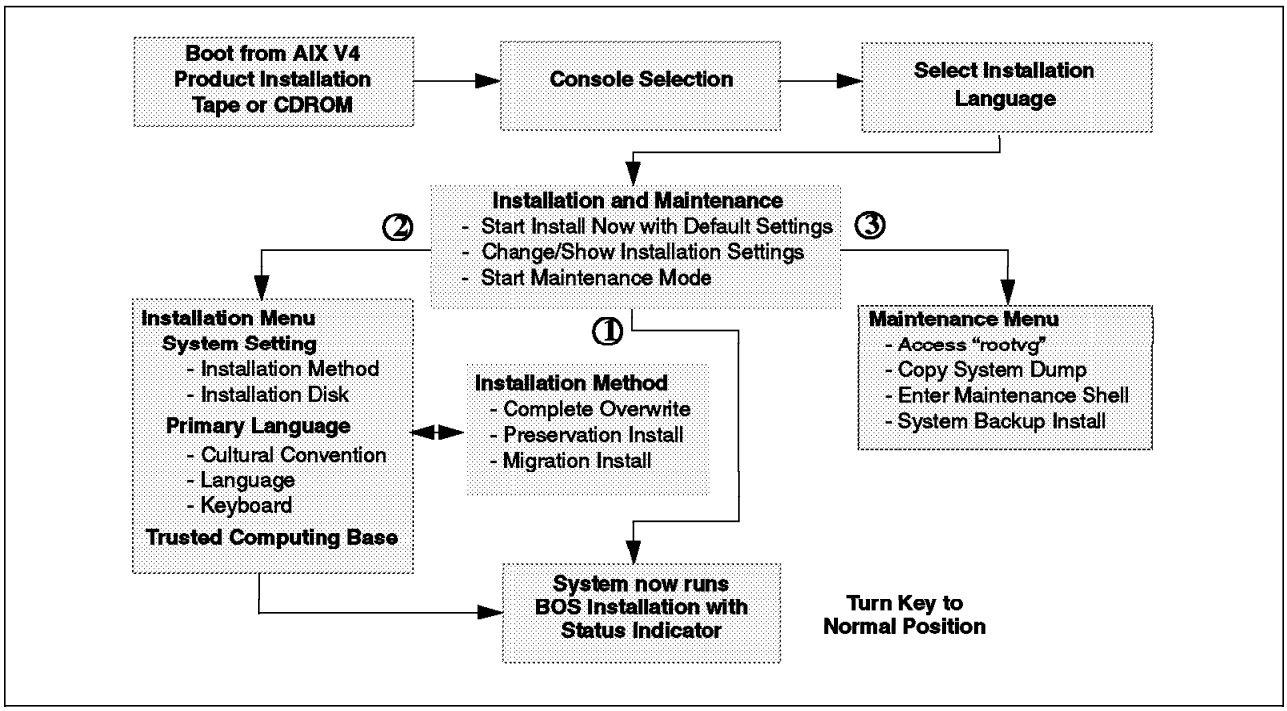


Figure 149. AIX V4 Installation Flow

The system will install at this point. A status indicator will appear on the screen that describes the progress of the installation process, and the indicator will display the percentage of tasks completed and the elapsed time (in minutes).

```
    Approximate            Elapsed time
% tasks complete           (in minutes)


        57                      10          < ... Status Message ... >
```

The key should now be turned to the Normal position so that the system will reboot without intervention when the BOS installation has completed. Changing the key to Normal can take place at any time during the installation phase before the reboot, but it makes sense to do it at this point.

The following are some of the tasks you might have to perform. These tasks depend on your specific environment.

- Set the system date and time for your time zone.
- Set a root user account password to restrict access to system resources.
- Confirm or change the install device you want to use for installing additional software. The device may be a CD-ROM, a tape drive, or a local or remote directory.
- Check the system storage and paging space needed for installing and using additional software applications.
- Set your National Language Support (NLS) environment.
- Import user-defined volume groups.
- Create user accounts and passwords.
- Install third-party device drivers (for example, LAN printer server)
- Set your system network configuration (if applicable).
- Create local and remote terminals.

- Configure local and remote printers.
- Install and configure additional LPPs.
- Install and configure third-party software.

Installation assistant: Most of the tasks listed above can be obtained with the Installation Assistant. The Installation Assistant will appear after you enter a name to acknowledge the Software License, and it will appear for all types of AIX V4 installation (except mksysb) unless you are using a customized bosinst.data file from diskette or with NIM for network installation. The Installation Assistant can also be brought up after installation, should you require it, by using either:

```
# smitty assist
```

for the ASCII version or

```
# install_assist &
```

for the GUI version, as shown in Figure 151 on page 226

```
                        Installation Assistant

  Move cursor to desired item and press Enter.

     Set Date and Time
     Set root Password
     Set Installation Device
     Configure Network Communications
     Manage System Storage and Paging Space (rootvg)
     Manage Language Environment
     Create Users
     Define Printers
     Import Existing Volume Groups
     Install Software Applications
     Back Up the System
     Tasks Completed - Exit to AIX Login


F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 150. ASCII Installation Assistant

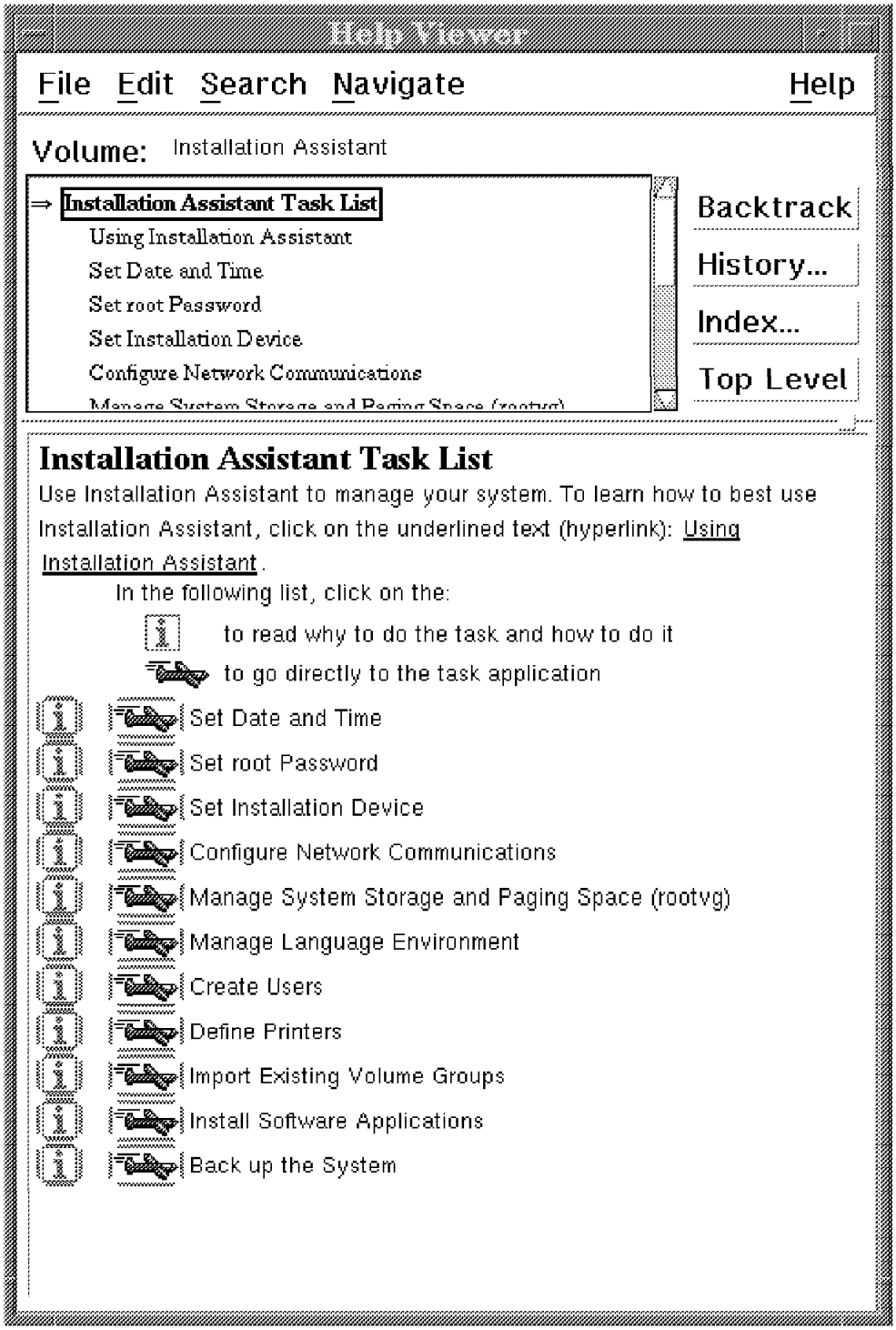


Figure 151. GUI Installation Assistant

Note: After you have used the desired options, always select the **Tasks Completed - Exit to AIX Login** so that the Installation Assistant entry is removed from the /etc/inittab file. Otherwise, the Installation Assistant will appear every time the system is booted.

Paging and Dump Devices: During a New and Overwrite Installation, a paging space (/dev/hd6) of 32 MB will be created, and this will be used for dumps as well. If you do not increase the size of the paging space using the Installation

Assistant, the paging space will be increased in size according to the AIX V3.2 defaults (for systems less than 64 MB, paging space = 2 x real memory; for systems with 64 MB or more, paging space = real + 16 MB).

If the default dump device is used, and a system crash occurs, the dump image is copied from the paging space to the /var file system on reboot. If the system has a lot of memory, there may not be enough space in this file system for the dump image. The copy will fail, and the system will prompt the user/administrator to select a media device to copy the dump image, or type 88 to exit and continue the reboot. This is obviously not acceptable if the system is unattended or in a remote location. The system administrator can choose to discard any dump images by setting:

```
# sysdumpdev -d /var/adm/ras
```

and continue to use the paging space (/dev/hd6) as the dump device.

The best solution for a large, multiuser server is to use a dedicated dump device so that the /var file system will not fill up with core files, and the dump image will be stored in the dump device. The drawback to this is that it does require dedicated disk space.

Check the estimated size of the dump image. This is given in bytes; so you need to work out the number of 4 MB partitions required for the dump device.

```
# sysdumpdev -e
```

Create the dedicated dump logical volume using the previous calculation for the number of partitions (for this example, 3):

```
# mklv -y hd7 -t sysdump rootvg 3
```

Set the primary dump device to the new dedicated dump device

```
# sysdumpdev -P -p /dev/hd7
```

and ensure that the system will automatically reboot after a crash, by checking the autorestart flag.

```
# lsattr -E -l sys0
```

If the autorestart flag is false, then change it to true

```
# chdev -l sys0 -a autorestart=true
```

If a dump occurs now, the dump image will be saved in the dump logical volume for later analysis, and the system will reboot immediately so that it is available again for the users.

Installation Messages: During installation, a number of messages are displayed on the console and can be viewed at a later time if the installation was unattended.

The BOS installation messages can be retrieved using:

```
# alog -o -t bosinst
```

The fileset installation information can be retrieved using:

```
# pg /var/adm/ras/devinst.log
```

Similarly, the messages that are displayed on the console while the system is booting can also be retrieved using:

```
# alog -o -t boot
```

bosinst.data and image.data Files: You have the option of customizing AIX installations and avoid having to answer prompts on your console when installing a system. This is useful to install another system once the first AIX V4 system has been installed. However, if you wish to clone that first installation to other systems, you should refer to 8.3.7, “Cloning” on page 257.

To customize an AIX V4 installation, you need to follow the steps detailed below:

1. Login as root and go to the / directory.

   ```
   # cd /
   ```

2. Copy the file /var/adm/ras/bosinst.data to /bosinst.data.

   ```
   # cp /var/adm/ras/bosinst.data .
   ```

3. Edit the variables in that file as per the comments in the file itself. Another reference is the *AIX Version 4.1 Installation Guide, SC23-2550* or InfoExplorer.

4. Create a file called signature that has a one-line entry with the word ″data″ in it.

   ```
   # echo data > signature
   ```

5. Back these two files up to a diskette:

   ```
   # ls ./signature ./bosinst.data | backup -iqvf/dev/rfd0
   ```

6. Put the install tape in the tape drive and the diskette in the diskette drive; put the key in the Service position, and switch the system on. It will install with the options that you have now specified in the bosinst.data file.

As an example, in order to install AIX V4 onto hdisk0 in a system, the following bosinst.data file was used:

```
control_flow:
    CONSOLE = /dev/tty0
    INSTALL_METHOD = overwrite
    PROMPT = no
    EXISTING_SYSTEM_OVERWRITE = yes
    INSTALL_X_IF_ADAPTER = yes
    RUN_STARTUP = yes
    RM_INST_ROOTS = no
    ERROR_EXIT =
    CUSTOMIZATION_FILE =
    TCB = no
    INSTALL_TYPE = full
    BUNDLES =

target_disk_data:
    LOCATION =
    SIZE_MB =
    HDISKNAME = hdisk0

locale:
    BOSINST_LANG = en_US
    CULTURAL_CONVENTION = en_US
    MESSAGES = en_US
    KEYBOARD = en_US
```

These procedures are described in detail in the *AIX Version 4 Installation Guide* in the chapter "Customizing the BOS Install Program".

You can also modify the image.data file that contains information that describes the image installed during the BOS installation process. This information includes the sizes, names, maps, and mount points of logical volumes and file systems in the root volume group. The installation program takes information from the image.data file regarding defaults for the machine being installed. See the *AIX Version 4.1 Files Reference, SC23-2512* for a description of the image.data file.

8.2.7 mksysb Installation

A system can be also installed by restoring a backup of a previously installed system. Figure 152 shows the installation flow from such a backup.

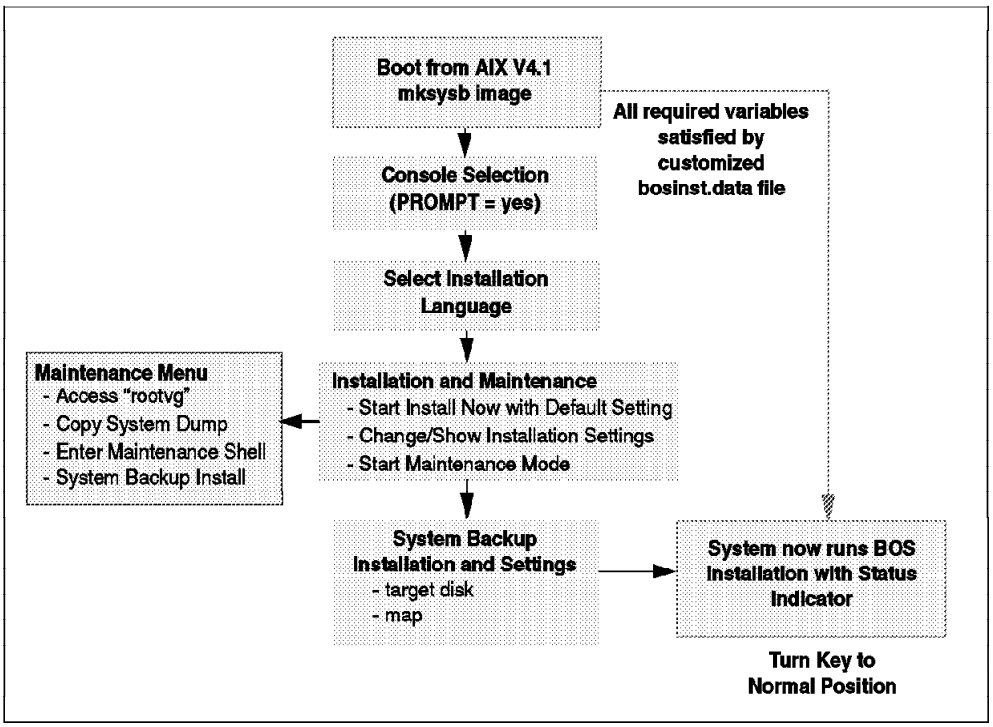


Figure 152. Installation Flow for mksysb

In order to create a mksysb, you can use the fast path `smitty mksysb`, which is the same as:

```
# smitty
System Storage Management (Physical & Logical Storage)
-> System Backup Manager
---> Backup the System
```

to get the following screen :

```
                                Back Up the System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                          (Entry Fields)
    WARNING:  Execution of the mksysb command will
              result in the loss of all material
              previously stored on the selected
              output medium. This command backs
              up only rootvg volume group.

* Backup DEVICE or FILE                              (/dev/rmt0)         +/
  Create MAP files?                                   no                 +
  EXCLUDE files?                                      no                 +
  Make BOOTABLE backup?                               yes                +
     (Applies only to tape)
  EXPAND /tmp if needed?                              no                 +
     (Applies only to bootable tape)
  Number of BLOCKS to write in a single output        ()                  #
     (Leave blank to use a system default)



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 153. mksysb Backup Screen

When installing a mksysb tape, the image.data file will indicate that BOS install should take the mksysb install path, and the user will not be prompted for installation method.

If the installation tape will not boot because of a corrupted boot image, then there will be a maintenance path provided with the product BOS install utilities which will allow the installation of the mksysb image.

Attention!

Since we are now dealing with minimum install environments in AIX V4.1.4, it is possible that a mksysb from one system will not install on another machine because of differing requirements in device support.

8.2.8 Network Installation

The New or Complete Overwrite Installation discussed previously can be also done through a network by using Network Installation Manager (NIM). The difference here is that a NIM server is used instead of an installation device (CD-ROM or tape drive).

In order to perform an installation via a Local Area Network (LAN), NIM must be installed and configured on a server running AIX V4. The installation resources, a spot and the AIX systems, will have to be defined as objects on the NIM server.

Installation resources have to be allocated to the system objects.

8.2.8.1 Example of an SMP Installation Using NIM

In this example, we use an RS/6000 Model 250 as a NIM master and server to install a J30* SMP server over a token-ring network.

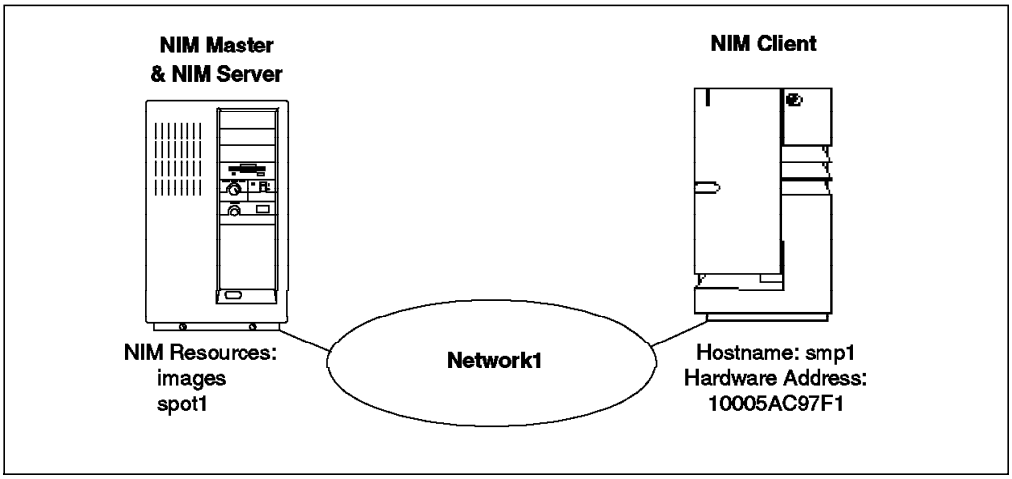


Figure 154. NIM Setup

1. Check the network communication. You have to make sure that the complete TCP/IP setup for your environment is done before you set up NIM. This includes definition of networks, initialization of gateways, name serving, and routing. All the host names for the systems participating in the NIM environment (master, servers, and all clients) must be resolvable. In our example, we use a domain name server.

   NFS must also run on the system planned to be the NIM master. If you are unsure about this, you can check with the command `lssrc -g nfs` for active subsystems, and use the fast path `smitty mknfs` to start NFS. We use the ASCII version of SMIT, called smitty, because it is faster.

   We do not go into more details about TCP/IP and NFS here because we assume that your network and servers are already set up since you were already using them before the installation.

2. Create file systems for some of the NIM resources for easier administration :

   - /export

     Use the /export file system for directories that contain the spot, the bosinst_data, and the installp_bundle resources. A size of 150 MB should be sufficient for one spot resource.

     The name /export is arbitrary; this means that you can use another file system name. However, by convention, the spot resource is located under /export/exec. If you use the name /export and you plan to serve the spot resource with NIM master, it is best to create the file system before the installation of the NIM filesets. During the installation of the bos.sysmgt.nim.master fileset, some subdirectories are created in the /export directory that you would have to recreate if you create the file system after the master is configured.

   - /tftpboot

     During the creation of the spot resource, the network boot images are generated and put into the /tftpboot directory. If you do not use an extra file system for this system, it is created in the / file system. NIM creates several network boot images for different system platforms (rs6k,

rs6ksmp or rspc) and for different network interfaces (token-ring, Ethernet, and FDDI). Together, they need about 20 MB of disk space, which would enlarge the / file system considerably.

Boot images are always created in the /tftpboot directory. If you create an extra file system for these images, the file system must be mounted on a directory called /tftpboot.

- /lpp_images

  This file system is created for the lpp_source resource. As all software that the clients need for installation is stored here, it can grow very big. The minimum is more than 300 MB, but it can grow to more than 1 GB. Therefore, you should consider creating an extra volume group for it.

  The name /lpp_images is arbitrary; this means that you can use another file system name.

To create the three file systems, you can specify a single Physical Partition (PP) as the size because NIM takes care of the necessary increase during the creation of the resources. This will avoid creating file systems that are bigger than necessary.

You can use the SMIT fast path `smitty crjfs` to create the file systems. You can also use the path:

```
# smitty
System Storage Management (Physical & Logical Storage)
-> File System
--> Add / Change / Show / Delete File Systems
---> Journaled File Systems
---> Add a Journaled File System
```

The following is a sample of a SMIT screen showing the creation of the /export file system in the rootvg:

```
                              Add a Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Volume group name                                    rootvg
* SIZE of file system (in 512-byte blocks)            [1]                        #
* MOUNT POINT                                         [/export]
  Mount AUTOMATICALLY at system restart?               yes +
  PERMISSIONS                                          read/write +
  Mount OPTIONS                                       [] +
  Start Disk Accounting?                               no +
  Fragment Size (bytes)                                4096 +
  Number of bytes per inode                            4096 +
  Compression algorithm                                no +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

You should make sure that the file systems are mounted automatically after system restart and that they are in read/write mode. You can also use the crjfs command directly from the command line with the correct options to create the file systems.

```
# crfs -v jfs -g rootvg -a size=1 -m /export -A yes
# mount /export
# crfs -v jfs -g rootvg -a size=1 -m /tftpboot -A yes
# mount /tftpboot
# crfs -v jfs -g rootvg -a size=1 -m /lpp_images -A yes
# mount /lpp_images
```

Please do not forget to mount the file systems after the creation. The output of your mount command should look similar to this:

```
# mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs    Sep 11 08:56 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs    Sep 11 08:56 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs    Sep 11 08:56 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs    Sep 11 08:56 rw,log=/dev/hd8
         /dev/hd1         /home            jfs    Sep 11 08:56 rw,log=/dev/hd8
         /dev/lv00        /export          jfs    Sep 11 08:56 rw,log=/dev/hd8
         /dev/lv01        /lpp_images      jfs    Sep 11 08:56 rw,log=/dev/hd8
         /dev/lv02        /tftpboot        jfs    Sep 11 08:56 rw,log=/dev/hd8
```

3. Install the NIM filesets on the NIM master. As we plan to use the NIM master also as the server, the following filesets must be installed :
   - bos.sysmgt.nim.master
   - bos.sysmgt.nim.spot

   You can use both the fastpath smitty install_latest or the SMIT menus to get the following SMIT screen:

```
                    Install Software Products at Latest Level

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* INPUT device / directory for software               /dev/cd0
* SOFTWARE to install                                 [4.1.4.0  Network Insta>
  PREVIEW only? (install operation will NOT occur)    no +
  COMMIT software updates?                            yes +
  SAVE replaced files?                                no +
  ALTERNATE save directory                            []
  AUTOMATICALLY install requisite software?           yes +
  EXTEND file systems if space needed?                yes +
  OVERWRITE same or newer versions?                   no +
  VERIFY install and check file sizes?                no +
  Include corresponding LANGUAGE filesets?            yes +
  DETAILED output?                                    yes +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The fileset bos.sysmgt.nim.client is installed automatically as a prerequisite.

Note: If you are working with CD-ROM, be sure to use an AIX V4.1.4 Server CD-ROM because the NIM master fileset, bos.sysmgt.nim.master, is not part of the Client CD-ROMs.

4. Configure the NIM master fileset. The system where the NIM master fileset is installed and configured becomes the unique NIM master system that controls the whole NIM environment. From this system, all the following setup and installation operations are initiated.

   With the fast path `smitty nim`, you can jump directly to the NIM main menu in SMIT; from there you can reach all the NIM menus.

```
                          Network Installation Management

Move cursor to desired item and press Enter.

  Configure Network Installation Management Master Fileset
  Manage Network Objects
  Manage Machine Objects
  Manage Resource Objects
  Create IPL ROM Emulation Media

F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

You can use the fast path smitty nimconfig, which is the same as the path:

```
# smitty
Software Installation and Maintenance
-> Network Installation Management
--> Configure Network Installation Management Master Fileset
```

to get to the following screen:

```
                   Configure Network Installation Management Master Fileset

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Network Object Name                                [Network1]
* Primary Network Install Interface                  [tr0] +
* Port Number for Network Install Communications     [1058] #
    Ring Speed                                       [16]
    Cable Type                                                    +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

If you prefer to work with the command line, you can also use the following command:

```
# nimconfig -a netname=Network1 -a pif_name=tr0
-a master_port=1058 -a ring_speed=16
```

Here, the network object, Network1, that represents the token-ring to which your systems are connected, is created. You have to specify a primary network install interface. This is the network interface the NIM master will use to communicate with the network. It must be defined because it could lead to confusion if the master had several interfaces and it was not clear which one should be used for the communication with the NIM environment.

The TCP/IP port number, 1058, is the default for the NIM network communication between the NIM master and its clients. You can change the port number in case of conflicts. Check the /etc/services file for potential conflicts before selecting a port number.

The ring speed must be specified for token-ring LANs. For Ethernet LANs, the cable type must be chosen instead.

At this point, to see the objects within the NIM database, you can issue the command:

```
# lsnim
```

To see more information about the NIM master, you can issue the command:

```
# lsnim -l master
```

```
 # lsnim -l master
master:
   class       = machines
   type        = master
   Cstate      = ready for a NIM operation
   reserved    = yes
   platform    = rs6k
   serves      = boot
   serves      = nim_script
   comments    = machine which controls the NIM environment
   Mstate      = currently running
   prev_state  =
   if1         = Network1 aixsrv1.itsc.austin.ibm.com 10005A4F195A
   master_port = 1058
   ring_speed1 = 16
```

Since the system is going to be a stand-alone system, in order to see what the required and optional resources are, enter:

```
# lsnim -q bos_inst -t standalone
```

```
# lsnim -q bos_inst -t standalone

the following resources are required:
        spot
        lpp_source

the following resources are optional:
        bosinst_data
        image_data
        installp_bundle
        mksysb
        script

the following attributes are optional:
        -a source=<value>
        -a no_nim_client=<value>
        -a installp_flags=<value>
        -a filesets=<value>
        -a preserve_res=<value>
        -a debug=<value>
        -a verbose=<value>
        -a force_push=<value>
        -a no_client_boot=<value>
        -a set_bootlist=<value>
        -a auto_expand=<value>
```

This tells that you need a SPOT and an lpp_source resource.

5. Define and create the NIM lpp_source resource (this can take up to an hour to complete). The lpp_source will be used as the source for the client installation. Also, NIM needs a set of required images (the support images) to define the directory /lpp_images as the lpp_source.

   Attention !

   Be sure to use your server product media for this step. Not all necessary software products may be included in the client media.

   You can use the fast path `smitty nim_mkres`, which is the same as the path:

   ```
   # smitty
   Software Installation and Maintenance
   -> Network Installation Management
   --> Manage Resource Objects
   ---> Define a Resource Object
   ```

   and choose the **lpp_source resource type** to get the following SMIT screen:

```
                                Define a Resource Object

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Resource Object Name                                 [images]
* Resource Type                                         lpp_source
* Server of Resource                                   [master] +
* Location of Resource                                 [/lpp_images] /
  Source of Install Images                             [/dev/cd0] +/
  Names of Option Packages                             []
  Comments                                             []

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

You can also use the following command :

```
# nim -o define -t lpp_source -a location=/lpp_images
-a server=master -a source=/dev/cd0 images
```

The object name, images, is arbitrary and can be chosen to describe the content of the resource. The server and the location of the resource specify the system and the directory where the resource is located.

If nothing is missing, the simages attribute will be set for the lpp_source. If you get errors, NIM will usually tell you which support images are missing. You can load missing support images into the directory /lpp_images by using bffcreate. When this is done, NIM must be told to check the lpp_source again for the missing support images by using the smitty nim_res_op fast path or the nim -o check images command.

To be sure that the simages attribute is set for your lpp_source, you can run the following command after the definition of the lpp_source or after the check operation :

```
# lsnim -l images
```

It gives you the following output:

```
# lsnim -l images
images:
   class       = resources
   type        = lpp_source
   location    = /lpp_images
   server      = master
   alloc_count = 0
   Rstate      = ready for use
   prev_state  = unavailable for use
   simages     = yes
```

The line

```
simages      = yes
```

is present if the simages attribute is set.

6. Define the SPOT resource. During the SPOT creation, the boot images for the network boot of the clients are generated.

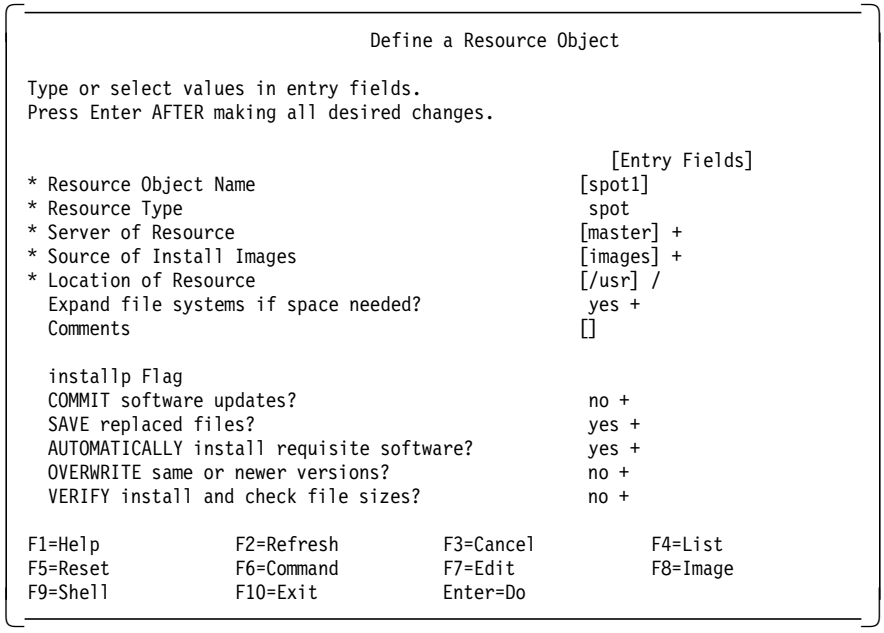
   On the server, you can either use the server's /usr file system as a SPOT resource, or you can create an extra directory or an extra file system. The advantage of using the server's /usr file system is that you save disk space on the server because files are not duplicated. The disadvantage is that you should no longer use the standard installp commands and interfaces (for example, VSM) to install software into /usr. You have to use NIM instead, and that is often more complex. If you use an extra directory or a file system, you do not have this restriction, but this will require more disk space.

   In our example, we chose to use the /usr file system as a SPOT, and we called it spot1.

   In order to define the SPOT resource, you can use the fast path smitty nim_mkres, which is the same as the path :

   ```
   # smitty
   Software Installation and Maintenance
   -> Network Installation Management
   --> Manage Resource Objects
   --> Define a Resource Object
   ```

   and choose **spot resource type** to get the following SMIT screen:



```
                                Define a Resource Object

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Resource Object Name                                 [spot1]
* Resource Type                                         spot
* Server of Resource                                   [master] +
* Source of Install Images                             [images] +
* Location of Resource                                 [/usr] /
  Expand file systems if space needed?                  yes +
  Comments                                             []

  installp Flag
  COMMIT software updates?                              no +
  SAVE replaced files?                                  yes +
  AUTOMATICALLY install requisite software?             yes +
  OVERWRITE same or newer versions?                     no +
  VERIFY install and check file sizes?                  no +

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

   The following command achieves the same thing:

   ```
   # nim -o define -t spot -a location=/usr -a server=master
   -a source=images -a auto_expand=yes spot1
   ```

   In our case, where the NIM master is also the server, we specify the lpp_source resource called images as the source of the install images. This is the fastest way to create the spot because only disk-to-disk operations are involved. You can also specify the CD-ROM or tape devices with appropriate product media loaded as sources, but this takes longer.

Again, it is important that we allow that the file systems can be expanded if space is needed because we only created the /export and /tftpboot file systems with a size of one PP, for example 4 MB.

If you want to check the state of the SPOT resource you created, you can use the command `lsnim -l <spot_name>`. For the SPOT named spot1, we get the following output:

```
# lsnim -l spot1
spot1:
   class        = resources
   type         = spot
   version      = 04
   location     = /usr
   server       = master
   alloc_count  = 0
   Rstate       = ready for use
   prev_state   = unavailable for use
   if_supported = rs6k ent
   release      = 01
   if_supported = rs6k fddi
   if_supported = rs6k tok
   if_supported = rs6ksmp ent
   if_supported = rs6ksmp tok
   if_supported = rspc ent
   if_supported = rspc tok
```

You should get a similar output for your SPOT resource.

7. Create a bosinst.data file. The NIM bosinst_data resource that is built from this file will be used to specify the installation parameters for the clients. Without this file, an unattended installation of the clients is not possible.

   In our example, we want to perform a force-push installation. Also, we have to create a bosinst.data file.

   Probably you will need several different bosinst.data files and resources for different client types.

   Therefore, we created the directory /export/bosinst_data as the repository for the bosinst.data file :

   ```
   # mkdir /export/bosinst_datas
   ```

   To this directory, we first copy the bosinst_data template, which is located in /var/adm/ras:

   ```
   # cp /var/adm/ras/bosinst.data /export/bosinst_datas/bosinst.data.inst
   ```

   Now, use an ASCII editor, such as vi, to edit the bosinst.data file. In the first part of the file, the different parameters and possible values are explained. You can use the following setup for your installation.

```
control_flow:
    CONSOLE = /dev/tty0
    INSTALL_METHOD = overwrite
    PROMPT = no
    EXISTING_SYSTEM_OVERWRITE = yes
    INSTALL_X_IF_ADAPTER = yes
    RUN_STARTUP = yes
    RM_INST_ROOTS = no
    ERROR_EXIT =
    CUSTOMIZATION_FILE =
    TCB = no
    INSTALL_TYPE = full
    BUNDLES =

target_disk_data:
    LOCATION =
    SIZE_MB =
    HDISKNAME = hdisk0

locale:
    BOSINST_LANG = en_US
    CULTURAL_CONVENTION = en_US
    MESSAGES = en_US
    KEYBOARD = en_US
¬
```

As the definition of the bosinst.data file is finished, you have to define the bosinst.data resource in NIM.

You can use the fast path `smitty nim mkres`, which is the same as the path:

```
# smitty
Software Installation and Maintenance
-> Network Installation Management
--> Manage Resource Objects
---> Define a Resource Object
```

and choose the **bosinst_data resource type** to get the following screen:

```
                                Define a Resource Object

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Resource Object Name                               [parametre]
* Resource Type                                       bosinst_dat
* Server of Resource                                 [master] +
* Location of Resource                               [/export/bosinst_data/>
  Comments                                           []









F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

You can also use the following command:

```
# nim -o define -t bosinst_data -a server=master
-a location=/export/bosinst_datas/bosinst.data.inst parametre
```

Again, the resource name is arbitrary and can be chosen to best describe the content of the resource.

The NIM master is now set up. Now, you need to begin the definition of the clients and the resources required to install them.

8. Define NIM clients. In our example, we have only one client, an SMP, hostname smp1. We have also chosen to attribute it the NIM object name of smp1.

   You can use the fast path `smitty nim_mkmac`, which is the same as the path:

   ```
   # smitty
   Software Installation and Maintenance
   -> Network Installation Management
   --> Manage Resource Objects
   ---> Define a Machine Object
   ```

   and choose the **hardware platform**, the **machine type** and the **primary network**, which are, in our example, **rs6ksmp**, **stand-alone** and **Network1** to get the SMIT screen:

```
                              Define a Machine Object

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Machine Object Name                                [smp1]
  Hardware Platform Type                              rs6ksmp
  Machine Object Type                                 standalone
  Primary Network Install Interface
    Network Object Name                               Network1
*   Host Name                                         [smp1]
    Network Adapter Hardware Address                  [0]
    Network Adaptor Logical Device Name               []
*   Ring Speed                                        [16] #
  IPL ROM Emulation Device                            [] /
  CPU Id                                              []
  Comments                                            []



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

   You can also use the following command :

   ```
   # nim -o define -t standalone -a platform=rs6ksmp
   -a if1='Network1 smp1 0' -a ring_speed1=16 smp1
   ```

   You must specify the platform type (rs6k, rs6ksmp, or rspc) and the machine type (can be diskless, dataless, or stand-alone). For more information about these platforms type, please refer to the section 8.1.2.3, “Platform Types” on page 208.

   You must also tell NIM the IP host name of the system. For easier administration, it is better to use the same name for the NIM object name and the IP host name. This is not necessary; the NIM object name is arbitrary and is only used by NIM.

   Note: The hardware address of the network adapter always needed to be specified before AIX V4.1.3. Starting with AIX V4.1.3, a “0” can be specified.

   The other details are either not needed, or are obtained later by NIM itself.

9. Prepare the client. In order to be able to push boot the SMP, one preparatory step must be done on it. It must have a .rhosts file in the / directory to allow the NIM master root access to the client.

   The content of the /.rhosts file should therefore look like this:

   ```
   <nim_master_hostname> root
   ```

   In our example, the /.rhosts file looks like this:

   ```
   aixsrv1 root
   ```

   The file can be copied to the client by using the File Transfer Protocol (FTP) or another mechanism.

   Attention !

   In order to perform a force-push installation, you need to create a /.rhosts file. This is the only way to perform such an installation if AIX is up and running on the target. Nevertheless, we want you to pay attention to the use of this file which is not secured. In case you want to use NIM to perform a migration, we recommend you to remove this file at the end of the migration.

   The key switch of the client must be set to Normal before the bos_inst operation is performed because of the force_push attribute we used.

10. Allocate the resources needed for installation to the client. The resources are: lpp_source, SPOT and bosinst_data.

    You can use the fast path `smitty nim_alloc`, which is the same as the path:

    ```
    # smitty
    Software Installation and Maintenance
    -> Network Installation Management
    --> Manage Machine Objects
    ---> Manage Network Install Resource Allocation
    ---> Allocate Network Install Resources
    ```

    and choose the **smp1 machine object** to get the SMIT screen:

```
                        Available Network Install Resources

Move cursor to desired item and press F7.
ONE OR MORE items can be selected.
Press Enter AFTER making all selections.

> images        lpp_source
> spot1         spot
> parametre     bosinst_data




F1=Help                 F2=Refresh              F3=Cancel
F7=Select               F8=Image                F10=Exit
Enter=Do                /=Find                  n=Find Next
```

    Use **F7** to select all the resources for the allocation.

You can also use the following command:

```
# nim -o allocate -a lpp_source=images -a spot=spot1
-a bosinst_data=parametre smp1
```

11. Start the bos_inst operation. The installation of BOS and additional LPPs on the client will take place in one step because of our preparation. All bos_inst operations are always started on the NIM master.

    To start the installation, you can use the fast path `smitty nim_mac_op`, which is the same as the path:

    ```
    # smitty
    Software Installation and Maintenance
    -> Network Installation Management
    --> Manage Machine Objects
    ---> Perform Operations on Machine Objects
    ```

    and choose the **smp1 machine object** and the **bos_inst operation** to get the following SMIT screen:

```
                        Perform a Network Install

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Machine Object Name                                 smp1
  Source for BOS Runtime Files                        rte +
  installp Flags                                     [-agX]
  Fileset Names                                      []
  Remain NIM client after install?                    yes +
  Initiate Boot Operation on Client?                  yes +
  Set Boot List if Boot not Initiated on Client?      no +
  Force Unattended Installation Enablement?           yes +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

    You can also use the following command :

    ```
    # nim -o bos_inst -F -a source=rte -a installp_flags=-agX
    -a force_push=yes smp1
    ```

    Specifying `rte` as the source for the BOS runtime files means that the lpp_source is used as the software source for the installation. This is the only possible choice.

    The specified flags for the `installp` command indicate that the software is applied with prerequisites and that file systems on the client can be extended if necessary.

    Setting the unattended installation enablement to *yes* means switching the force_push attribute to *yes*.

As a client can enable or disable the NIM master's push permissions, the flag -F in the NIM command overrides this restriction. So, if you want to be sure to perform a push on the client, you need to set this flag.

12. Customize your installation. In our example, we want to install a 604 SMP server via NIM. This means that we need 604 SMP updates filesets listed in the section8.2.1, "What is AIX V4.1.4 with 604 SMP Updates?" on page 216. As these updates are not in our AIX V4.1.4 CD-ROM, but in an other CD-ROM, we need to put the second CD-ROM, which contains these filesets, and define a new lpp_source for the customization. You can use the same method as described below to define and allocate the new lpp_source (with SMIT menus), or you can follow these steps:

```
# crfs -v jfs -g rootvg -a size= 1 -m /updt_images -A yes
# mount /updt_images
# nim -o define -t lpp_source -a location=/updt_images \
-a server=master -a source=/dev/cd0 updates
# nim -o allocate -a lpp_source=updates smp1
# nim -o cust -a fixes=update_all smp1
```

When the fixes attribute is set to the special keyword update_all, all software installed on the client will be updated with applicable updates from the installation media specified.

The above example was used to push a new installation from a server. For further information, please refer to the "*AIX Version 4.1 Network Installation Management Guide and Reference*".

8.3 Installing an SMP Server with AIX V4.2

AIX V4.2 represents a significant enhancement of capabilities to the AIX Version 4.1 operating system. While delivering many new features in a system designed for usability and growth, AIX V4.2 maintains a high degree of commitment to binary compatibility with AIX V4.1. AIX V4.2 provides new features in the following areas:

- Complex Application Support
- Standards Compliance
- Value Packaging
- Internet Implementation

AIX V4.2 has many additional performance enhancements in I/O, streams, networking, C language run-time, and so on.

For more details about AIX V4.2 additional features, please refer the IBM site:

`http://www.austin.ibm.com/software/OS/aix42.html`

You can also refer to this following site, which is only available for IBMers:

`http://w3.austin.ibm.com/afs/austin/depts/service/public_html/releases/`

Alternatively, you might want to refer to the redbook *"AIX 4.2 Differences Guide - SG24-4807"* for more detailed information on AIX V4.2.

8.3.1 AIX V4.2 Packaging

AIX V4.2 is proposed in seven packages:

1. AIX Version 4.2 for Entry Clients

   This is designed as a single-user desktop, commercial UNIX client on AIX and is restricted to systems that are in processor group D5.

2. AIX Version 4.2 for Workgroups

   In AIX V4.2, the AIX V4.1 Client package has been renamed AIX for Workgroups to reflect the fact that the package contains full NFS, NIS, and TCP/IP server support. It contains the functions and services contained in the Entry Client package plus:

   - SMP support
   - 1-2 users license
   - NFS Server, NIS Server, and TCP/IP Server functions
   - Microchannel device support

3. AIX Version 4.2 Connections Package

   This package integrates the AIX V4.2 for Workgroups package and the AIX V4.2 Connections Option.

4. AIX Version 4.2 for Entry Servers, 1-16 Users (available for Processor Group D5)

5. AIX Version 4.2 for Entry Servers, Unlimited Users (available for Processor Group D5)

6. AIX Version 4.2 for Advanced Servers, 1-2 Users (available for Processor Group G5 or lower)

7. Bonus Pack for AIX Package

   The Bonus Pack for AIX V4.2 is being introduced to provide a vehicle for the delivery of new and exciting software tools as part of the AIX V4.2 deliverable. The content of the Bonus Pack will vary depending on the specific AIX package ordered.

As in AIX V4.1.4, the AIX V4.2 for Advanced Servers is likely to be ordered for the SMP servers.

8.3.2 AIX V4.2 Bundles

The AIX V4.2-supplied bundle files are stored in the /usr/sys/inst.data/sys_bundles directory, and the files are:

```
# ls /usr/sys/inst.data/sys_bundles
ASCII.autoi           Client.def            Hdwr-Diag.def
App-Dev.bnd           DCE-Client.bnd        Media-Defined.bnd
App-Dev.def           DCE-Client.def        Pers-Prod.bnd
BOS.autoi             GOS.autoi             Server.bnd
Client.bnd            Graphics-Startup.bnd  Server.def
```

Additional bundles can be created by the system administrator using the Install and Update Software Manager (part of Visual System Management); these files are created in the /usr/sys/inst.data/user_bundles directory.

Client

This bundle includes a set of BOS packages deemed to provide the most common client functionality. This bundle has a base component (Client.bnd) and a graphical component.

DCE Client

This bundle contains the software required to be a client in a Distributed Computing Environment (DCE) network. This bundle has a base component (DCE-Client.bnd).

Server

This bundle installs packages and options that provide a more robust, full-function server. It essentially installs commonly used AIX server functionality as well as enhanced RAS (Reliability, Availability, Serviceability) functionality. This bundle also has a base component (Server.bnd) and a graphical component.

Personal Productivity

This bundle, Pers-Prod.bnd, installs packages and options that provide an enhanced *Personal Productivity* environment for AIX V4.1.4 users. It essentially includes the same type of functionality as the client bundle with the addition of CDE (COSE Desktop) and COSE applets. This environment is only available if the X11 runtime is installed.

Application Development

This bundle, App-Dev.bnd, is essentially the same as the client with the addition of the filesets for application development and debugging.

Media Defined

It allows bundles to be defined by the installation media. This allows Independent Software Vendors (ISVs) to provide bundled installation of their software products.

Graphics-Startup

This bundle installs all the basic software that is required for systems that have a graphics console attached to them.

Other

Update, Maintenance Level, and All can be used as *bundles* for installation.

8.3.3 AIX V4.2 Installation Methods

The installation methods for AIX V4.2 are very similar to those for AIX 4.1.4, please see *8.2.5, “AIX V4.1.4 Installation Methods” on page 221* for more details.

8.3.4 Installing or Overwriting an SMP Server

In order to perform a new installation, in case AIX was not preinstalled or if you want to reinstall your system, the method is the same as with AIX V4.1.4. You might want to refer to the section titled 8.2.6, “Installing or Overwriting an SMP Server” on page 222.

However, there is a new bosinst.data file with three new entries in the control flow stanza. It is located in the /var/adm/ras directory, and it looks like this:

```
control_flow:
    CONSOLE =
    INSTALL_METHOD = overwrite
    PROMPT = yes
    EXISTING_SYSTEM_OVERWRITE = yes
    INSTALL_X_IF_ADAPTER = yes
    RUN_STARTUP = yes
    RM_INST_ROOTS = no
    ERROR_EXIT =
    CUSTOMIZATION_FILE =
    TCB = no
    INSTALL_TYPE = full
    BUNDLES =
    SWITCH_TO_PRODUCT_TAPE =
    RECOVER_DEVICES =
    BOSINST_DEBUG =

target_disk_data:
    LOCATION = 00-07-A1-3,0
    SIZE_MB = 1003
    HDISKNAME = hdisk0

locale:
    BOSINST_LANG = en_US
    CULTURAL_CONVENTION = en_US
    MESSAGES = en_US
    KEYBOARD = en_US
```

8.3.5 mksysb Installation

In AIX V4.2, the mksysb installation method is not much different from the mksysb installation in AIX V4.1.4. However, there are some enhancements we want to highlight in this section.

In AIX V4.2, additional stanzas are included in the image.data file to add support to Logical Volume Striping and the Large File-Enabled Journaled file systems.

mksysb uses the dd command for the system boot image files and the backup command for the volume group images. Both commands are supported for files greater than 2 GB.

The mksysb command has been changed to include to new flags:

-v List files as they are backed up
-p Do not pack files as they are backed up

The -B option to create bootable tape has been removed. All system backup tapes created with mksysb in AIX V4.2 are bootable tapes.

There is also a new SMIT menu that can be obtained by two different ways:

```
# smitty
Software Installation and Maintenance
-> System Backup Manager
--> Back Up the System
```

or

```
# smitty
System Storage Management (Physical & Logical Storage)
-> System Backup Manager
--> Back Up the System
```

```
                              Back Up the System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
    WARNING:  Execution of the mksysb command will
              result in the loss of all material
              previously stored on the selected
              output medium. This command backs
              up only rootvg volume group.

* Backup DEVICE or FILE                              [/dev/rmt0]             +/
  Create MAP files?                                   no                      +
  EXCLUDE files?                                      no                      +
  List files as they are backed up?                   no                      +
  Generate new /image.data file?                      yes                     +
  EXPAND /tmp if needed?                              no                      +
  Disable software packing of backup?                 no                      +
  Number of BLOCKS to write in a single output        []                       #
     (Leave blank to use a system default)

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The mksysb command in AIX V4.2 allows you to create a system backup that can be installed on a target system that does not have the same hardware configuration as the source system. For example, the target system and the source system can have different architecture (uniprocessor or SMP). The device and kernel support required on the target system does not actually need to be installed on the source system. For example, if the target system requires device support not needed on the source system, this support could be copied from the product CD-ROM to the target system without ever being installed on the source system. This process is known as cloning, and is discussed in 8.3.7, “Cloning” on page 257.

The following are the practical limitations:

- The target system must have enough disk space to support the system backup image.
- The system backup image must support the target system processor and devices.
- Configuration information is not preserved if the hardware configuration is not the same.

8.3.6 Network Installation

In AIX V4.1, configuring a NIM environment is a time consuming and laborious process, requiring considerable expertise. This is true even for a simple network install environment, where the flexibility offered was not always needed.

In AIX V4.2, the new NIM Easy Startup facility hides much of the complexity of NIM configuration by providing a method for simple configuration of a sensible default environment. This allows a less experienced systems administrator to set up and configure a NIM environment more easily than before.

In order to show you the main NIM enhancements in AIX V4.2, we are going to use the same example as shown in section 8.2.8, “Network Installation” on page 230 for AIX V4.1.4.

In this example, we use an RS/6000 Model 250 as a NIM master and server to install a J30* SMP server over a token-ring network.

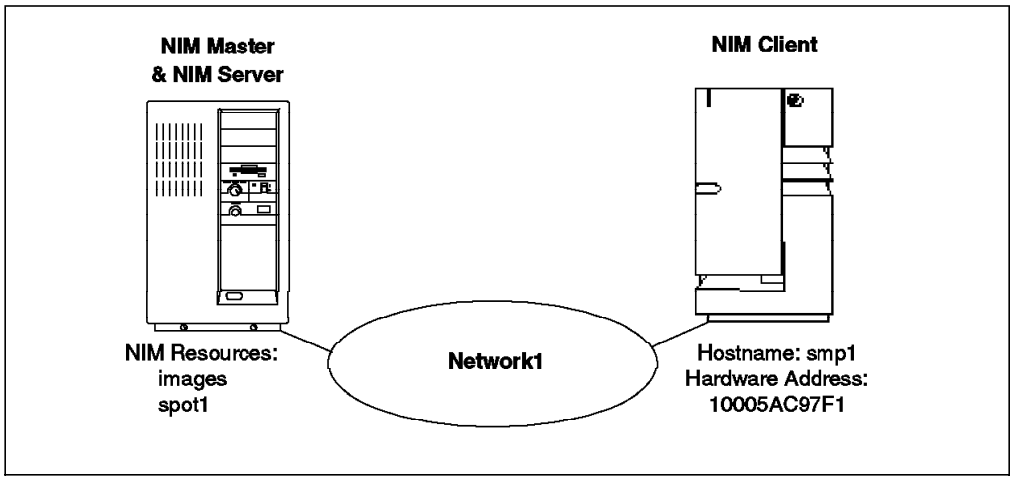


Figure 155. NIM Setup

1. Check the network communication. You have to make sure that the complete TCP/IP setup for your environment is done before you set up NIM. This includes definition of networks, initialization of gateways, name serving, and routing. All the host names for the systems participating in the NIM environment (master, servers, and all clients) must be resolvable. In our example, we use a domain name server.

   NFS must also run on the system planned to be the NIM master. If you are unsure about this, you can check with the command `lssrc -g nfs` for active subsystems, and use the fast path `smitty mknfs` to start NFS. We use the ASCII version of SMIT, called smitty, because it is faster.

   We do not go into more details about TCP/IP and NFS here because we assume that your network and servers are already set up since you were already using them before the installation.

2. Install the NIM filesets on the NIM master. As we plan to use the NIM master also as the server, the following filesets must be installed :

   - bos.sysmgt.nim.master
   - bos.sysmgt.nim.spot

   You can use both the fastpath `smitty install_latest` or the SMIT menus to get the following SMIT screen:

```
                        Install Software Products at Latest Level

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* INPUT device / directory for software                 /dev/cd0
* SOFTWARE to install                                   [4.2.0.0 Network Instal> +
  PREVIEW only? (install operation will NOT occur)      no +
  COMMIT software updates?                              yes +
  SAVE replaced files?                                  no +
  ALTERNATE save directory                              []
  AUTOMATICALLY install requisite software?             yes +
  EXTEND file systems if space needed?                  yes +
  OVERWRITE same or newer versions?                     no +
  VERIFY install and check file sizes?                  no +
  Include corresponding LANGUAGE filesets?              yes +
  DETAILED output?                                      yes +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The fileset bos.sysmgt.nim.client is installed automatically as a prerequisite.

Note: If you are working with CD-ROM, be sure to use an AIX V4.2.0 Server CD-ROM because the NIM master fileset bos.sysmgt.nim.master is not part of the Client CD-ROMs.

3. Set up the NIM environment. The new SMIT Easy Startup panel allows the systems administrator to setup a basic NIM environment by supplying a minimum of two pieces of information, namely:

   - Input device for installation images
   - Primary network interface

   You can use the fast path `smitty nim_config_env` method or the SMIT panels as follows:

   ```
   # smitty
   Software Installation and Maintenance
   -> Network Installation Management
   --> Configure the NIM Environment
   ---> Configure a Basic NIM Environment (Easy Startup)
   ```

   to get the following screen:

```
              Configure a Basic NIM Environment (Easy Startup)

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
  Initialize the NIM Master:
* Primary Network Interface for the NIM Master       [tr0] +

  Basic Installation Resources:
* Input device for installation images               [/dev/cd0] +
* LPP_SOURCE Name                                    [lpp_source1]
* LPP_SOURCE Directory                               [/export/lpp_source] +
    Create new filesystem for LPP_SOURCE?            [yes] +
    Filesystem SIZE (MB)                             [200] #
    VOLUME GROUP for new filesystem                  [rootvg] +
* SPOT Name                                          [spot1]
* SPOT Directory                                     [/export/spot] +
    Create new filesystem for SPOT?                  [yes] +
    Filesystem SIZE (MB)                             [200] #
    VOLUME GROUP for new filesystem                  [rootvg] +

  Create Diskless/Dataless Machine Resources?        [no] +
  Specify Resource Name to Define:
    ROOT   (required for diskless and dataless)      [root1]
    DUMP   (required for diskless and dataless)      [dump1]
    PAGING (required for diskless)                   [paging1]
    HOME        (optional)                           [home1]
    SHARED_HOME (optional)                           [shared_home1]
    TMP         (optional)                           [tmp1]
  Diskless/Dataless resource directory               [/export/dd_resource]
    Create new filesystem for resources?             [yes] +
    Filesystem SIZE (MB)                             [60] #
    VOLUME GROUP for new filesystem                  [rootvg] +

  Define NIM System Bundles?                         [yes] +

  Add Machines from a Definition File?               [no] +
  Specify Filename                                   []

* Remove all newly added NIM definitions            [no] +
  and filesystems if any part of this
  operation fails?

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Default values, which can be overridden, are provided for the remaining options.

On successful completion of this one SMIT panel, the following actions will have been completed:

- NIM master initialized on the primary network interface
- NIM daemons running
- lpp_source resource created and available
- SPOT resource created and available

When the user selects the default action of creating file systems for the lpp_source and SPOT resources, the new file systems are created with the specified sizes. During creation of the resources, the file systems will be expanded automatically as required for the resources to fit.

Creating a file system for each resource makes system storage management easier, particularly in environments where resources are added and removed frequently.

Setting up resources for diskless and dataless machines and configuring the list of NIM clients from a stanza file can also be carried out at the same time from this SMIT panel.

When you want to perform either of these tasks after you have already used the SMIT Easy Startup Panel, they should be performed from the SMIT Advanced Configuration Panel. This is because the SMIT Easy Startup Panel configures the NIM master, and a second attempt to configure the master will result in an error.

When the primary network is token-ring, and the machine that is to be the NIM master has a PCI bus token-ring card, ensure that the ring speed of the adapter is set to the correct value for the network. Although the PCI token-ring adapter supports *autosense* as a valid ring speed, NIM only recognizes 16 and 4 as valid ring speeds. If the ring speed is set to *autosense*, an error will occur when trying to configure the NIM master.

Each of the tasks featured in the SMIT Easy Startup panel can be performed individually from the SMIT Advanced Configuration panel.

4. Create a bosinst.data file. The NIM bosinst_data resource that is built from this file will be used to specify the installation parameters for the clients. Without this file, an unattended installation of the clients is not possible.

   In our example, we want to perform a force-push installation. Also, we have to create a bosinst.data file.

   You will probably need several different bosinst.data files and resources for different client types. Therefore, we recommend creating a directory, /export/bosinst_datas, as the repository for the bosinst.data file:

   ```
   # mkdir /export/bosinst_datas
   ```

   To this directory, we first copy the bosinst_data template that is located in /var/adm/ras:

   ```
   # cp /var/adm/ras/bosinst.data /export/bosinst_datas/bosinst.data.inst
   ```

   Now, use an ASCII editor, such as vi, to edit the bosinst.data file. In the first part of the file, the different parameters and possible values are explained. You can use the following setup for your installation.

```
control_flow:
    CONSOLE = tty0
    INSTALL_METHOD = overwrite
    PROMPT = no
    EXISTING_SYSTEM_OVERWRITE = yes
    INSTALL_X_IF_ADAPTER = yes
    RUN_STARTUP = yes
    RM_INST_ROOTS = no
    ERROR_EXIT =
    CUSTOMIZATION_FILE =
    TCB = no
    INSTALL_TYPE = full
    BUNDLES =
    SWITCH_TO_PRODUCT_TAPE =
    RECOVER_DEVICES =
    BOSINST_DEBUG =

target_disk_data:
    LOCATION =
    SIZE_MB =
    HDISKNAME = hdisk0

locale:
    BOSINST_LANG = en_US
    CULTURAL_CONVENTION = en_US
    MESSAGES = en_US
    KEYBOARD = en_US
```

Once the definition of the bosinst.data file is finished, you have to define the bosinst.data resource in NIM.

You can use the fast path smitty nim_mkres, which is the same as the path:

```
# smitty
Software Installation and Maintenance
-> Network Installation Management
--> Perform NIM Administration Tasks
---> Manage Resources
----> Define a Resource
```

and choose the **bosinst_data resource type** to get the following screen:

```
                              Define a Resource

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Resource Name                                      [parametre]
* Resource Type                                       bosinst_data
* Server of Resource                                 [master] +
* Location of Resource                               [/export/bosinst_datas>
  Comments                                           []









F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

This completes the NIM master setup. Now, you need to begin the definition of the clients and the resources required to install them.

5. Define NIM clients.

   In AIX Version 4.1, the procedure for defining a NIM client machine was as follows:

   a. Define the NIM network the client is connected to if it has not already been defined.
   b. Define the route between the client network and a network interface defined on the master if it has not already been defined.
   c. Define the client, which includes selecting the correct NIM network.

   This required the system administrator to use up to three SMIT panels to define one client machine.

   The process of defining a NIM client machine has been enhanced in AIX Version 4.2. One SMIT panel is used to define a NIM client machine. The initial dialogue prompts for the fully qualified hostname of the client machine. The NIM system resolves the name to an IP address that is used to determine whether a NIM network has been defined for the subnet the client is connected to. If the NIM network has been defined, the second dialogue panel of the SMIT screen will already contain details of the network.

   In our example, we have only one client, an SMP, whose hostname is smp1.

   You can use the fast path `smitty nim_mkmac`, which is the same as the path:

```
# smitty
Software Installation and Maintenance
-> Network Installation Management
--> Configure the NIM Environment
---> Advanced Configuration
---> Define NIM Client Machines
---> Add a NIM Client
```

   and enter the host name of the client, which is **smp1** in our example, to get the following screen:

```
                                Define a Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* NIM Machine Name                                   [smp1]
* Machine Type                                       [standalone] +
* Hardware Platform Type                             [rs6k] +
  Kernel to use for Network Boot                     [mp] +
  Primary Network Install Interface
*   Ring Speed                                       [16] #
*   NIM Network                                       network1
*   Host Name                                         smp1
    Network Adapter Hardware Address                 [0]
    Network Adapter Logical Device Name              []
  IPL ROM Emulation Device                           [] +/
  CPU Id                                             []
  Machine Group                                      [] +
  Comments                                           []

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

   In AIX Version 4.1, each NIM client machine had to be defined in the NIM environment in a separate operation. In a network with a large number of NIM client systems, this would be a very time-consuming and laborious task.

AIX Version 4.2 has added the facility to define multiple NIM client machines in a single operation. The nimdef command parses a definition stanza file to build the list of commands required to add NIM client definitions to the NIM environment. The nimdef command can be instructed to parse the definition file and report errors, invoke the commands it generates, or it can display the commands as output which can be redirected to a file and invoked at a later time.

6. Prepare the client. In order to be able to push boot the SMP, one preparatory step must be done on it. It must have a .rhosts file in the / directory to allow the NIM master root access to the client.

   The content of the /.rhosts file should therefore look like this:

   ```
   <nim_master_hostname> root
   ```

   In our example, the /.rhosts file looks like this:

   ```
   aixsrv1 root
   ```

   The file can be copied to the client by using the File Transfer Protocol (FTP) or another mechanism.

   Attention !

   In order to perform a force-push installation, you need to create an /.rhosts file. This is the only way to perform such an installation if AIX is up and running on the target. Nevertheless, we want you to pay attention to the use of this file which is not secured. In case you want to use NIM to perform a migration, we recommend you to remove this file at the end of the migration.

   The key switch of the client must be set to Normal before the bos_inst operation is performed because of the force_push attribute we used.

7. Install the Base Operating System. To start the installation, you can use the fast path smitty nim_task_inst, which is the same as the path:

   ```
   # smitty
   Software Installation and Maintenance
   -> Network Installation Management
   --> Perform NIM Software Installation and Maintenance Tasks
   ---> Install and Update Software
   ---> Install the Base Operating System on Standalone Clients
   ```

   and choose the **smp1 target**, the **rte installation type**, the **spot1 resource** and the **lpp_source1 resource** to get the following SMIT screen:

```
          Install the Base Operating System on Standalone Clients

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Installation Target                                  smp1
* Installation TYPE                                    rte
* SPOT                                                 spot1
* LPP_SOURCE                                           lpp_source1
  MKSYSB

  BOSINST_DATA to use during installation              [parametre] +
  IMAGE_DATA to use during installation                [] +
  RESOLV_CONF to use for network configuration         [] +
  Customization SCRIPT to run after installation       [] +

  Remain NIM client after install?                     [yes] +
  PRESERVE NIM definitions for resources on            [yes] +
  this target?

  FORCE PUSH the installation?                         [yes] +

  Initiate reboot and installation now?                [yes] +
    -OR-
  Set bootlist for installation at the                 [no] +
     next reboot?

  Additional BUNDLES to install                        [] +
    -OR-
  Additional FILESETS to install                       [] +
    (bundles will be ignored)

  installp Flags
    COMMIT software updates?                           [yes] +
    SAVE replaced files?                               [no] +
    AUTOMATICALLY install requisite software?          [yes] +
    EXTEND filesystems if space needed?                [yes] +
    OVERWRITE same or newer versions?                  [no] +
    VERIFY install and check file sizes?               [no] +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Specifying `rte` as the source for the BOS run-time files means that the lpp_source is used as the software source for the installation.

The specified flags for the `installp` command indicate that the software is applied with requisites, and the file systems on the client can be extended if necessary.

8.3.7 Cloning

AIX V4.1 has been packaged so that only the devices that are needed are actually installed on the system. This has made the cloning process difficult. In AIX V3.2, you were able to create a mksysb tape on one machine and install it on almost any other machine because most of the device drivers were in bos.obj.

In AIX V4.1, the devices were separated from the base in separate filesets, devices.* (for example, devices.scsi.disk). This, coupled with all the new hardware that has been released in the last three years (PCI-based systems, SMPs, graphics adapters, disks and so on), has made cloning machines almost impossible. Up to this point we have said that the only “supported” method of cloning was to install all the devices on a machine before you created a mksysb image. This is a waste of space because you do not need all the devices on that

machine and you do not know for sure which devices you will need until you are actually installing.

We have come up with a way to do this, but you still have to have a product media at the same level as your mksysb. This procedure is very important for not only cloning systems, but also for reproducing problems on different hardware.

The methods for cloning differ slightly, depending on whether you are using AIX V4.1 or AIX V4.2.

Attention!

A binary compatability problem was found with the 604 updates on 4.1.4.0. The problem occurs when the bos.rte.up fileset is installed at level 4.1.4.6 (or higher) and the bos.rte.libs fileset is installed with at least 4.1.4.5.

If you are installing a UP mksysb to an MP system with these levels, and you are using the base 4.1.4.0 media, then the bos.rte.mp fileset that will be installed by the cloning script will only be level 4.1.4.0. When the system reboots, you may not be able to login because of an error about “can’t set user credentials”.

To fix this problem, you will need to boot into maintenance mode from the 4.1.4.0 media, and update the bos.rte.mp fileset to at least the 4.1.4.6 level. The libs and kernel will then be in sync and you will be able to login.

The problem has been fixed in 4.1.5.0. If you are using the new 4.1.4.0+ media with the 604 updates, then this will not be a problem, because when the bos.rte.mp kernel fileset is installed, the updates on the media will also be installed so the kernels will match.

Thus, if you can check the level of the bos.rte.up kernel before you create your mksysb (if you are going to clone to an MP system) then you can install the MP kernel and its updates before you create your mksysb. If you are not in that position, you may just need to boot into maintenance mode after the install and install the mp kernel updates.

8.3.7.1 AIX V4.1 Cloning

Note

More information regarding the cloning procedure, and copies of all the scripts used, are available on the internet from the following IBM external URL:

`http://service.software.ibm.com/aix.us/go?/pdocs/os/clone.html`

For this procedure, you will need an AIX CD-ROM at the same version as the mksysb, the mksysb tape, and a blank AIX-formatted diskette.

You will need to create a customized diskette and carry out the following steps:

- Create a temporary directory, for example:

```
# mkdir /tmp/clone
# cd /tmp/clone
```

- Create a file called ./signature, containing the 4 characters ‘data’, such as:

  ```
  # echo data > signature
  ```

- Copy the master bosinst.data file to your temporary directory, for example:

  ```
  # cp /var/adm/ras/bosinst.data .
  ```

- Edit this file using a text editor, for instance:

  ```
  # vi bosinst.data
  ```

  Please refer to the section “Customizing the BOS Install Program” in InfoExplorer for more details about creating a bosinst.data file. You may want to customize it for your system to get a no-prompt install, in which case you will need to set more than is listed below (such as CONSOLE and PROMPT).

  Note: Make sure that the control_flow**:** stanza has this entry set (it is the most important part of the bosinst.data for this procedure):

  ```
  CUSTOMIZATION_FILE = cloner
  ```

  This tells the BOS install to run the cloner script after it has restored the mksysb image.

  Also, the target_disk_data stanza should be “zeroed” out as shown below so that the BOS install program will install on the best fit disk(s). Here is a sample bosinst.data file:

  ```
  control_flow:
      CONSOLE =
      INSTALL_METHOD = overwrite
      PROMPT = yes
      EXISTING_SYSTEM_OVERWRITE = yes
      INSTALL_X_IF_ADAPTER = yes
      RUN_STARTUP = yes
      RM_INST_ROOTS = no
      ERROR_EXIT =
      CUSTOMIZATION_FILE = cloner
      TCB = no
      INSTALL_TYPE =
      BUNDLES =
  target_disk_data:
      LOCATION =
      SIZE_MB =
      HDISKNAME =
  locale:
      BOSINST_LANG = C
      CULTURAL_CONVENTION = C
      MESSAGES = en_US
      KEYBOARD = en_US
  ```

- Create a customization script called cloner containing the following commands:

  ```
  #!/usr/bin/ksh
  set -x
  RV=`bootinfo -z`
  if [ "$RV" -eq 1 ]
  then
      ln -fs /usr/lib/boot/unix_mp /usr/lib/boot/unix
  fi
  if [ "$RV" -eq 0 ]
  then
  ```

```
        ln -fs /usr/lib/boot/unix_up /usr/lib/boot/unix
    fi
    BLVDISK=`lslv -l hd5 | grep hdisk | head -1 | cut -d' ` -f1`
    ln -f /dev/r$BLVDISK /dev/ipldevice
    bosboot -a -d /dev/ipldevice
    bootlist -m normal $BLVDISK
        rm -f /etc/firstboot
    sync
    sync
    sync
    exit 0
```

- Backup the three files (signature, cloner, and bosinst.data in our example) to a diskette, as follows:

  ```
  # find . -print | backup -ivqf/dev/rfd0
  ```

- Create an AIX V4.1.4 mksysb tape from the system that you want to clone. It can be a UP system that you want to clone to an MP system or an rs6k platform that you want to clone to an rspc platform. Any configuration should work.
- Boot up off the AIX V4.1.4 CD-ROM. Make sure the tape drive is turned on, but do not insert the tape yet as it might try to boot from the tape.
- Select the console: for example, on a tty, press **1** followed by **Enter**.
- From the next menu, select the installation language and press **Enter**, (for example, for English, press **1**).
- The maintenance menu will now be displayed on the console, select option **3** to **Start Maintenance for System Recovery** and press **Enter**. At this point, you should insert the AIX V4.1.4 mksysb backup tape into an appropriate tape drive.
- Select option **4** to **Install from a System Backup** and press **Enter**.
- A list of available tape drives will be displayed, select the tape drive you are using and press **Enter**.
- The system will access the tape drive and read the content of the diskette. Then the Welcome to Base Operating System Installation and Maintenance menu will come up. You will probably see an error on the next screen warning that the target disk cannot be found. This is normal, as the target SCSI address stored in the mksysb probably does not exist on the current system.
- Select option **2** to **Change/Show Installation Settings** and press **Enter**. Ensure that the target SCSI ID(s) indicated are the one(s) that you wish to overwrite. Once you are satisfied with the settings, press **0** followed by **Enter** to start the restore process.
- Turn the key to the Normal position.
- You should see the mksysb being restored, and then you should see installp called later when `cfgmgr` is called from the script to install any additional devices it detects.

Attention!

When installing a mksysb on a different platform type (that is, an rs6k mksysb to an rs6ksmp, or an rs6k to an rspc) the first bosboot that BOS install attempts will FAIL, and a message will be printed out to ask if you want to do maintenance or continue. You should continue at this point, and the cloner script will run although you will NOT see any output until the copyright screen at the end and the machine reboots.

This problem has been fixed in AIX V4.1.5.

Note

When cloning systems, please bear in mind that you will also clone IP addresses. If both servers are connected to the network at the same time, you will not be able to communicate with either of them.

8.3.7.2 AIX V4.2 Cloning

The procedure for cloning systems in AIX V4.2 has been greatly simplified. All you need is the AIX V4.2 mksysb tape that you wish to restore and the AIX V4.2 CD-ROM.

To clone an AIX V4.2 system, carry out the following steps:

- Boot up off the AIX V4.2 CD-ROM.
- Select the console: for example, on a tty, press **1** followed by **Enter**.
- From the next menu, select the installation language and press **Enter**, (for example, for English, press **1**).
- The maintenance menu will now be displayed on the console, select option **3** to **Start Maintenance for System Recovery** and press **Enter**. At this point, you should insert the AIX V4.2 mksysb backup tape into an appropriate drive.
- Select option **4** to **Install from a System Backup** and press **Enter**.
- A list of available tape drives will be displayed, select the tape drive you are using and press **Enter**.
- The system will now load the menus off the mksysb tape. You will probably see an error on the next screen warning that the target disk cannot be found. This is normal, as the target SCSI address stored in the mksysb probably does not exist on the current system.
- The standard mksysb menu (Welcome to Base Operating System Installation an Maintenance menu) should now be displayed. Select option **2** to **Change/Show Installation Settings** and press **Enter**. Ensure that the target SCSI ID(s) indicated are the one(s) that you wish to overwrite. Once you are satisfied with the settings, press **0** followed by **Enter** to start the restore process.
- Turn the key to the Normal position.

The system will now be restored from the mksysb tape. Once the restore has completed, any device drivers or other mandatory software (for example, bos.mp) will be restored off the CD-ROM. The machine will then reboot, and you will get a login prompt at on the console.

Note

When cloning systems, please bear in mind that you will also clone IP addresses. If both servers are connected to the network at the same time, you will not be able to communicate with either of them.

Chapter 9. UP to SMP Upgrade

This chapter covers the upgrade of a uniprocessor (UP) system running AIX V4.1.4 with 604 SMP Updates or later, to an SMP system. Its objective is to help you prepare an upgrade from a UP system to the SMP servers (G40, J40, R40). This chapter assumes that your UP System is already running AIX V4.1.4 with 604 SMP Updates or later. We shall reference these versions as SMP-supported AIX versions in the course of this chapter.

In this chapter we look into the various concerns involved in performing a UP to SMP upgrade. You can obtain more information on hardware upgrades in 4.9, “UP to 604 SMP Upgrade Paths” on page 103. Also, Chapter 8, “Installing an SMP Server” on page 203, covers the AIX V4 installation aspects.

If you are currently using AIX V3.2 and would like to do a UP to SMP upgrade, you will have to migrate AIX V3.2 to AIX V4.1.4 with 604 SMP Updates or later, first before you can do the upgrade. You might need to refer to the redbook entitled *A Holistic Approach to AIX V4 Migration,* SG24-4652, for more details on AIX V3.2.5 to AIX V4 migration methodology.

An upgrade from a uniprocessor to a multiprocessor is not just a case of loading a new operating system and testing the environment. The SMP technology is very different from the UP technology, and going from one to the other must be done carefully.

Because of the changes that have occurred between the two architectures, there are quite a few questions that have to be answered regarding the system environment before starting any migration to an SMP system.

- Has my application been certified to work with the SMP-supported AIX versions?
- If my application has been verified to work with these versions, is it SMP safe?
- Are my existing licensed products able to work with these versions of AIX?
- Must I have new license keys generated for my software?
- How does one change the kernel to take advantage of an SMP?
- Can I transfer the existing microchannel adapters and associated hardware to the SMP system?
- If the adapters and hardware cannot be transferred, what is available to take their place?
- Is there any third-party hardware installed in the UP system?
- If there is third-party hardware installed, is the hardware supported on the SMP system running the above SMP-supported versions of AIX, and do I need additional device drivers?

9.1 Related Publications

The following publications may be useful during the migration of a uniprocessor to a multiprocessor.

- *7012 G Series Operator Guide*, SA23-2740
- *7012 G Series Service Guide*, SA23-2741
- *7013 J Series Operator Guide*, SA23-2724
- *7013 J Series Service Guide*, SA23-2725
- *7015 Model R30 CPU Enclosure Operator Guide*, SA23-2742
- *7015 Model R30 CPU Enclosure Service Guide*, SA23-2743
- *7015 Model R00 Rack Installation and Service Guide R00*, SA23-2744
- *Supplemental Information for 7012 G Series Models, 7013 J Series Models, 7015 Models R30 and R40, Adapters, Devices and Cable Information, Diagnostics Information*, Part Number 40H7073

9.2 Planning the Upgrade

In order to help you in preparing your UP to SMP upgrade, you will find below a list of items that should be checked. This is not an exhaustive list and should be used as a guideline. The rest of this document assumes that your uniprocessor system is running SMP-supported AIX versions.

- Third-party applications:

  Check with the vendor of each third-party application that the application has been certified for the SMP-supported AIX versions and is SMP safe. If the application has not been certified to run on these versions, the upgrade cannot be performed. Also, if the application has been certified for these versions but is not SMP safe, the upgrade cannot be performed either.

- Custom-made applications:

  Check for any custom-made applications. These custom-made applications might have to be ported to SMP-supported AIX versions in order for them to run on an SMP system. If the custom-made applications have not, or cannot, be ported to these versions, the upgrade cannot be performed.

- LPP migration paths:

  Check the installed LPPs, and verify that they are available in the SMP-supported AIX versions. If yes, check the actual upgrade path to the respective SMP-supported version of each LPP (same or new product, free or fee upgrade). Check also to see if the equivalent LPP is SMP safe. If some required LPPs are not available in the SMP-supported AIX versions or they are not SMP safe, the upgrade cannot be performed. You might want to refer to the redbook *A Holistic Approach to AIX V4 Migration, Planning Guide,* SG24-4651 for more details.

- License keys:

  Check for any products that has an iFOR/LS or Net/LS license key. The CPU-ID will not be kept through the upgrade to the SMP; thus new keys for installed products requiring a license key will be needed. Check also to see if installed products that do not require license keys in AIX V4.1 will require license keys in SMP-supported AIX versions.

- IBM hardware compatibility:

  Check all installed adapters and I/O devices for compatibility with the SMP system. If the installed adapters and I/O devices are not compatible, have replacement adapters and I/O devices been ordered? If the replacement adapters and I/O devices have not been ordered and delivered, the upgrade cannot be performed.

  If the installed adapters and I/O devices are not compatible with the SMP system, and there are no substitute adapters and I/O devices, are you aware of this? Do you have an alternative to the incompatible adapters and I/O devices? If you do not have any alternatives to the incompatible adapters and I/O devices, then the upgrade cannot be performed.

- Third-party hardware compatibility:

  The list of third-party hardware vendors that manufacture hardware products for the RS/6000 is quite large; therefore IBM has no way of verifying what third-party hardware products will work with new versions of AIX and IBM RS/6000.

  If you have any third-party hardware installed in your system, IBM recommends that you contact the hardware manufacturer, and ask the manufacturer if his product has been tested to be compatible with the version of AIX that you are installing.

  If the hardware has been tested and certified with the version of AIX that you are installing, check if there are any additional device drivers for the hardware or if a generic device driver that is standard with AIX can be used.

  Customers or IBM employees that require information regarding the description of the hardware and/or physical addresses and phone numbers of third-party hardware vendors can obtain the information at the following locations:

  `http://www.developer.ibm.com/products/oemhw/oemhw_2.html`

  If the third-party hardware is not supported by a multiprocessor system, the hardware cannot be migrated to the multiprocessor system, and the migration cannot be performed until support is available or until the customer purchases supported hardware.

9.2.1 UP to SMP Upgrade Methods

Attention!

The information that follows in this chapter might change in the future. When performing such a UP to SMP upgrade, we strongly recommend that you use the procedures and tools that are shipped with the hardware upgrade. The purpose of this document is to help understanding issues related to UP to SMP upgrades, planning the upgrade and preparing appropriate test procedures.

There are two different recommended methods for upgrading from a UP to an SMP system, depending on the version of AIX that you are currently using on the UP. This is because of problem that exists in AIX V4.1.4 whereby its is possible that a system will not be able to boot after the upgrade. To resolve this issue, a slightly different upgrade method is used, as follows:

1. AIX V4.1.4 - The additional software and device drivers that are required for the SMP server are force installed, prior to restoring on the SMP.
2. AIX V4.1.5 and V4.2 - The additional software and device drivers required for the SMP server are automatically installed while restoring on the SMP.

These methods are discussed in the following pages.

9.2.2 Terminal and Printer Migration Issues

When a system is migrated from a uniprocessor to a multiprocessor, the tty and printer configurations will have to be re-created. This is because the system numbering (for example; sa0) of the asynchronous ports and adapters changes. The uniprocessor has two (2) serial ports, and the multiprocessor has three (3) serial ports, by default.

```
Uniprocessor

sa0        Available 00-00-S1     Standard I/O Serial Port 1
sa1        Available 00-00-S2     Standard I/O Serial Port 2

Multiprocessor

sa0        Available 00-00-S1     Standard I/O Serial Port 1
sa1        Available 00-00-S2     Standard I/O Serial Port 2
sa2        Available 00-00-S3     Standard I/O Serial Port 3
```

If the system has a 128-port adapter installed, the system numbering on a uniprocessor for the first 16-port Remote Asynchronous Node (RAN) EIA-232 for the 128-port adapter will be sa2, and on a multiprocessor, the system numbering will be sa3.

```
Uniprocessor

sa0        Available 00-00-S1     Standard I/O Serial Port 1
sa1        Available 00-00-S2     Standard I/O Serial Port 2
cxma0      Available 00-01        128-Port Asynchronous Adapter
sa2        Available 00-01-11     16-Port RAN EIA-232 for 128-Port Adapter

Multiprocessor

sa0        Available 00-00-S1     Standard I/O Serial Port 1
sa1        Available 00-00-S2     Standard I/O Serial Port 2
sa2        Available 00-00-S3     Standard I/O Serial Port 3
cxma0      Available 00-01        128-Port Asynchronous Adapter
sa3        Available 00-01-11     16-Port RAN EIA-232 for 128-Port Adapter
```

Hence, when the mksysb from the uniprocessor tries to restore the ttys and printers onto sa2 on the multiprocessor (trying to duplicate the uniprocessors configuration), the configuration methods routine will fail when the system tries to rebuild the Object Data Manager (ODM). This is because sa2 on the uniprocessor is the 16-port RAN for the 128-port adapter, and sa2 on the multiprocessor is the standard I/O serial port, 3.

All the ttys and printers that should be configured onto the 16-port RAN for the 128-port asynchronous adapter will be deleted and will not be configured onto the 16-port RAN for the 128-port asynchronous adapter.

When the multiprocessor is rebooted, none of the ttys or printers that should have been configured for the 128-port adapter will be available; they will have to be reconfigured.

If you use the AIX V4.1.4 upgrade method, there will be a file in /tmp that will enable you to recreate the printers and ttys. This file is:

- /tmp/reconfig1

If, however, you use the AIX V4.1.5 or V4.2 upgrade method, you will have to create this file, using the following command:

```
# /usr/lpp/bosinst/rda -R -d /tmp/bos/objrepos.inst/CuDv.sav -a
/tmp/bos/objrepos.inst/CuAt.sav -s /tmp/reconfig1
```

We explain in the upgrade procedure how to use this file to reconfigure the ttys and printers.

9.3 Upgrading AIX V4.1.4 UP to SMP

When upgrading a uniprocessor running AIX V4.1.4 to a 604 SMP, AIX needs first to be upgraded to AIX Version 4.1.4 plus 604 SMP Updates before any hardware changes can take place.

After installing the 604 SMP updates, the environment must be tested to be certain that the applications and AIX versions are in order before the upgrade can continue.

After the environment has been tested and you are satisfied that the site is functional on the AIX version, you must install the filesets and device drivers to enable the multiprocessor kernel and the new hardware to be utilized.

After the filesets and device drivers have been installed and the multiprocessor kernel enabled, you are ready to upgrade to a multiprocessor system. You will be able to back up the system, move the reusable hardware to the SMP, and restore the system backup.

9.3.1 Upgrade Steps

The main steps in the upgrade from a UP system to an SMP system are the following:

- Check that all the necessary resources you need during the upgrade are available prior to the upgrade.
- Document the configuration of your system.
- Back up your entire UP system, including rootvg, user-defined volume groups, and raw devices, if any.
- Use the mp_prep script to install all required filesets.
- Create a bootable backup tape.
- Migrate the reusable hardware.
- Restore the system backup on your SMP system.
- Restore user-defined volume groups and raw devices.
- Reconfigure asynchronous devices.

- Test the overall SMP installation.
- Back up the SMP system.

9.3.2 Checking Resources

Make sure that you have the following resources available to you before proceeding with the upgrade. If these resources are not available, you will not be able to complete the upgrade.

The required resources are as follows:

- AIX V4.1.4 with 604 SMP Updates (CD-ROM)
- Enough tape cartridges to complete three system backups.
- A supported ASCII terminal, (for example, IBM 3151)
- A half-height internal tape drive or an external tape drive and a CD-ROM drive
- The hardware upgrade kit (checked for completeness) that should have been delivered with your SMP upgrade including the Model Upgrade Utilities diskette containing the mp_prep shell script.
- A list of the additional hardware, if any, that has been ordered with the upgrade

  Note: Check that these additional features have been included in the upgrade kit. Depending on the procedures of the manufacturing plant that shipped this upgrade kit, some or all of these additional features may have been already installed within this upgrade. If they were factory installed, then a shipping list describing which ones were installed should have been included with the upgrade kit.
- Publications relating to the upgrade

You should also be aware that any currently installed features that are not supported on the SMP system cannot be moved to the upgraded machine.

Note that:

- The date and time will need to be reset upon completion of this upgrade.
- The system will be down for approximately six (6) to eight (8) hours.

9.3.3 Documenting the UP System

Before making any modifications to a system or server, it is essential to thoroughly document the current setup. You should ensure that you have the following items documented (printed) before proceeding with the upgrade:

- Hard disk layout, including the location of the Boot Logical Volume (blv)

  ```
  # lslv -l hd5
  ```
- Physical Volumes

  ```
  # lspv
  ```
- SCSI devices and adapters configuration

  ```
  # lsdev -Cs scsi
  ```
- Memory size and location

  ```
  # lscfg -vl mem*
  ```

- Adapter types and locations

  ```
  # lsdev -C -c adapter
  ```

- Asynchronous device names and locations (including ttys and printers)

  ```
  # lsdev -C | grep sa
  # lsdev -C | grep tty
  # lsdev -C | grep lp
  ```

- Installed LPPs, installation state and versions

  ```
  # lslpp -l
  ```

- Any other devices and/or software

More information concerning the required documentation for a UP to SMP migration can be obtained from the *Installation Instructions* document that is included in the SMP upgrade package.

9.3.4 Backing Up the UP System

At this step, you should back up your entire uniprocessor system and create a bootable tape. User-defined volume groups, raw disks, and raw Logical Volumes should be backed up as well.

For more information on how to back up your entire AIX V4 system, we recommend that you refer to the redbook entitled *A Holistic Approach to AIX V4 Migration, Volume 1,* SG24-4652.

9.3.5 Installing the Required Device Drivers and Filesets

The upgrade kit includes a Model Upgrade Utilities diskette containing a shell script called mp_prep that will automatically install the required filesets, create the MP kernel links, and the MP boot image. You will find the content of the mp_prep script at the end of this chapter.

Load the mp_prep script into a temporary directory, and execute it using the following syntax:

```
# ./mp_prep <installation_device>
```

Where `<installation_device>` is the name of the device that contains the AIX product media, for example, cd0.

9.3.6 Creating an MP Backup Tape

You are now ready to create a bootable backup tape of the modified rootvg.

Use the SMIT backup utility to achieve this.

```
# smitty mksysb
```

Note: At this step, it is not necessary to back up user-defined volume groups again.

9.3.7 Migrating the Hardware

The upgrade will swap the down-level chassis with the new chassis. All supported adapters, memory, and physical volumes will have to be moved to the new chassis.

Attention!

We recommend that you label all asynchronous cables and adapters before you disconnect them from the UP system. This will greatly help in reconnecting them on the SMP and in reconfiguring all the asynchronous devices (printers, ttys) after the system backup has been restored.

Each upgrade is specific. A detailed procedure on how to migrate the hardware will be provided with the upgrade kit. We recommend that you follow it carefully.

9.3.8 Restoring the System Backup on the SMP

After the hardware has been upgraded, it is necessary to recover rootvg and the user-defined volume groups and/or raw logical volumes.

The following section gives a procedure on how to accomplish this.

1. Check that the asynchronous printer and tty cables are correctly labeled and unplugged from the asynchronous adapters.

   Note: The reason these cables must be disconnected is that you do not want the asynchronous 16-port RANs to be configured after the system backup has been restored. This will enable you to restore the printers and ttys to their original adapters later in this procedure.

2. Plug the external options and devices into the power outlets.
3. Ensure the Main Power switch, which is located at rear of the system, is off (″0″ position). Plug the system unit power cord into the power outlet.
4. Turn on all external options and devices.
5. Set the key mode switch to the Service position (the wrench symbol).
6. Turn the system Main Power switch on. The words Stand-By should appear on the operator panel display on the front of the system.
7. Leave the system key in the Service position.
8. Push in the power button located on the operator panel.
9. Wait for the power on tests to complete.
10. The console will now display the Maintenance Menu.

```
                    MAINTENANCE MENU (Rev. 06.02)

  0> DISPLAY CONFIGURATION
  1> DISPLAY BUMP ERROR LOG
  2> ENABLE SERVICE CONSOLE
  3> DISABLE SERVICE CONSOLE
  4> RESET
  5> POWER OFF
  6> SYSTEM BOOT
  7> OFF-LINE TESTS
  8> SET PARAMETERS
  9> SET NATIONAL LANGUAGE









 SELECT :
```

11. Select option **6> System Boot** from the Maintenance Menu.
12. The System Boot menu will now be displayed.

```
                    SYSTEM BOOT

  0 > BOOT FROM LIST

  1 > BOOT FROM NETWORK

  2 > BOOT FROM SCSI DEVICE









SELECT [x:exit] :
```

13. We recommend that you put a product CD-ROM in the CD-ROM drive as the boot media. Do not insert the mksysb tape into the tape drive until after you have selected the tape option in the later part of the installation.
14. Select option **0> Boot from List**.

    The following will now be displayed on the console:

```
processor 0 on ipl start
{{243}}
{{243}}
<<299>>
```

    After approximately 10 minutes, you will be asked to define the system console by pressing **1**. You will then be asked to select the language that

you require. Next to be displayed on the console will be the Installation and Maintenance Menu.

```
               Welcome to Base Operating System
                 Installation and Maintenance

Type the number of your choice and press Enter. Choice is indicated by >>

  >>> 1 Start Install Now with Default Settings
      2 Change/Show Installation Settings and Install
      3 Start Maintenance Mode for System Recovery









  88 Help ?
  99 Previous Menu

 >>> Choice [1] :
```

Attention!

If your SMP system boots on diagnostics, but you wanted it to boot from a media, there should be two explanations:

- First, the Service mode bootlist is probably not set correctly. In order to change the Service mode bootlist, you can enter the following command at a shell prompt:

  ```
  # bootlist -m service <device> (for example, cd0)
  ```

  Or you can follow these main steps in the diagnostics menus:

  ```
  Task Selection (Diagnostics, Advanced Diagnostics, Service Aids, etc.)
  -> Display or Change Bootlist
  --> Service mode bootlist
  ---> Alter Current bootlist
  ```

- Second, if it still does not boot on the media, you have to ensure that the flag, called “Boot Multi-user AIX in Service”, is disabled. In order to disable it, you can follow these main steps in the diagnostics menus:

  ```
  Task Selection (Diagnostics, Advanced Diagnostics, Service Aids, etc.)
  -> Display or Change BUMP Configuration
  --> Display or Change Flags and Configuration
  ---> Change Diagnostic Flags
  ```

  You can refer to Chapter 5, “SystemGuard” on page 109 for more information about this flag.

15. Select option **3> Start Maintenance Mode for System Recovery**.
16. At the Maintenance Menu, select option **4> Install from a System Backup**.

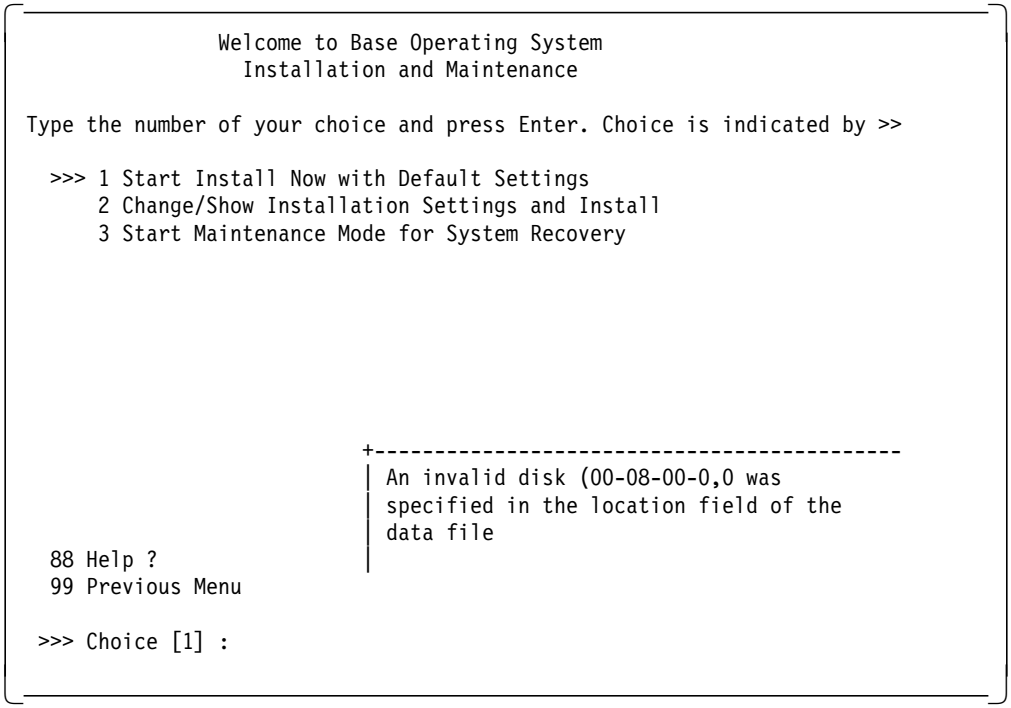
```
                    Maintenance

Type the number of your choice and press Enter.


>>> 1 Access a Root Volume Group
    2 Copy a System Dump to Removable Media
    3 Access Advanced Maintenance Functions
    4 Install from a System Backup









88 Help ?
99 Previous Menu

>>> Choice [1] :
```

Next, the Choose Tape Drive menu will be displayed.

```
              Choose Tape Drive

Type the number of the tape drive containing the system backup to be
installed and press Enter.

      Tape Drive                      Path Name
>>> 1 tape/scsi/8mm                  /dev/rmt0
    2 tape/scsi/525mb                 /dev/rmt1









88 Help ?
99 Previous Menu

>>> Choice [1] :
```

17. Select the tape drive that contains the mksysb.

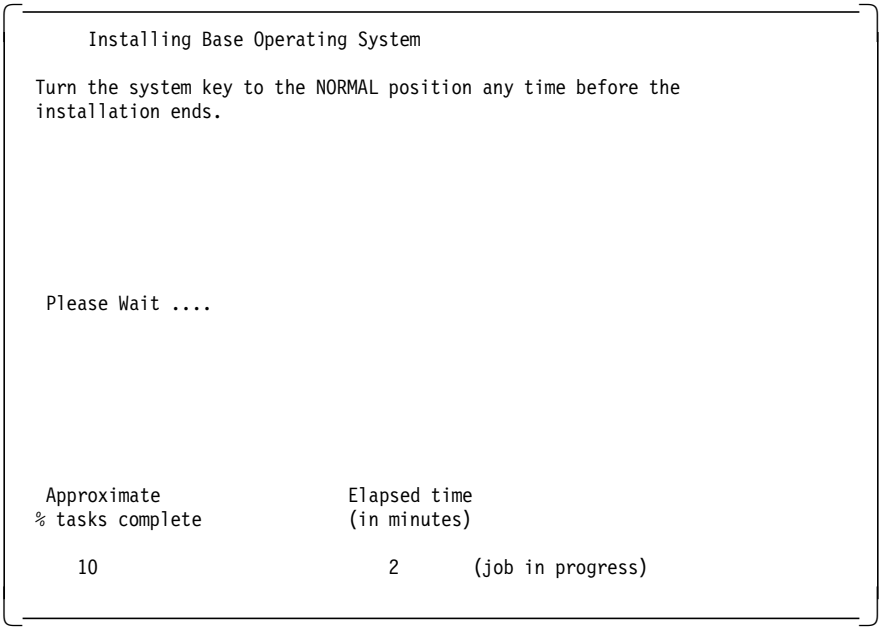
    After you have selected the tape drive, you can insert the mksysb tape into the drive at this point in time; the console will revert back to the Installation and Maintenance menu. An error may occur stating that there is an invalid disk being specified, you can safely ignore this message as the original mksysb configuration for hdisk0 address does not match with the new hardware configuration.



18. Select option **2 Change/Show Installation Settings and Install** on the Installation and Maintenance menu. Ensure that the target SCSI ID(s) indicated are the one(s) that you wish to overwrite.

    Once you are satisfied with the settings, press **0** followed by **Enter**.

    When the Installing Base Operating System screen is displayed on the console, your system is now busy restoring the mksysb.



    At this time, turn the system key from the SERVICE position to the NORMAL position.

19. Once the system has restored the backup, it will reboot and you will see the login prompt, log into the system as root.

20. Run diagnostics by entering:

    ```
    # diag
    ```

21. The Diagnostic Operating Instructions menu should be displayed.

    Note

    If the menu does not appear, run diagnostics from tape or CD-ROM. This will require rebooting the system in the Service mode.

    You may be required to initialize the terminal when running diagnostics.

22. Press the **Enter** key.
23. Select **Advanced Diagnostic Routines** on the Function Selection menu.
24. Select **System Verification** on the Diagnostic Mode Selection menu.

    Run diagnostics on the base system and all devices on the Advanced Diagnostic Selection menu.

25. After you have tested the system and all of the tests have passed, exit out of diagnostics (F10=EXIT).

9.4 Upgrading AIX V4.1.5 and V4.2 UP to SMP

The steps for upgrading a UP system running AIX V4.1.5 or AIX V4.2 to an SMP are very similar to the V4.1.4 procedure, except that instead of preloading the UP system with the SMP devices and software, these are loaded after the system has been restored onto the SMP server. This is achieved by using the cloning techniques outlined in 8.3.7.1, “AIX V4.1 Cloning” on page 258 for AIX V4.1.5 and 8.3.7.2, “AIX V4.2 Cloning” on page 261 for AIX V4.2.

Note

More information regarding the cloning procedure, and copies of all the scripts used, are available on the internet from the following IBM external URL:

`http://service.software.ibm.com/aix.us/go?/pdocs/os/clone.html`

9.4.1 Upgrade Steps

The main steps in the upgrading from a UP system to an SMP system are the following:

- Check that all the necessary resources you need during the upgrade are available prior to the upgrade.
- Document the configuration of your system.
- Back up your entire UP system, including rootvg, user-defined volume groups, and raw devices, if any.
- Create a bootable backup tape.
- Migrate the reusable hardware.
- Clone the system backup onto your SMP system.
- Restore user-defined volume groups and raw devices.
- Reconfigure asynchronous devices.

- Test the overall SMP installation.
- Back up the SMP system.

9.4.2 Checking Resources

Make sure that you have the following resources available to you before proceeding with the upgrade. If these resources are not available, you will not be able to complete the upgrade.

The required resources are as follows:

- An AIX CD-ROM at the same version as you UP system, that is, V4.1.5 or V4.2.
- Enough tape cartridges to complete three system backups.
- A supported ASCII terminal (for example, IBM 3151)
- A half-height internal tape drive or an external tape drive and a CD-ROM drive
- The hardware upgrade kit (checked for completeness) that should have been delivered with your SMP upgrade
- A list of the additional hardware, if any, that has been ordered with the upgrade

  Note: Check that these additional features have been included in the upgrade kit. Depending on the procedures of the manufacturing plant that shipped this upgrade kit, some or all of these additional features may have been already installed within this upgrade. If they were factory installed, then a shipping list describing which ones were installed should have been included with the upgrade kit.
- Publications relating to the upgrade

You should also be aware that any currently installed features that are not supported on the SMP system cannot be moved to the upgraded machine.

Note that:

- The date and time will need to be reset upon completion of this upgrade.
- The system will be down for approximately six (6) to eight (8) hours.

9.4.3 Documenting the UP System

Before making any modifications to a system or server, it is essential to thoroughly document the current setup. You should ensure that you have the following items documented before proceeding with the upgrade:

- Hard disk layout, including the location of the Boot Logical Volume (blv)

  ```
  # lslv -l hd5
  ```
- Physical Volumes

  ```
  # lspv
  ```
- SCSI devices and adapters configuration

  ```
  # lsdev -Cs scsi
  ```
- Memory size and location

  ```
  # lscfg -vl mem*
  ```

- Adapter types and locations

  ```
  # lsdev -C -c adapter
  ```

- Asynchronous device names and locations (including ttys and printers)

  ```
  # lsdev -C | grep sa
  # lsdev -C | grep tty
  # lsdev -C | grep lp
  ```

- Installed LPPs, installation state and versions

  ```
  # lslpp -l
  ```

- Any other devices and/or software

More information concerning the required documentation for a UP to SMP migration can be obtained from the *Installation Instructions* document that is included in the SMP upgrade package.

9.4.4 Backing Up the UP System

At this step, you must backup your entire system including rootvg, user-defined volume groups, raw disks, and raw Logical Volumes.

To create a bootable backup tape of rootvg, use the SMIT backup utility:

```
# smitty mksysb
```

For more information on how to back up your entire AIX V4 system, we recommend that you to refer to the redbook entitled *A Holistic Approach to AIX V4 Migration, Volume 1,* SG24-4652.

9.4.5 Migrating the Hardware

The upgrade will swap the down-level chassis with the new chassis. All supported adapters, memory, and physical volumes will have to be moved to the new chassis.

Attention!

We recommend that you label all asynchronous cables and adapters before you disconnect them from the UP system. This will greatly help in reconnecting them on the SMP and in reconfiguring all the asynchronous devices (printers, ttys) after the system backup has been restored.

Each upgrade is specific. A detailed procedure on how to migrate the hardware will be provided with the upgrade kit. We recommend that you follow it carefully.

9.4.6 Restoring the System Backup on the SMP

This procedure differs depending on the version of AIX that is running on the UP:

9.4.6.1 AIX V4.1.5

You will need an AIX formatted floppy disk for this procedure:

- Create a temporary directory, for example:

  ```
  # mkdir /tmp/clone
  # cd /tmp/clone
  ```

- Create a file called ./signature, containing the 4 characters ‘data’, such as:

```
# echo data > signature
```

- Copy the master bosinst.data file to your temporary directory, for example:

  ```
  # cp /var/adm/ras/bosinst.data .
  ```

- Edit this file using a text editor, for instance:

  ```
  # vi bosinst.data
  ```

  Please refer to the section “Customizing the BOS Install Program” in InfoExplorer for more details about creating a bosinst.data file. You may want to customize it for your system to get a no-prompt install, in which case you will need to set more than is listed below (such as CONSOLE and PROMPT).

  Note

  Make sure that the control_flow: stanza has this entry set (it is the most important part of the bosinst.data for this procedure):

  ```
  CUSTOMIZATION_FILE = cloner
  ```

  This tells the BOS install to run the cloner script after it has restored the mksysb image.

  Also, the target_disk_data stanza should be “zeroed” out as shown below so that the BOS install program will install on the best fit disk(s). Here is a sample bosinst.data file:

  ```
  control_flow:
      CONSOLE =
      INSTALL_METHOD = overwrite
      PROMPT = yes
      EXISTING_SYSTEM_OVERWRITE = yes
      INSTALL_X_IF_ADAPTER = yes
      RUN_STARTUP = yes
      RM_INST_ROOTS = no
      ERROR_EXIT =
      CUSTOMIZATION_FILE = cloner
      TCB = no
      INSTALL_TYPE =
      UNDLES =
  target_disk_data:
      LOCATION =
      SIZE_MB =
      HDISKNAME =
  locale:
      BOSINST_LANG = C
      CULTURAL_CONVENTION = C
      MESSAGES = en_US
      KEYBOARD = en_US
  ```

- Create a customization script called cloner containing the following commands:

  ```
  #!/usr/bin/ksh
  set -x
  RV=`bootinfo -z`
  if [ "$RV" -eq 1 ]
  then
      ln -fs /usr/lib/boot/unix_mp /usr/lib/boot/unix
  fi
  ```

```
if [ "$RV" -eq 0 ]
then
    ln -fs /usr/lib/boot/unix_up /usr/lib/boot/unix
fi
BLVDISK=`lslv -l hd5 | grep hdisk | head -1 | cut -d' ` -f1`
ln -f /dev/r$BLVDISK /dev/ipldevice
bosboot -a -d /dev/ipldevice
bootlist -m normal $BLVDISK
rm -f /etc/firstboot
sync
sync
sync
exit 0
```

- Backup the three files (signature, cloner, and bosinst.data in this example) to a diskette, as follows:

  ```
  # find . -print | backup -ivqf/dev/rfd0
  ```

- Insert the AIX V4.1.5 product CD-ROM in the tape drive, the diskette in the diskette drive and restore the AIX V4.1.5 system backup as described in 9.3.8, "Restoring the System Backup on the SMP" on page 270.

  Do not forget to check that all asynchronous cables are labeled and unplugged before starting the restore process.

- When the restore process is started, you can turn the key to Normal so the machine will reboot when the installation is complete.

- You should see the mksysb being restored, and then you should see `installp` called later when `cfgmgr` is called from the script to install any additional devices it detects.

- When the mksysb has been restored go to 9.5, "Post Upgrade Tasks" on page 280.

Note

When cloning systems, please bear in mind that you will also clone IP addresses. If both servers are connected to the network at the same time, you will not be able to communicate with either of them.

9.4.6.2 AIX V.2

The procedure in AIX V4.2 is simpler. All you need is the AIX V4.2 mksysb tape that you want to restore on the SMP and the AIX V4.2 CD-ROM.

In order to restore the AIX V4.2 mksysb tape to your SMP system, carry out the following steps:

- Insert the AIX V4.2 product CD-ROM and restore the AIX V4.2 system backup as described in 9.3.8, "Restoring the System Backup on the SMP" on page 270.

  Do not forget to check that all asynchronous cables are labeled and unplugged before starting the restore process.

- When the restore process is started, you can turn the key to the Normal position.

Once the restore has completed, any device drivers or other mandatory software (for example, bos.mp) will be restored off the CD-ROM. The machine will then reboot, and you will get a login prompt at the console.

- When the mksysb has been restored go to 9.5, “Post Upgrade Tasks.”

Note

When cloning systems, please bear in mind that you will also clone IP addresses. If both servers are connected to the network at the same time, you will not be able to communicate with either of them.

9.5 Post Upgrade Tasks

After completing the upgrade, you will need to reconfigure the printers and ttys. There are also some additional, optional tasks, that you may wish to perform at this time.

9.5.1 Recreating User-Defined Volume Groups

You are now ready to reload any user-defined volume groups and raw logical volumes from the UP system, using either or both of the following methods, depending on wether or not the physical volumes were migrated to the SMP server:

- If the physical volumes were migrated, the volume groups can be recreated by importing the volume groups, either using SMIT or the `importvg` command.
- If the physical volumes were not migrated, any user-defined volume groups and raw logical volumes will have to be recreated on the SMP and then reloaded from the backup tapes.

9.5.2 Reconfiguring TTYs and Printers

After the MP system backup has been restored, you must reconfigure the ttys and printers.

- Reconnect the cables for the asynchronous adapters, and switch on all the connected asynchronous nodes and devices.

  Note: Make sure that the cables are inserted into the correct location on the asynchronous adapters.

- At the command prompt, execute the following command:

  ```
  # cfgmgr -v
  ```

  Attention!

  We assume in the following procedure that the uniprocessor had an ASCII terminal connected and defined on S1 (tty0 on sa0).

  If the uniprocessor had a graphical display and no ASCII terminal configured on S1, the procedure is slightly different.

  This will configure all of the asynchronous nodes and assign sa numbers to them.

- If you used the AIX V4.1.4 upgrade method, there will be a file in /tmp that will enable you to recreate the printers and ttys. This file is:

/tmp/reconfig1

- If, however, you used the AIX V4.1.5 or V4.2 method of upgrading, you will have to create this file, using the following command:

  ```
  # /usr/lpp/bosinst/rda -R -d /tmp/bos/objrepos.inst/CuDv.sav -a
  /tmp/bos/objrepos.inst/CuAt.sav -s /tmp/reconfig1
  ```

- Copy /tmp/reconfig1 to /tmp/reconfig1.original:

  ```
  # cp /tmp/reconfig1 /tmp/reconfig1.original
  ```

  The reconfig1 file will look similar to the following:

  ```
  #!/bin/ksh
  export TALLY=
  cmd_1='/etc/methods/define -l tty1 -c tty -s rs232 -t tty -p sa2 -w 0 '
  cmd_2='/etc/methods/cfgtty -l tty1 '
  cmd_3='/etc/methods/define -l tty2 -c tty -s rs232 -t tty -p sa2 -w 1 '
  cmd_4='/etc/methods/cfgtty -l tty2 '
  cmd_5='/etc/methods/define -l tty3 -c tty -s rs232 -t tty -p sa2 -w 2 '
  cmd_6='/etc/methods/cfgtty -l tty3 '
  cmd_7='/etc/methods/define -l tty4 -c tty -s rs232 -t tty -p sa2 -w 3 '
  cmd_8='/etc/methods/cfgtty -l tty4 '
  cmd_9='/etc/methods/define -l tty5 -c tty -s rs232 -t tty -p sa2 -w 4 '
  cmd_10='/etc/methods/cfgtty -l tty5 '
  cmd_11='/etc/methods/define -l tty6 -c tty -s rs232 -t tty -p sa2 -w 5 '
  cmd_12='/etc/methods/cfgtty -l tty6 '
  cmd_13='/etc/methods/define -l tty7 -c tty -s rs232 -t tty -p sa2 -w 6 '
  cmd_14='/etc/methods/cfgtty -l tty7 '
  cmd_15='/etc/methods/define -l tty8 -c tty -s rs232 -t tty -p sa2 -w 7 '
  cmd_16='/etc/methods/cfgtty -l tty8 '
  cmd_17='/etc/methods/define -l tty9 -c tty -s rs232 -t tty -p sa2 -w 8 '
  cmd_18='/etc/methods/cfgtty -l tty9 '
  cmd_19='/etc/methods/define -l tty10 -c tty -s rs232 -t tty -p sa2 -w 9 '
  cmd_20='/etc/methods/cfgtty -l tty10 '
  cmd_21='/etc/methods/define -l tty11 -c tty -s rs232 -t tty -p sa2 -w 10 '
  cmd_22='/etc/methods/cfgtty -l tty11 '
  cmd_23='/etc/methods/define -l tty12 -c tty -s rs232 -t tty -p sa2 -w 11 '
  cmd_24='/etc/methods/cfgtty -l tty12 '
  cmd_25='/etc/methods/define -l tty13 -c tty -s rs232 -t tty -p sa2 -w 12 '
  cmd_26='/etc/methods/cfgtty -l tty13 '
  cmd_27='/etc/methods/define -l tty14 -c tty -s rs232 -t tty -p sa2 -w 13 '
  cmd_27='/etc/methods/cfgtty -l tty14 '
  cmd_29='/etc/methods/define -l tty15 -c tty -s rs232 -t tty -p sa2 -w 14 '
  cmd_30='/etc/methods/cfgtty -l tty15 '
  cmd_31='/etc/methods/define -l tty16 -c tty -s rs232 -t tty -p sa2 -w 15 '
  cmd_32='/etc/methods/cfgtty -l tty16 '
  cmd_33='chdev -l tty0
          -a login=enable
          -a term=ibm3151'
  cmd_34='chdev -l tty1
          -a login=enable
          -a term=ibm3151'
  cmd_35='chdev -l tty2
          -a login=enable
          -a term=ibm3151'
  cmd_36='chdev -l tty3
          -a login=enable
          -a term=ibm3151'
  cmd_37='chdev -l tty4
          -a login=enable
  ```

```
        -a term=ibm3151'
cmd_38='chdev -l tty5
        -a login=enable
        -a term=ibm3151'
cmd_39='chdev -l tty6
        -a login=enable
        -a term=ibm3151'
cmd_40='chdev -l tty7
        -a login=enable
        -a term=ibm3151'
cmd_41='chdev -l tty8
        -a login=enable
        -a term=ibm3151'
cmd_42='chdev -l tty9
        -a login=enable
        -a term=ibm3151'
cmd_43='chdev -l tty10
        -a login=enable
        -a term=ibm3151'
cmd_44='chdev -l tty11
        -a login=enable
        -a term=ibm3151'
cmd_45='chdev -l tty12
        -a login=enable
        -a term=ibm3151'
cmd_46='chdev -l tty13
        -a login=enable
        -a term=ibm3151'
cmd_47='chdev -l tty14
        -a login=enable
        -a term=ibm3151'
cmd_48='chdev -l tty15
        -a login=enable
        -a term=ibm3151'
cmd_49='chdev -l tty16
        -a login=enable
        -a term=ibm3151'
num_cmds=50
i=0
while [ $i -lt $num_cmds ]
do
    eval \$cmd_$i > /dev/null
    if [ $? -ne 0 ]
    then
        TALLY="$TALLY$(eval echo \$cmd_$i)\n"
    fi
    (( i+=1 ))
done
if [ -n "$TALLY" ]
then
    echo Machine not identical to previous configuration.
    echo The following items failed:
    echo $TALLY
fi
```

- Print the information regarding the asynchronous adapters numbering by executing the following command:

  ```
  # lsdev -C | grep sa | lp
  ```

The printout of the `lsdev -C | grep sa | lp` command is for example:

```
sa0        Available 00-00-S1     Standard I/O Serial Port 1
sa1        Available 00-00-S2     Standard I/O Serial Port 2
sa2        Available 00-00-S3     Standard I/O Serial Port 2
sa3        Available 00-04-21     16-Port RAN EIA-232 for 128-Port Adapter
```

- Using the `lsdev -C` output of the ttys, printers, and sa numbers that you recorded before the upgrade, map the sa numbers from the uniprocessor system to the sa numbers of the multiprocessor. Following is a diagram of a typical uniprocessor asynchronous port configuration that had its configuration mapped to a multiprocessor after the `cfgmgr` command was executed.

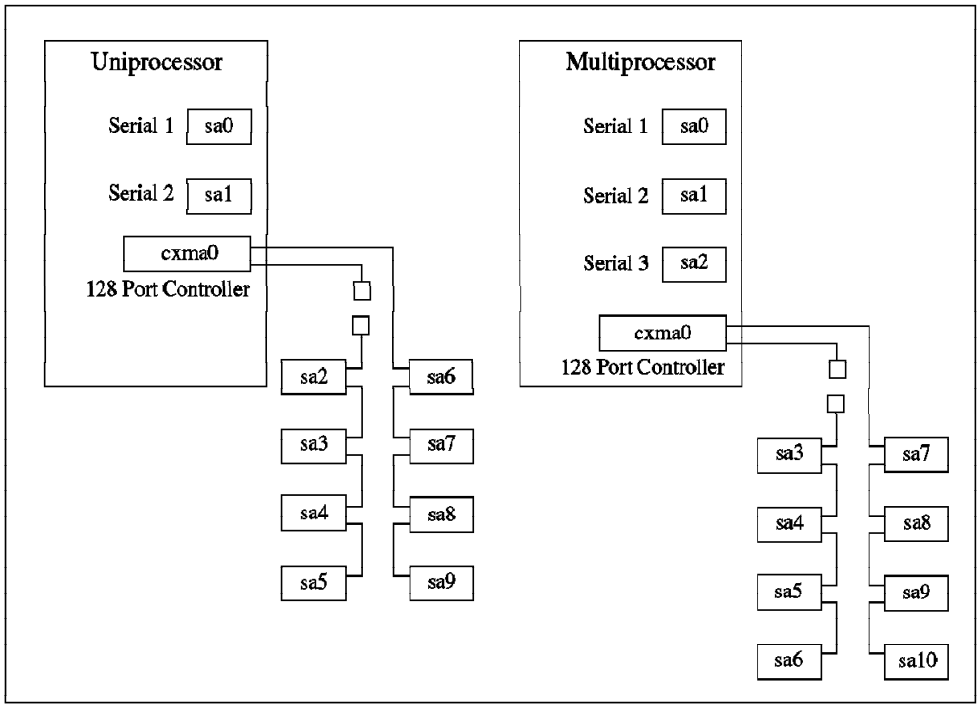


Figure 156. Asynchronous Ports Configuration

- Following is a sample table that maps the sa numbers, ttys, and printers from the uniprocessor configuration to the multiprocessor configuration.

Uniprocessor				Multiprocessor			
sa0	00-00-s1	tty0	00-00-s1-00	sa0	00-00-s1	tty0	00-00-s1-00
sa1	00-00-s2	lp0	00-00-s2-00	sa1	00-00-s2	lp0	00-00-s2-00
sa2	00-01-21	tty1	00-01-21-00	sa2	00-00-s3		
		tty2	00-01-21-01	sa3	00-04-21	tty1	00-04-21-00
		tty3	00-01-21-02			tty2	00-04-21-01
		tty4	00-01-21-03			tty3	00-04-21-02
		tty5	00-01-21-04			tty4	00-04-21-03
		tty6	00-01-21-05			tty5	00-04-21-04
		tty7	00-01-21-06			tty6	00-04-21-05
		tty8	00-01-21-07			tty7	00-04-21-06
		tty9	00-01-21-08			tty8	00-04-21-07
		tty10	00-01-21-09			tty9	00-04-21-08
		tty11	00-01-21-10			tty10	00-04-21-09
		tty12	00-01-21-11			tty11	00-04-21-10
		tty13	00-01-21-12			tty12	00-04-21-11
		tty14	00-01-21-13			tty13	00-04-21-12
		tty15	00-01-21-14			tty14	00-04-21-13
		tty16	00-01-21-15			tty15	00-04-21-14
						tty16	00-04-21-15

Figure 157. Asynchronous Ports Mapping

- Now that you have a map of the sa numbers, ttys, and printers. You must substitute the parameter -p sa2 to -p sa3. Edit the /tmp/reconfig1 file to look similar to the following:

```
/etc/methods/define -l tty1 -c tty -s rs232 -t tty -p sa3 -w 0
/etc/methods/cfgtty -l tty1
/etc/methods/define -l tty2 -c tty -s rs232 -t tty -p sa3 -w 1
/etc/methods/cfgtty -l tty2
/etc/methods/define -l tty3 -c tty -s rs232 -t tty -p sa3 -w 2
/etc/methods/cfgtty -l tty3
/etc/methods/define -l tty4 -c tty -s rs232 -t tty -p sa3 -w 3
/etc/methods/cfgtty -l tty4
/etc/methods/define -l tty5 -c tty -s rs232 -t tty -p sa3 -w 4
/etc/methods/cfgtty -l tty5
/etc/methods/define -l tty6 -c tty -s rs232 -t tty -p sa3 -w 5
/etc/methods/cfgtty -l tty6
/etc/methods/define -l tty7 -c tty -s rs232 -t tty -p sa3 -w 6
/etc/methods/cfgtty -l tty7
/etc/methods/define -l tty8 -c tty -s rs232 -t tty -p sa3 -w 7
/etc/methods/cfgtty -l tty8
/etc/methods/define -l tty9 -c tty -s rs232 -t tty -p sa3 -w 8
/etc/methods/cfgtty -l tty9
/etc/methods/define -l tty10 -c tty -s rs232 -t tty -p sa3 -w 9
/etc/methods/cfgtty -l tty10
/etc/methods/define -l tty11 -c tty -s rs232 -t tty -p sa3 -w 10
/etc/methods/cfgtty -l tty11
/etc/methods/define -l tty12 -c tty -s rs232 -t tty -p sa3 -w 11
/etc/methods/cfgtty -l tty12
/etc/methods/define -l tty13 -c tty -s rs232 -t tty -p sa3 -w 12
/etc/methods/cfgtty -l tty13
/etc/methods/define -l tty14 -c tty -s rs232 -t tty -p sa3 -w 13
/etc/methods/cfgtty -l tty14
/etc/methods/define -l tty15 -c tty -s rs232 -t tty -p sa3 -w 14
/etc/methods/cfgtty -l tty15
/etc/methods/define -l tty16 -c tty -s rs232 -t tty -p sa3 -w 15
/etc/methods/cfgtty -l tty16
chdev -l tty0 -a login=enable -a term=ibm3151
chdev -l tty1 -a login=enable -a term=ibm3151
chdev -l tty2 -a login=enable -a term=ibm3151
chdev -l tty3 -a login=enable -a term=ibm3151
chdev -l tty4 -a login=enable -a term=ibm3151
chdev -l tty5 -a login=enable -a term=ibm3151
chdev -l tty6 -a login=enable -a term=ibm3151
chdev -l tty7 -a login=enable -a term=ibm3151
chdev -l tty8 -a login=enable -a term=ibm3151
chdev -l tty9 -a login=enable -a term=ibm3151
chdev -l tty10 -a login=enable -a term=ibm3151
chdev -l tty11 -a login=enable -a term=ibm3151
chdev -l tty12 -a login=enable -a term=ibm3151
chdev -l tty13 -a login=enable -a term=ibm3151
chdev -l tty14 -a login=enable -a term=ibm3151
chdev -l tty15 -a login=enable -a term=ibm3151
chdev -l tty16 -a login=enable -a term=ibm3151
```

- When you have finished editing the /tmp/reconfig1 file, execute the file with the following command:

  ```
  # ./tmp/reconfig1
  ```

  This command will reconfigure the ttys and printers onto the controllers and asynchronous nodes on the multiprocessor as they were on the uniprocessor.

Attention!

If, for example, the uniprocessor does not have any terminal connected and configured on its native serial port S1 because a graphical display was used as a console and you migrate to a multiprocessor that requires an ASCII console, your first tty defined on your 16-port RAN might not be configured on the multiprocessor.

Below is an example with a different asynchronous port mapping. The uniprocessor does not have any tty configured on the native serial ports and the multiprocessor requires a tty0 as a console.

Uniprocessor				Multiprocessor			
sa0	00-00-s1		00-00-s1-00	sa0	00-00-s1	tty0	00-00-s1-00
sa1	00-00-s2		00-00-s2-00	sa1	00-00-s2		00-00-s2-00
sa2	00-01-21	tty0	00-01-21-00	sa2	00-00-s3		
		tty1	00-01-21-01	sa3	00-04-21	tty16	00-04-21-00
		tty2	00-01-21-02			tty1	00-04-21-01
		tty3	00-01-21-03			tty2	00-04-21-02
		tty4	00-01-21-04			tty3	00-04-21-03
		tty5	00-01-21-05			tty4	00-04-21-04
		tty6	00-01-21-06			tty5	00-04-21-05
		tty7	00-01-21-07			tty6	00-04-21-06
		tty8	00-01-21-08			tty7	00-04-21-07
		tty9	00-01-21-09			tty8	00-04-21-08
		tty10	00-01-21-10			tty9	00-04-21-09
		tty11	00-01-21-11			tty10	00-04-21-10
		tty12	00-01-21-12			tty11	00-04-21-11
		tty13	00-01-21-13			tty12	00-04-21-12
		tty14	00-01-21-14			tty13	00-04-21-13
		tty15	00-01-21-15			tty14	00-04-21-14
						tty15	00-04-21-15

Figure 158. Asynchronous Ports Mapping

After running the modified /tmp/reconfig1 file, you will notice that tty0 has been configured on the first native serial port, named sa0. No tty will be configured on the first port (00) of the 16-port RAN.

You will then need to retrieve the original tty0 configuration, and recreate with SMIT a tty16 at location 00-04-21-00 (port 00 of the 16-port RAN).

9.5.3 Optional Tasks

Once all your terminals and printers are properly configured, some additional tasks might need to be performed. This list is not exhaustive:

- Check to see if all the device drivers you need have been installed. Put your product CD-ROM or tape in the installation media, and run the following command:

  ```
  # cfgmgr -i /dev/<device_name>
  ```

  where <device_name> is your installation device.

- Shut down the system.
- Check that the key is in Normal and reboot.
- Check again your overall device configuration.
- Install additional products.
- Install license keys for all the products that require them. Please refer to the redbook titled *A Holistic Approach to AIX V4 Migration, Volume 1,* SG24-4652, for more information on installing license keys.

- Test your entire system.
- Back up your system.

9.6 Upgrade Utility Shell Scripts

You will find below the shell script that is provided with the UP to SMP upgrade in the Model Upgrade Utilities diskette.

9.6.1 mp_prep Shell Script

This shell script prepares the UP system for the upgrade. It installs filesets that are required to enable the MP kernel and support SMP-specific devices. It also create the links that are necessary to build an MP boot image.

```
#!/usr/bin/ksh
#
#   Function: mp_prep
#
#   Description: Prepares a system for hardware upgrade from UP to MP
#
#   ORIGINS: 27
#
#   (C) COPYRIGHT International Business Machines Corp. 1995
#   All Rights Reserved
#   Licensed Materials - Property of IBM
#   US Government Users Restricted Rights - Use, duplication or
#   disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#

# Make sure that this is being executed with root authority
check_root () {
    idname=`id`
    if [ 0 -ne `expr "$idname" : ".*root"` ]
        then return 0
        else {
    echo "You must have root permission to successfully run this program."
    exit 1
        }
    fi
}

# Make sure that the system is at least at 4.1.2
check_412 () {

    [[ $1 -gt 4 ]] && return 0
    [[ $1 -lt 4 ]] && return 1

    [[ $2 -gt 1 ]] && return 0
    [[ $2 -lt 1 ]] && return 1

    [[ $3 -ge 2 ]] && return 0

    #
    # List the maintenance levels which have been seen by this system

    echo
    echo
    echo "ERROR: This system is not fully installed with AIX 4.1.2 or greater."
```

```
    echo "       The highest completely installed level is $1.$2.$3.$4"
    echo
    echo "The levels of AIX known to this system are:"
    /usr/bin/oslevel -q 2>/dev/null
    echo
    echo "If the only level of AIX known to this system level is 4.1.0.0,"
    echo "then the system must be completely reinstalled with AIX 4.1.2.0
    or later."

    return 1
}

# Introduction
introduction () {
    echo
    echo "This script performs the software operations required to install"
    echo "the multi-processor support necessary to prepare a system backup"
    echo "image to be used after converting a uni-processor system to
    a multi-processor"
    echo "system."
    echo
    echo "Continue? (y/n) \c"

    read ans
    [[ -n "$ans" ]] && [[ "$ans" != "y" ]] && {
         echo "Exiting at user request"
    exit 8
    }
}

#============================================================================
# Function:  is_level_greater
#
# Parameters:  2 V.R.M.F. Levels
#
# Returns:  0 if the first level is greater than the second level
#           1 if they are equal
#           2 if the second is greater than the first
#
#============================================================================
function is_level_greater {
   level1=$1
   level2=$2

   [[ $level1 = $level2 ]] && return 1

   greater=`echo "${level1}\n${level2}" |
            sort -t. -k1,1n -k2,2n -k3,3n -k4,4n |
            tail -1`

   if [[ $level1 = $greater ]]; then
      return 0
   else
      return 2
   fi
}
```

```
# Installation device used as input
if [[ $# > 1 ]] || [[ $1 = -* ]]
then
    echo "Usage: mp_prep <installation device>"

    check_root
    if [ $? -eq 0 ]
    then
        echo
        echo "Known installation devices are listed below:"
        /usr/lib/instl/sm_inst list_devices
    fi
    exit 1
fi

nparams=$#
inputdev=$1

introduction

check_root

version=$(/usr/bin/uname -v)
[[ $version -lt 4 ]] && {
       echo
       echo
       echo "ERROR: The system must be installed with at least AIX 4.1.2"
       echo "       This system is installed with some form of Version $version."
       exit 3
}

# Call oslevel to make sure that this system is at least at 4.1.2.0
echo "Verifying that system is installed with at least AIX 4.1.2...\c"
curlevel=$(/usr/bin/oslevel)
[[ -z "$curlevel" ]] && {
    echo
    echo
    echo "ERROR determining system level - could not run /usr/bin/oslevel"
    exit 2
}

# Determine if we're at least at 4.1.2.0
OIFS="$IFS"
IFS="."
set -- $curlevel
check_412 $1 $2 $3 $4

[[ $? -ne 0 ]] && exit 3
# Finish status of 412 check
echo ok

IFS="$OIFS"

if [[ $nparams -eq 0 ]]
then
    echo "Known installation devices are listed below:"
    /usr/lib/instl/sm_inst list_devices
    echo
    while [[ -z "$DEVICE" ]]
```

```
    do
       echo “Enter the name of the installation device to be used: \c”
       read DEVICE < /dev/tty
    done
else
    DEVICE=$inputdev
fi


# Call sm_inst to handle the device manipulation and the installation

# Get system language from /var/adm/ras/bosinst.data
mpmsg=
SYSLANG=`grep -v “–\#” /var/adm/ras/bosinst.data | grep “MESSAGES =” \
| awk `{print $3}’`

echo “Installing multi-processor and system backup software support...”

# Call sm_inst differently based on whether AIX release is 4.1 or 4.2
release=$(/usr/bin/uname -r)

if [[ $release -eq 1 ]]; then

   [[ -n “$SYSLANG” ]]  && [[ “$SYSLANG” != “C” ]] && {
      mpmsg=”bos.msg.$SYSLANG.rte.mp”
   }

   mp_filesets=”bos.rte.mp bos.sysmgt.sysbr devices.mca.fed9 \
devices.mca.8efc devices.rs6ksmp.base $mpmsg”
   /usr/lib/instl/sm_inst installp_cmd -T iems -q -a -X -d \
   $DEVICE -e /tmp/mp_prep.log $mp_filesets

else

   [[ -n “$SYSLANG” ]]  && [[ “$SYSLANG” != “C” ]] && {
      mpmsg=”bos.msg.$SYSLANG.mp”
   }

   mp_filesets=”bos.mp bos.sysmgt.sysbr devices.mca.fed9 \
   devices.mca.8efc devices.rs6ksmp.base $mpmsg”
   echo $mp_filesets >/tmp/mp_filesets.file
   /usr/lib/instl/sm_inst installp_cmd -X -d $DEVICE -f /tmp/mp_filesets.file \
2>&1 | tee /tmp/mp_prep.log
   rm -f /tmp/mp_filesets.file
fi

[[ $? -ne 0 ]] && {
    echo
    echo “ERROR: Failed to install multi-processor software support.”
    [[ -f /tmp/mp_prep.log ]] && {
        echo “       See /tmp/mp_prep.log for further information.”
    }
    exit 4
}

# Link in the mp kernel
[[ -f /usr/lib/boot/unix_mp ]] || {
    echo
    echo “ERROR: /usr/lib/boot/unix_mp does not exist.”
```

```
    [[ -f /tmp/mp_prep.log ]] && {
        echo "       See /tmp/mp_prep.log for further information."
    }

    exit 5
}

/usr/bin/rm -f /unix /usr/lib/boot/unix

/usr/bin/ln -s /usr/lib/boot/unix_mp /unix || {
    echo
    echo "ERROR: Failure linking /usr/lib/boot/unix_mp to /unix."
    exit 6
}

/usr/bin/ln -s /usr/lib/boot/unix_mp /usr/lib/boot/unix || {
    echo "ERROR: Failure linking /usr/lib/boot/unix_mp to /usr/lib/boot/unix."
    exit 7
}

# Fix the .proto files if necessary so that the tape can boot mp and up

[[ ! -f /usr/lib/boot/protoext/tape.proto.ext.base.com.upmp ]] && {
  cat >/usr/lib/boot/protoext/tape.proto.ext.base.com.upmp <<EOF
cfgsys     ---- 777 0 0 /usr/lib/methods/cfgsys
cfgsys_p   ---- 777 0 0 /usr/lib/methods/cfgsys_p
EOF
}

/usr/bin/cat /tmp/mp_prep.log >> /smit.log 2>/dev/null
[ $? -eq 0 ] && {
    echo "The software installation log has been appended to /smit.log"
    echo "for future reference."
}
/usr/bin/rm -f /tmp/mp_prep.log

COOL="yes"

if [[ $release -eq 1 ]]; then

   UP_LEVEL=`lslpp -Lqc bos.rte.up | cut -d":" -f3`
   MP_LEVEL=`lslpp -Lqc bos.rte.mp | cut -d":" -f3`

   if is_level_greater $UP_LEVEL $MP_LEVEL; then
       echo
       echo "WARNING:"
       echo "The bos.rte.up and bos.rte.mp fileset levels do not match."
       echo
       LANG=C lslpp -Lq bos.rte.up bos.rte.mp | grep -v State | grep -v -e "--"
       echo "The level of bos.rte.mp should be at least $UP_LEVEL."
       echo
       echo "Please install the update and run mp_prep again"
       echo "before creating the mksysb tape."
       echo
       COOL="no"
   fi

fi
```

```
lppchk -v > /tmp/lppchk.mp_prep.out 2>&1

[[ -s /tmp/lppchk.mp_prep.out ]] && {

    echo
    echo “lppchk -v has returned the following errors:”
    echo
    cat /tmp/lppchk.mp_prep.out
    echo
    echo “Please install the appropriate updates and run mp_prep again”
    echo “before creating the mksysb tape.”
    echo
    rm -f /tmp/lppchk.mp_prep.out
    COOL=”no”
}

if [[ $COOL = “yes” ]]; then

echo
echo “System is now ready to be backed up in preparation for the hardware upgrade.”
echo “Insert blank tape cartridge into the appropriate tape drive and run”
echo “\”smitty mksysb\” to complete preparation.”
echo

fi

exit 0
```

Chapter 10. SMP Performance Tools

Tuning is an integral requirement for any system. There are a variety of tools included with AIX V4.2 or separately purchasable that can help you in monitoring your SMP system. The objective of this chapter is to introduce the major performance tools that are available to monitor an SMP system.

10.1 AIX V4.1 Performance Tools Considerations

The AIX V4.2 performance tools can be categorized as following:

- Tools that are totally new in AIX V4
- Tools that are SMP specific
- Standard tools or commands that were available in AIX V3 and have been modified to support the SMP environment
- Tools that were included in AIX V3 but are now part of an LPP
- Trace-based tools, available via anonymous FTP for use with AIX V4.1.4 or later

Thus, a review of available tools is necessary before starting in order to see if all is correctly installed and ready to work. First, let's take a look at what these tools do and where we can find them:

The following is a list of tools that did not exist before in AIX V3 and a small description of each of them:

- BigFoot: This tool collects the memory footprint of a running program. It reports the virtual-memory pages touched by the process. BigFoot consists of two commands:
  - `bf` - Collects information about pages touched during the execution of a program. It generates the complete data from the run in a file named _bfrpt.
  - `bfrpt` - Filters the _bfrpt file to extract the storage references made by a given process.
- Performance Diagnostic Tool (PDT)—This tool assesses the current state of a system and tracks changes in workload and performance. It is located in the bos.perf.diag_tool fileset. After this fileset has been installed, the configuration script, `pdt_config`, must be run as root user. It attempts to identify incipient problems and suggest solutions before the problems become critical. PDT is available only on AIX Version 4. For the most part, PDT functions with no required user input.

  PDT data collection and reporting are easily enabled, and no further administrator activity is required. Periodically, data is collected and recorded for historical analysis, and a report is produced and mailed to the `adm` user ID. Normally, only the most significant apparent problems are recorded on the report.
- `stem` (Scanning Tunneling Encapsulation Microscope)—This tool allows insertion of instrumentation code at the entry and exit points of existing programs and library subroutines.

The following tools are SMP specific:

- The cpu_state command shows the number of processors and the current state of each processor within the system. A processor may be enabled, disabled or unknown. This command is part of the bos.rte.mp fileset (bos.mp in AIX 4.2).
- The bindprocessor command enables the binding of individual processes to a processor within the SMP system. This command is also part of the bos.mp fileset.
- The lockstat command provides information on kernel locks caused by the current contention between threads on the system.

The following tools were available on AIX V3, and their output and syntax has been changed to provide related thread and SMP information. As much as possible, the standard options of each tool have been manipulated in order to support SMP and threads; new options have only been created where the existing tools options do not fulfill the new needs.

time — This command measures the real, user, and system time of a program. It takes into account multiple processors.

ps — This command gives processes and threads status.

pstat — This command displays related threads information.

sar — This command shows the system activity and CPU activity for all the processors or for a specific processor.

vmstat — This command displays system activity (memory and CPU usage) for all the processors.

iostat — This command shows the balance between the disks.

Note: The vmstat, iostat and sar commands are part of the bos.acct fileset in AIX V4.

The following AIX performance tools were available with AIX V3 and have not changed in that they still show either global system performance or process performance. However, these tools were previously included in AIX V3. They are now part of the Performance Aide product which is a purchasable product (Program Number 5696-899). In many cases, it would be beneficial to order this LPP since it contains the perfagent.tools fileset with commands such as filemon, fileplace, lvedit, netpmon, rmss, stripnm, svmon, genkex, tprof and lockstat. This LPP is required to monitor this client with Performance Toolbox.

filemon — Monitors the performance of the file system and reports the I/O activity on behalf of logical files, virtual memory segments, logical volumes, and physical volumes

fileplace — Displays the placement of a file's blocks within logical or physical volumes

lvedit — Logical volume editor used for interactive definition and placement of logical volumes within a volume group

netpmon — Monitors activity and reports statistics on network I/O and network-related CPU usage

rmss — Simulates a system with various sizes of memory for performance testing of applications

stripnm	Displays the symbol information of a specified object file
svmon	Captures and analyzes a snapshot of virtual memory
tprof	Reports CPU usage for a specific program

More information about these tools can be found in the *AIX Performance and Tuning Guide,* SC23-2365.

The following trace-based tools are available on the Internet via anonymous FTP from ftp.software.ibm.com under the directory /aix/tools/perftools

These tools are in installp packages and can be installed using SMIT or via the command line by using installp.

utld	Analyzes trace data and gives the user detailed information on system utilization, especially locking
why	Reports on why processes/threads are being delayed in their execution
par	Reports detailed file and device I/O activity

10.2 What Filesets Must be Installed?

Here is the list of the filesets that need to be installed in order to get appropriate performance information, accurate value of system load, and tuning operations.

bos.adt.samples	Contains vmtune, schedtune for kernel tuning
bos.acct	Contains vmstat, iostat, sar
bos.perf.pmr	Contains the monitor for performance pmr
bos.perf.diag_tool	Contains pdt
bos.sysmgt.trace	Contains the trace command
bos.adt.include	Contains the lock classes
perfagent.tools	Contains filemon, svmon, genkex
perfagent.server	Contains xmservd
perfmgr.common	Contains common part of xmperf
perfmgr.local	Contains the local part of xmperf
perfmgr.remote	Contains the network part of xmperf

Note that AIX V4.2 Performance Toolbox can be ordered for local usage or for network usage if you want to monitor remote machines.

10.3 Processes and Threads Status

The ps command is one of the most useful commands for dealing with single-threaded or multithreaded processes on an SMP. The ps command writes to standard output the current status of active processes and, if the -m flag is given, the status of associated kernel threads.

Note: You must use the -o THREAD flag in conjunction with the -m flag to display extra thread-related columns.

In the following example, the TID column shows the thread ID, while the BND column shows which processes and threads are bound to a processor.

```
# ps -mo THREAD
USER   PID  PPID    TID ST  CP PRI SC   WCHAN        F     TT BND COMMAND

root 10318  4684      - A    1  60  1       -   240001  pts/0   - -ksh
   -     -     -  10335 S    1  60  1       -      400      -   - -
```

The ps -m -o THREAD columns have the following meaning:

- USER: User name
- PID: Process ID
- PPID: Parent Process ID
- TID: Kernel Thread ID for threads
- ST: State of the process or kernel thread:
  - A: Active
  - R: Running
  - S: Sleeping
- PRI: Priority of the process or kernel thread
- SC: Suspend Count of the process or kernel thread
- WCHAN: Wait channel of the process or kernel thread. A value in this column means that the thread is waiting. For a kernel thread, this field is blank if the kernel thread is running.
- F: Flags of the process or kernel thread
- BND: The CPU to which the process or kernel thread is bound
- COMMAND: The command being executed by the process

The following screen shot shows the output of the ps command:

```
# 4everunbound &
[1]     6416
# ps -mo THREAD
   USER   PID  PPID    TID ST  CP PRI SC    WCHAN        F     TT BND COMMAND
   root  4114  7422      - A   12  66  1        -   200001  pts/0   - ps -mo TH
READ
      -     -     -  11075 R   12  66  1        -        0      -   - -
   root  6416  7422      - A  480  64  1    8b06c   200001  pts/0   - 4everunbo
und
      -     -     -   4929 R  120 124  0        -        0      -   - -
      -     -     -   6975 R  120 124  0        -        0      -   - -
      -     -     -  10041 S    0  64  1    8b06c      420      -   - -
      -     -     -  11581 R  120 124  0        -        0      -   - -
      -     -     -  11835 R  120 124  0        -        0      -   - -
   root  7422  9980      - A    1  60  1        -   240001  pts/0   - -ksh
      -     -     -  10247 S    1  60  1        -      400      -   - -
```

You can see in this example that the 4everunbound process is active and has four threads running and one sleeping. The sleeping thread is in fact the initial thread corresponding to the main part of the process's code. The initial thread

starts the other threads and goes to sleep. The 4everunbound process is a four-threads process.

Note that, in this example, no threads are bound to a specific processor (- in the BND column).

The following screen is another example where the process and all the threads are bound to processor 3 (see the BND column).

```
# ps -mo THREAD
   USER   PID  PPID    TID ST  CP PRI SC    WCHAN        F     TT BND COMMAND
   root  4118  7422      - A   11  65  1        -   200001  pts/0   - ps -mo TH
   READ
      -     -     -  11079 R   11  65  1        -        0      -   - -
   root  6416  7422      - A  171  64  1    8b06c   200001  pts/0   3 4everunbo
   und
      -     -     -   4929 R   43  85  0        -        0      -   3 -
      -     -     -   6975 R   43  85  0        -        0      -   3 -
      -     -     -  10041 S    0  64  1    8b06c      420      -   3 -
      -     -     -  11581 R   43  85  0        -        0      -   3 -
      -     -     -  11835 R   42  85  0        -        0      -   3 -
   root  7422  9980      - A    2  61  1        -   240001  pts/0   - -ksh
      -     -     -  10247 S    2  61  1        -      400      -   - -
```

Note: If you issue the ps -mo THREAD on a uniprocessor system, you will see, for all of the processes, the value 0 instead of the - in the BND column.

10.4 Binding a Process

AIX V4 allows a user to bind a process to a specific processor by using the bindprocessor command. That process will run only on the designated processor. If the process is multithreaded, all the related threads will be bound to the same processor.

From the command line, it is only possible to bind a process that is currently running. Binding may have some implications in terms of system performance. Since binding is the strongest form of processor affinity, binding a single-threaded process on a specific processor will avoid processor switching for that process. Most of its data will already be in the processor's caches. Thus, binding might slightly increase the performance of that process.

However, binding does not prevent from other processes to be dispatched on the processor on which you bound your process. Binding is different from partitioning. It is not, for example, possible in AIX V4 to dedicate a set of processors to a specific workload and another set of processors to another workload.

This means that a higher priority process might be dispatched on the processor where you bound your process. In this case your process will not be dispatched on other processors. So, you will not always increase the performance of the bound process.

In fact, binding a single-threaded process will improve its performance on an idle system. In this case, if the process is not bound, it will bounce around all the processors and then might suffer a high cache miss rate.

Typically, if you bind a single-threaded program on an idle SMP, you will increase its performance. On the other hand, if you bind the same process on a heavily loaded system, you might decrease its performance because when a processor becomes idle, the process will not be able to run on the idle processor if it is different from the processor on which the process is bound.

If the process is multithreaded, binding the process will bind all its threads to the same processor. This means that the process will not take advantage of the multiprocessing. You will not improve the performance of the process by doing this.

Attention!

Process binding should be used with care because it disrupts the natural load balancing provided by AIX V4, and the overall performance of the system could degrade.

If the workload of the machine changes from that which is monitored when making the initial binding, system performance may suffer. If you do use the bindprocessor command, take care to monitor the machine regularly because the environment may change, making the bound process adversely affect system performance.

The procedure for binding a process to a processor is as follows. Determine which CPU has the lowest workload, in this example, by using sar:

```
# sar -P ALL 1 <n>
```

where <n> is the number of seconds to run the sar command.

Bind the process to that processor by using the following command:

```
# bindprocessor <PID> <procnum>
```

where <PID> is the process ID and <procnum> is the logical processor number.

The following is an example where we bind the smitty process to logical processor 2.

```
# ps -mo THREAD
    USER   PID  PPID    TID ST  CP PRI SC    WCHAN        F     TT BND COMMAND
    root  7296  9580      - A    0  60  1        -   200001  pts/0   - smitty
       -     -     -   7817 S    0  60  1        -      400      -   - -
    root  9580 10346      - A    0  60  1        -   240001  pts/0   - -ksh
       -     -     -  10101 S    0  60  1        -      400      -   - -
```

The smitty process has the process ID 7296. We bind it to logical processor 2 using the bindprocessor command:

```
# bindprocessor 7296 2
```

If we run the ps command again, we can see that process 7296 is bound to processor 2. The BND column indicates on which processor the process is bound.

```
# ps -mo THREAD
    USER   PID  PPID    TID ST  CP PRI SC    WCHAN        F     TT BND COMMAND
    root  7296  9580      - A    0  60  1        -   200001  pts/0   2 smitty
       -     -     -   7817 S    0  60  1        -      400      -   2 -
    root  9580 10346      - A    0  60  1        -   240001  pts/0   - -ksh
       -     -     -  10101 S    0  60  1        -      400      -   - -
```

If we unbind the same process and issue the ps command again, you can see that the process is unbound again.

```
# bindprocessor -u 7296
# ps -mo THREAD
    USER   PID  PPID    TID ST  CP PRI SC    WCHAN        F     TT BND COMMAND
    root  7296  9580      - A    0  60  1        -   200001  pts/0   - smitty
       -     -     -   7817 S    0  60  1        -      400      -   - -
    root  9580 10346      - A    0  60  1        -   240001  pts/0   - -ksh
       -     -     -  10101 S    0  60  1        -      400      -   - -
```

Note: A process cannot be bound until it is already running; that is, it must exist in order to be bound.

When using the bindprocessor command, if the process does not exist, you will get the following message:

```
# bindprocessor 13039 3
# 1730-002: Process 13039 does not match an existing process
```

If the processor does not exist, you will get the following message:

```
# bindprocessor 13038 4
# 1730-001: Processor 4 is not available.
```

The bindprocessor command can also be used to query available processors. It uses the logical numbers. Following is the output of the bindprocessor command.

```
bindprocessor -q
The available processors are:  0 1 2 3
```

10.5 Binding a Thread

It is not possible to bind a specific thread to a processor from the command line. In other words, you cannot enter:

bindprocessor <TID> <procnum>

where <TID> is the thread ID and <procnum> is the logical processor number.

The bindprocessor command expects a process ID, not a thread ID.

But you can bind one or several threads within a process at the programming level by using the bindprocessor(What, Who, Processor) call. The bindprocessor() call must be used in the source code of your program.

What BINDPROCESS (for a process) or BINDTHREAD (for a thread)

Who Process or thread identifier

Processor Logical processor number

If the bindprocessor() call is used within a piece of code to bind threads to processors, the threads will stay with these processors and can be unbound individually with bindprocessor(What, Who, PROCESSOR_CLASS_ANY).

However, if the bindprocessor command is used on that process, all the threads belonging to that process will then be bound to the same processor. That is, the bindprocessor command will supersede the threads binding. If you then unbind the whole process, the threads will all be unbound and will bounce around all the processors. You will lose the original threads' binding.

10.6 Using the Standard Performance Tools on your SMP

This section describes how to use the standard performance tools, such as sar, vmstat, pstat and commands such as the time command, which is useful for measuring the throughput or scalability of a system.

This section references a number of test programs that have been written to show what happens, for example, when threads are bound to processors or to test loading of the CPUs in various ways.

These programs have their source listed in Appendix B, "Sample Programs" on page 333. You can enter and compile them yourselves. If you are an IBM employee, you can get them by entering the following command:

TOOLS SENDTO WTSCPOK TOOLS AIXDISK GET AIXV4SMP PACKAGE

Below is a short description of these sample programs:

100unbound A four-thread process with threads not bound to any processor

100boundon1 A four-thread process with all the threads bound on one processor

100boundon2 A four-thread process with threads bound on two processors

4everunbound An everlasting four-thread process with no threads bound

4everboundon1 An everlasting four-thread process with all threads bound to one processor

4everboundon2 An everlasting four-thread process with threads bound to two different processors

4everboundon4 An everlasting four-thread process with threads bound to four different processors

cpubound A single-threaded process bound on one processor

All these sample programs help to illustrate the use of the standard performance tools on the SMP system.

10.6.1 Multiprocessing Effect

The output of the `time` command takes a new meaning in an SMP although its output did not change. When measuring the execution time for a process on an SMP, the real or elapsed time can be smaller than the user time. This is quite unusual on a UP.

In fact, on an SMP, the user time is the sum of all the user times spent by the process's threads on all the processors.

In the following example, we measured the running time of the 100unbound program, which is a process with four threads.

Running it on a UP system (model 250: PowerPC 601, 67 MHz) shows the time results where the real time is greater than the user time.

```
# time ./100unbound

real    0m11.70s
user    0m11.09s
sys     0m0.08s
```

Running it on a four-way SMP system (PowerPC 601, 75 MHz) shows that the real time is only about a fourth of the previous real time.

```
# time ./100unbound

real    0m2.59s
user    0m9.99s
sys     0m0.01s
```

This shows that the multithreaded process takes advantage of the multiprocessor. The user time on the four-way SMP is similar to the user time on the UP. But the real time (the time it takes to get the result) is about four times faster on the four-way SMP.

Therefore, having several processors improves the performance of multithreaded applications and improves the overall throughput of the system.

10.6.2 SMP Scaling

Also the `time` command allows you to measure the scaling effect of the SMP. In order to look at the scaling effect of the SMP, we ran different versions of the same four-thread program.

If you run all the threads on one processor only, the real time and the user time of the process are approximately equal (like on a UP system).

```
# time ./100boundon1

real    0m10.04s
user    0m9.93s
sys     0m0.02s
```

When you run the threads on two processors only, the user time is still the same, but the real time is approximately half of the previous real time.

```
# time ./100boundon2

real    0m5.08s
user    0m9.98s
sys     0m0.01s
```

When you run the threads on four processors, the user time is still the same, but the real time is about one fourth of the real time we got when running on one processor only.

```
# time./100bound

real    0m2.56s
user    0m9.97s
sys     0m0.02s
```

Note: All these examples were run on an SMP that had nothing else running on it. If the system is doing work, results will differ. Also, the first time you run a test, there might be different answers as a result of what is in memory.

Attention!

Binding all the process's threads to the same processor is not exactly equivalent to running the same program on a uniprocessor or a one-way SMP. When you bind all the threads to the same processor, system activity (like the scheduler) can still run on the other processors. In a UP system or a one-way SMP system, user activity and system activity have to compete for the same processor resource (the unique CPU).

The best way to measure the scalability of an SMP system is to disable all the processors except one and then enable the processors one at a time.

10.6.3 Threads-Related Information

The `pstat` command is the noninteractive form of the `crash` command. Because it has a number of new options, `pstat` is useful for looking at threads.

`-A`	Shows all entries in the kernel thread table
`-P`	Shows runnable kernel threads only
`-U <proc number>`	Shows thread slot user structure of kernel threads for any given processor
`-S`	Shows processor status (which thread is running on which processor at the time of the command)

Following is the output of the `pstat -S` command:

```
# pstat -S
STATUS OF PROCESSORS:

CPU     TID  TSLOT     PID  PSLOT  PROC_NAME
  0    40e9     64    3da8     61  crash
  1    3be5     59    3ba6     59  4everunbound
  2    1be3     27    3ba6     59  4everunbound
  3    3ce7     60    3ba6     59  4everunbound
```

In the above example, you can see which thread was running on which processor when the pstat command was issued. The TID of the thread is in hexadecimal.

Following is the output of the pstat -P command:

```
# pstat -P
THREAD TABLE:

SLT ST    TID      PID    CPUID  POLICY PRI CPU    EVENT  PROCNAME     FLAGS
  2 r     205      204        0    FIFO  7f  78               wait
        t_flags:  sigslih kthread
  3 r     307      306        1    FIFO  7f  78               wait
        t_flags:  sigslih kthread
  4 r     409      408        2    FIFO  7f  78               wait
        t_flags:  sigslih kthread
  5 r     50b      50a        3    FIFO  7f  78               wait
        t_flags:  sigslih kthread
 19 r    13a7     1068        2   other  5a  34           4everboundon2
        t_flags:
 27 r    1ba5     1068        3   other  5a  34           4everboundon2
        t_flags:
 31 r    1f05     26fc  unbound   other  3c   1            telnetd
        t_flags:
                  sel
 39 r    27a9     1068        3   other  5a  35           4everboundon2
        t_flags:
 43 r    2bb3     1972  unbound   other  57  36              pstat
        t_flags:
 46 r    2ea3     1068        2   other  5a  35           4everboundon2
        t_flags:
```

The meaning of the different fields are the following:

SLT — Shows the slot number in the threads table

ST — Shows the status of the thread as to whether it is running, sleeping or otherwise

PID — The process ID to which the thread belongs

CPUID — The ID of the processor on which the process is bound. If the thread is not bound, then the word unbound is displayed in this field

POLICY — The thread's scheduling policy

PRI — The priority in hexadecimal

CPU — Shows the short-term CPU usage of the thread. The maximum value for this field is 120 ticks

FLAGS — Displays the signal that the process is currently waiting on if the thread is waiting

In AIX Version 4, there are three possible values for thread-scheduling policy:

- FIFO: Once a thread with this policy is scheduled, it runs to completion unless it is blocked; it voluntarily yields control of the CPU or a higher-priority thread becomes dispatchable. Only fixed-priority threads can have a FIFO scheduling policy.
- RR: This is similar to the AIX Version 3 scheduler Round-Robin scheme based on 10ms time slices. When an RR thread has control at the end of its time slice, it moves to the tail of the queue of dispatchable threads of its priority. Only fixed-priority threads can have an RR scheduling policy.
- OTHER: This policy is defined by POSIX1003.4a as implementation-defined. In AIX V4, this policy is defined to be equivalent to RR, except that it applies to threads with nonfixed priority. The recalculation of the running thread's priority value at each clock interrupt means that a thread may lose control because its priority value has risen above that of another dispatchable thread. This is the AIX Version 3 behavior, and this is the default scheduling policy in AIX V4.

10.6.4 Measuring the Processors Load

The processors load can be measured with the sar command. The syntax of the sar command is the following:

```
sar [ -P <processor_id> [, ... ] | ALL ] <interval> <count>
```

Use of sar -A -P ALL <x> <y> where <x>=time interval in seconds and <y>=number of iterations shows information about all the SMP-related counters, such as forks, characters read, characters written, per processor, and so on.

```
# sar -P ALL 1 2

AIX smp1 1 4 00000000A000    04/04/95

18:37:57 cpu    %usr    %sys    %wio   %idle
18:37:58  0        0       0       0     100
          1        0       2       0      98
          2      100       0       0       0
          3      100       0       0       0
          -       50       0       0      50
18:37:59  0        0       1       0      99
          1        0       0       0     100
          2      100       0       0       0
          3      100       0       0       0
          -       50       0       0      50

Average   0        0       0       0     100
          1        0       1       0      99
          2      100       0       0       0
          3      100       0       0       0
          -       50       0       0      50
```

In the above example, processors 2 and 3 are 100 percent busy, while processors 0 and 1 are idle. For each sample interval, the fifth line is the average CPU utilization for the entire system. At the end, the sar command reports the average CPU usage for each processor. The last section gives the average over the sample period.

Using the -P <processor_id> reports per-processor statistics for the specified processor or processors separated by a comma.

The ALL keyword reports statistics for each individual processor and globally for all processors. Of the flags which specify the statistics to be reported, only the -a, -c, -m,-u, and -w flags are meaningful with the -P flag, while meaningless flags are silently ignored.

10.6.5 Global Memory and CPU Activity

The vmstat command reports CPU and disk-I/O activity as well as memory utilization data for the entire system.

The vmstat command syntax is unchanged from AIX V3.2.

```
vmstat <interval> <count>
```

However, the output has changed in that the leftmost column is now in terms of threads, not processes.

In the following example, we run a program called cpubound that loads one processor out of the four. The vmstat command shows a 25 percent user CPU usage. In fact, vmstat shows the overall CPU usage (for the entire system). With vmstat, you cannot see a per-processor CPU usage.

```
# vmstat 2 5
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 3  1  2972 25308   0   0   0   0    0   0 419  366 181 99  1 99  2
 1  1  2972 25308   0   0   0   0    0   0 421  237  42 25  0 74  0
 1  1  2972 25308   0   0   0   0    0   0 422  197  39 25  1 74  0
 1  1  2972 25308   0   0   0   0    0   0 420  196  39 25  0 75  0
 1  1  2972 25308   0   0   0   0    0   0 420  204  40 26  0 74  0
```

The vmstat command is generally used for an overall look at resource utilization while running a multiuser workload.

The vmstat output shows a kthr column that reports the kernel thread state changes per second over the sampling interval:

r Number of kernel threads placed in run queue

b Number of kernel threads placed in wait queue (awaiting resource or awaiting input/output)

Threads waiting for a lock to be released do not show up in either the run queue or block queue column.

10.7 Sizing an SMP

It may be useful to disable or enable processors on an SMP system in order to measure the scalability of the system on a specific workload. We saw previously that an SMP system will scale if all components of the system scale well. The hardware must scale, the operating system must scale, and the application itself must scale. From a user point of view, you cannot change the ability of the hardware or the operating system to scale very well. IBM has done a lot of work on the hardware side as well as the AIX side (optimization of AIX on four-way,

six-way, and eight-way SMPs). But some work can be done at the application level to improve the scalability of your SMP.

Thus, measuring the scalability of an SMP system on a specific workload may help you in improving the scalability of your application. Measuring the scalability of a system can be done by enabling or disabling processors.

Enabling or disabling processors can also be used to size a system for a specific application and to determine the number of processors that are needed to run the application with good performance.

In either case, the cpu_state command can be used to measure the scalability of a system or size a system.

The cpu_state command lists and controls which processors on a multiprocessor system will be active when the system is next started. It is the only command that shows ALL physically present processors.

The -l flag displays a report that would look like the following for a J30 with four processors.

```
# cpu_state -l
        Name    Cpu      Status         Location
        proc0   0       enabled         00-0P-00-00
        proc1   1       enabled         00-0P-00-01
        proc2   2       enabled         00-0Q-00-00
        proc3   3       enabled         00-0Q-00-01
```

where:

- Name is the ODM processor name. It is shown in the form procx, where x is the physical processor number.
- Cpu is the logical processor number. Only enabled processors have logical numbers.
- Status is the processor state for the next boot.
- Location is the ODM processor location code. It is shown in the form AA-BB-CC-DD, where:
  - AA is the main unit, always 00.
  - BB is the processor board number 0P, 0Q, 0R, or 0S, indicating, respectively, the first, second, third or fourth processor card.
  - CC is always 00.
  - DD is the processor position on the CPU card.

The -d or -e flags, respectively, disable or enable the processor identified by the processor number. An example of this follows:

```
# cpu_state -d 2
# cpu_state -l
        Name    Cpu      Status         Location
        proc0   0       enabled         00-0P-00-00
        proc1   1       enabled         00-0P-00-01
        proc2   2       disabled        00-0Q-00-00
        proc3   3       enabled         00-0Q-00-01
```

The system requires a reboot to actually disable the processor. The logical processor numbering will change after the reboot.

After the reboot, the Cpu number allocated to the physical processor that has been disabled becomes a -, indicating that the physical processor has now been fully disabled.

For example:

```
# cpu_state -l
        Name    Cpu      Status         Location
        proc0   0       enabled         00-0P-00-00
        proc1   1       enabled         00-0P-00-01
        proc2   -       disabled        00-0Q-00-00
        proc3   2       enabled         00-0Q-00-01
```

10.8 Lock Contention

In an SMP system, processes run across several processors to complete a specific task. Some of these processes access memory addresses that are shared with others; they must not update the same area of memory simultaneously because the outcome cannot be predicted. Locks are used to serialize access to shared data.

Finding the right granularity when using locks is one of the big challenges in an MP operating system.

AIX V4 was changed and continues to be enhanced to make it more MP-efficient. This means that the system is optimized to spend the minimum time waiting for and dealing with locks. AIX V4 defines subsystems comprised of 256 lock classes in /usr/include/sys/lockname.h.

AIX V4.2 was changed in order to reduce lock contention inside the kernel. Previously there was only one lock for the whole process table; now each process entry can be locked. For the JFS, the global JFS_LOCK has been cut into several smaller locks for operations on cache, directory, and inodes.

However, it is the kernel developer′s responsibility to define and implement an appropriate locking strategy to protect the program′s own data. This choice has a big consequence for the scalability of the system. The lockstat command supports the use of user-supplied lock names in files named /usr/include/sys/lockname_*.h, where * is a wildcard.

The /usr/include/sys/lockname.h file contains the list of all system lock classes.

AIX developers can choose between two types of locks:

- Simple locks are exclusive and allow the process to spin (run a small loop) while waiting for the lock to become available.
- Complex locks are read/write locks (one writer at a time and several readers) that block the process while waiting for the lock to be released.

Lock implementation in an application could make the application run faster or slower depending on the locking granularity. Finding the right granularity for locks implementation is a difficult task.

In AIX, you can use the lockstat command to see the use of locks. Only kernel locks can be seen with the lockstat command.

To enable the use of lockstat, you must install the bos.adt.include fileset and create a new boot image using the bosboot command with the -L flag, which enables MP lock instrumentation.

Your command should look like this:

```
# bosboot -a -d hdisk<n> -L
```

This is a sample output of the lockstat command:

```
# lockstat -a

Subsys  Name                      Ocn   Ref/s   %Ref   %Block  %Sleep
-------------------------------------------------------------------------

PROC    TOD_LOCK_CLASS             0    1667   15.61   34.97    0.00
PROC    PROC_INT_CLASS            --    1161   10.87   10.08    0.00
PROC    U_TIMER_CLASS             48     367    3.44   51.77    0.00

First 10 largest reference rate locks :

Subsys  Name                      Ocn   Ref/s   %Ref   %Block  %Sleep
-------------------------------------------------------------------------

VMM     VMM_LOCK_VMKER            --    2602   24.36    0.08    0.00
PROC    TOD_LOCK_CLASS             0    1667   15.61   34.97    0.00
PROC    PROC_INT_CLASS            --    1161   10.87   10.08    0.00
VMM     VMM_LOCK_PDT              --     963    9.02    0.10    0.00
PFS     ICACHE_LOCK_CLASS         --     527    4.93    0.00    0.00
VMM     VMM_LOCK_LV               23     513    4.80    0.00    0.00
PROC    U_TIMER_CLASS             48     367    3.44   51.77    0.00
VMM     VMM_LOCK_LV               --     314    2.94    0.00    0.00
XLVM    LVM_LOCK_CLASS             0     248    2.32    0.00    0.00
LOCKL   LOCKL                     45     243    2.28    0.00    0.00
```

The column headings in the lockstat command listing have the following meaning:

- Subsys: The subsystem to which the lock belongs:
  - PROC: scheduler, dispatcher, interrupt handler
  - VMM: pages, segments, free list
  - TCP: sockets, NFS
  - PFS: inodes, icache
- Name: The symbolic name of the lock class, which can be:
  - TOD_LOCK_CLASS: all interrupts that need the Time-Of-Day (TOD) timer
  - PROC_INT_CLASS: interrupts for processes
  - U_TIMER_CLASS: per process timer lock
  - VMM_LOCK_VMKER: free list
  - VMM_LOCK_PDT: paging device table
  - VMM_LOCK_LV: per paging space
  - ICACHE_LOCK_CLASS: inode cache

In AIX V4, the TOD_LOCK_CLASS seems to have the highest Ref/s count. This is because all the interrupt handlers need the TOD registers at the same time.

The LOCKL subsystem and class is for AIX Version 3 locks. This type of lock is still available in AIX V4 for running unchanged AIX V3.2 applications that use the lockl subroutine.

Ocn — Occurrence number of the lock in its class

Ref/s — Reference rate or number of lock requests per second

%Ref — Reference rate expressed as a percentage of all lock requests

%Block — Ratio of blocking lock requests to total lock requests. Block occurs whenever the lock cannot be taken immediately

%Sleep — Percentage of lock requests that cause the calling thread to sleep

If vmstat indicates that there is a significant amount of CPU idle time when the system seems subjectively to be running slowly, delays may be due to kernel locks contention.

In AIX Version 4, this possibility can be investigated with the lockstat command;

Look for the following pointers:

- Check lockstat output for Ref/s > 1000
- Identify subsystems and lock classes that have a high number of Ref/s

Application problems can only be seen indirectly. If there is locks contention, you must check for bottlenecks due to the application.

For example, if your application has a high number of processes that read and write in a unique message queue, you might have lock contention for the Virtual Memory Manager (VMM) subsystem. Adding more message queues may reduce the level of locks contention.

10.9 Additional Trace-Based Tools

The following trace-based tools were originally used in the development of AIX, but have now been made generally available; they can be obtained on the Internet, via anonymous FTP from ftp.software.ibm.com in the directory /aix/tools/perftools.

utld — Reports on system locks and delays. Although intended for use on all AIX platforms, this tool is especially useful in an SMP environment. It reports thread/processor affinity and gives a greater insight into locking that can be gained by using tools such as lockstat. This tool also provides a very useful summary of system utilization.

par — Reports on disk I/O transfers and utilization. The report produced contains three sections:
- Summary of all files referenced per process
- Logical read/write times per process/file
- Physical transfer times and utilization per disk

why — Reports the extent and reasons why processes/threads are delayed in their execution. It can also provide an optional report to examine the correlation of execution of a given thread with all other threads in the

system. In particular, this option may be used to examine the state of all processes when the idle process(es) execute.

Further documentation on all these commands is included in their respective packages. Only the `utld` command is discussed in greater detail in this chapter because it is particularly relevant when working with SMP servers.

Important!

Please note that these tools are currently available free of charge and as such are provided without warranty or support. As they are development tools, they may change significantly in future releases. The following notes are intended for use with Version 1.x of the tools, for AIX V4.1.4 or later.

Since these tools all depend on data captured using the `trace` command, we first take a look at how to capture trace data and also how to create a trace names file.

10.9.1 Trace

When using trace-based tools, information must first be captured by using the `trace` command. Running a trace adds significant system overhead in path length, disk space, and memory consumption, and therefore, should only be used selectively to address finite problems.

In this section, we make some recommendations on using AIX `trace`; however, for definitive documentation on using `trace`, please consult the standard AIX documentation. We recommend you invoke `trace` with the following set of parameters:

```
# trace -a -d -f -T 80000000 -L 80000000 -o ./trace.out
```

When using `trace`, the classic trade-off of time versus resource exists. On a fast, busy system with all trace hooks collected (the default), `trace` will produce mega-bytes of trace output for each second of real time. This necessitates zeroing in on the phenomenon you wish to study. In the above example, the `trace` command is started detached (`-a`) and deferred (`-d`). This puts `trace` running in the background but not yet collecting trace hooks. `trace` will collect 80 MB of trace data (`-T`) and stop collecting trace data when the buffer fills (`-f`). It is important that the system has sufficient free memory (> 80 MB) that can be dedicated to the `trace` command, or the command will fail or the workload will be perturbed (perhaps by excessive paging). The maximum length for the output file is 80 MB (`-L`), and the output is written to the file ./trace.out (`-o`).

When a deferred trace collection is used, `trace` is started via the `trcon` command and stopped using the `trcoff` command. So a trace collection for the `find` command might be done as follows:

```
# trace -a -d -f -T 80000000 -L 80000000 -o ./trace.out
# trcon ; find / -name munga.out -print ; trcoff
# trcstop
```

Because the `-f` flag is included, the `trace` will end if the buffer fills, so the `trcoff` command may fail with an error message indicating that the trace is not currently running.

If the workload is more complex, or is running in the background (for example, a database engine), the same basic technique is used. When the workload exhibits interesting behavior, the trace may be started as follows:

```
# trace -a -d -f -T 80000000 -L 80000000 -o ./trace.out
# trcon ; sleep 60 ; trcoff
# trcstop
```

It is unlikely, in the example above, that a full 60-second trace will be collected due to the use of the -f flag. You will actually get a trace corresponding to one full trace buffer (that is, 80 MB). Please bear in mind that trace adds considerable path length in system time; so using trace will distort the user/system time mix.

Note

Do not forget to stop the trace by using trcstop once you have collected your data.

The trace data collected will be in binary format that can be translated into ASCII by the trcrpt program. To translate a trace file (for example, trace.out), enter the following:

```
# trcrpt ./trace.out | pg
```

For further information regarding the trace command, please refer to the *AIX Performance Tuning Guide,* SC23-2365.

10.9.1.1 Capturing Names

In order for the trace tools to provide meaningful reports, the “names” of the system must be collected by using the trcnm tool. This tool builds a load map of what’s in the AIX kernel, so things like device drivers can be identified in the trace. The following command can be used to collect names:

```
# trcnm > names.out
```

If you are unsure of what device driver names map to adapters, use the command:

```
# lsdev -Cc adapter
```

to get an explanation of the function of each of the adapter drivers in your system. The third step is to unwrap the trace. Since the trace file is collected in a double buffered fashion, it must be put into the correct order. The following command will unwrap the trace:

```
# trcrpt -r trace.out > trace.unwrapped
```

Ensure you have enough space in the file system before entering this command because it effectively duplicates the trace. Note, this example required 260 MB of disk space. More space will be required on an active system.

10.9.2 utld

Once the trace has been unwrapped, the utld command can be invoked as follows:

```
# utld -i trace.unwrapped -n names.out > utld.out
```

Note that if the -n option is omitted, the utld command will automatically generate a list of names using the trcnm command as outlined above.

This command produces a summary report in the utld.out file. If you require all locking details, run the command as follows:

```
# utld -i trace.unwrapped -n names.out -d -l lock.out > utld.out
```

Note

In order for the trace command to capture information regarding locks, MP Lock Instrumentation must be enabled. For further details regarding this, please see the previous section *"Lock Contention"* on page 217.

If the -d -l <filename> option is specified when invoking utld, a separate report detailing the locking statistics can be obtained. To look at locking in greater detail, it is necessary to provide utld with a larger dictionary of names. The larger dictionary is created by using the getnames executable provided in the utld package, as follows:

```
# getnames > names.out
```

The getnames facility uses the genkex executable, provided as part of the perfagent.tools section of Performance Toolbox. This must be installed on the system in order for this to work.

Using getnames is only necessary if detailed locking information is required.

This time, utld will create the following files:

utld.out	A summary report
lock.out	A detailed report on process locks
lock.out.details	A detailed report on locks per process

Once the utld command has completed, the lock.out and lock.out.details files will be very large. The utld.out report contains some very useful information for anyone concerned with SMP performance. However, lock.out and lock.out.details are very in-depth technical reports designed primarily for use by application developers.

10.9.2.1 Sample utld Report

The following screen shots are examples of output contained in the utld.out report:

The first section of the utld report (see following figure) is a summary of CPU consumption during the trace period. This breaks down CPU usage into five categories:

Application	More commonly referred to as user mode
Kernel	The time spent in kernel mode

FLIH The amount of time spent dealing with first line interrupt handlers

SLIH The amount of time spent in second line interrupt handlers

```
                    SYSTEM SUMMARY

 processing         percent       percent
total (msecs)      tot  time     busy time     processing category
=============     =========     =========     ====================
  4834.901           47.452        47.452       APPLICATION
  3249.240           31.890        31.890       KERNEL
  1907.669           18.723        18.723       FLIH
   132.542            1.301         1.301       SLIH
    64.625            0.634         0.634       DISPATCH
-------------     ---------     ---------
 10188.976          100.000       100.000       CPU(s) busy time
     0.000            0.000                     WAIT
-------------     ---------
 10188.976          100.000                     TOTAL

          Total number of process dispatches          =  5041
          Average time between same process dispatch  =  20.475002 msec.
          Average process to processor affinity       =  0.528291
```

If the trace was collected on an SMP server (as in these examples), the CPU breakdown will include a weighted average of all the processors, as well as information for each processor. For an SMP, the report will also include the number of dispatches and a measure of the processor affinity of the threads.

Following the system summary, there is a section detailing the system usage by processor.

The next section presents a list of all processes/threads that ran during the trace, and for each, specifies the amount of time spent in kernel and user mode (see below).

```
                APPLICATION and KERNEL SUMMARY (Per Thread/Process)

--  processing total  (msecs) --    -- percent of total processing time --
  combined   application  kernel    combined      app      kernel   process name (proc id / thrd id)
  ========   ===========  ======    ========      ===      ======   =======================
   773.399    773.399      0.000      7.591      7.591      0.000   trace ( 18262  18527 )
   683.106    656.024     27.082      6.704      6.439      0.266   nroff ( 17974  18239 )
   660.407    656.398      4.008      6.482      6.442      0.039    no name ( 18456  18721 )
```

The above information is then condensed into time spent per type of process, headed by title information reporting the total number of threads, as follows:

```
                    APPLICATION and KERNEL SUMMARY (Per Process Type)
                    total number of threads = 57

--  processing total  (msecs) --    -- percent of total processing time --
  combined   application  kernel    combined      app      kernel   process name (count)
  ========   ===========  ======    ========      ===      ======   ====================
   773.399    773.399      0.000      7.591      7.591      0.000   trace ( 1 )
   730.833    698.473     32.361      7.173      6.855      0.318   nroff ( 2 )
  4560.335   2333.526   2226.809     44.758     22.902     21.855    no name ( 14 )
   332.740     15.698    317.042      3.266      0.154      3.112   cpio ( 1 )
   321.571    237.929     83.643      3.156      2.335      0.821   xlcentry ( 1 )
   189.248     62.744    126.504      1.857      0.616      1.242   ed ( 1 )
```

The third part of the report breaks down the total kernel execution time into a list of all system calls that were made:

```
                    KERNEL (System Call) Summary

 processing        percent                - path in msecs         -
total (msecs)     proc time     count     -min-     -avg-     -max-     system call
=============     =========     =====     =====     =====     =====     ===========
    744.791         7.310    000000193     1.598     3.859     7.323     creat
    431.640         4.236    000000659     0.098     0.655     2.109     statx
    317.197         3.113    000000204     0.935     1.555     4.080     chown
    305.065         2.994    000000205     0.902     1.488     3.697     chmod
    269.923         2.649    000000196     0.659     1.377     3.191     utimes
    207.466         2.036    000000284     0.193     0.731     5.273     open
     71.907         0.706    000000544     0.004     0.132     0.506     close
     69.426         0.681    000000016     2.385     4.339     7.686     unlink
```

The next two sections detail the FLIH and SLIH statistics (see below). The final section presents a summary of the idle process for each processor.

```
                    FLIH Summary

 processing        percent                - path in msecs         -
total (msecs)     proc time     count     -min-     -avg-     -max-     flih type
=============     =========     =====     =====     =====     =====     ====================
   1158.986         2.663    000004931     0.005     0.235     0.852     DECREMENTER
    437.893         1.006    000000616     0.034     0.711     1.482     DATA ACCESS PAGE FAULT
    182.108         0.418    000000495     0.010     0.368     0.897     I/O INTERRUPT
     13.889         0.032    000000089     0.003     0.156     0.552     level 50
      0.038         0.000    000000004     0.004     0.010     0.022     FLOATING POINT UNAVAIL
-------------     ---------     -----
   1792.913         4.119    000006135


                    SLIH Summary

 processing        percent                - path in msecs         -
total (msecs)     proc time     count     -min-     -avg-     -max-     slih type
=============     =========     =====     =====     =====     =====     ====================
     62.156         0.143    000000340     0.066     0.183     0.300     ascsiddpin
     50.828         0.117    000000155     0.070     0.328     0.921     tokdd
-------------     ---------     -----
    112.984         0.260    000000495
```

The `utld` command is an extremely powerful tool. It is beyond the scope of this redbook to cover all the options and reports available. More information can be obtained from the ultd.doc file provided with the package.

10.10 Monitoring your SMP with Performance Toolbox

This section introduces how to use the Performance Toolbox graphical tool to control and monitor the performance of your SMP system.

10.10.1 Performance Toolbox Introduction and Concepts

The Performance Toolbox (PTX) is a graphical tool to monitor the performance of your system. Performance Toolbox 2.1 for AIX V4.1 supports the SMP environment and can be used to monitor an SMP system.

Performance Toolbox is shipped as two LPPs:

- Performance Toolbox 2.1 for AIX V4.1 (program number 5696-900)—This product is used for AIX V4.1 only.
- Performance Toolbox 2.2 for AIX V4.2 (program number 5765-654)—This product has two versions, local and remote, and can be used for AIX V4.1 or AIX V4.2. It contains the performance agents for AIX V4.1 and V4.2 as features codes. It can be ordered for 1, 2, or more clients.
- Performance Aide 2.1 for AIX V4.1 (program number 5696-899)—This product, which was for the client, was withdrawn in July 1996.

Important!

When using Performance Toolbox, please ensure you obtain the very latest updates. This is especially relevant when using Version 2.1 to monitor an SMP server.

The following terms are used to refer to Performance Toolbox functions or components:

- A *Console* is a graphical window containing instruments that monitor the system. A console can have one or many instruments.
- An *Instrument* is a graphical view of monitored values, and each instrument can show one or more values that are monitored. The presentation of the values can be in form of graphs, gauges and so on.
- A *Value* is the unit to be monitored; it can be any piece of the system able to be monitored. For example, CPU usage, memory, disk activity, and so on.
- *Groups of statistics* are a functional part of the system. The values are grouped in relation to the functional part of the system they belong to. However, an instrument can have values from several groups.

Another very useful graphical performance tool shipped with Performance Toolbox is `3dmon`. The `3dmon` monitors the system(s) using 3D bars that dynamically change whenever any value measured changes.

In order to monitor an SMP with Performance Toolbox, you need a graphical display. While you may use a graphical display connected to the server, (either a G40 or J40 with a display adapter, or an Xstation), we strongly recommend that you use a separate graphics workstation. This will minimize the impact of running Performance Toolbox on the system load, and will give a true indication of system performance. You will need to install the following filesets at the location shown:

perfagent.server	on the client
perfagent.tools	on the client
perfmgr.common	on the server
perfmgr.local	if you only want to monitor your system
perfmgr.network	to monitor remote systems and use chmon

The fileset perfagent.server contains the xmservd daemon. This daemon must be installed and running on all monitored systems. The fileset perfagent.tools contains all the performance tools that were included in AIX V3.2, such as rmss, filemon, netpmon, svmon, and so on. It also contains new AIX V4.1 tools, such as lockstat, stem, bf, and fdpr.

The fileset perfmgr contains the graphical part of PTX. xmperf and 3dmon are part of this fileset.

The network manager also contains a curses-based monitor and its source code in the /usr/lpp/perfmgr/network/bin/chmon file.

Remote AIX, HP-UX, and Solaris 2.4 systems can be monitored with the remote option. HP and SUN data supplier daemons belong to the agent.

If you monitor your SMP from an X-station, all these filesets must be installed on the SMP system. But if your monitor your SMP from another workstation, you only need to install the perfagent filesets on the SMP and the perfmgr fileset on the graphical workstation.

Note: When you install the perfagent filesets for the first time on your SMP, you need to reboot the system or refresh the inetd daemon in order to start the xmservd daemon.

To refresh the inetd daemon, use the following command:

```
# refresh -s inetd
```

To start monitoring locally your SMP system, use the following command:

```
# xmperf
```

If you want to monitor your SMP from a remote host (a graphical workstation), enter the following command:

```
# xmperf -h <hostname>
```

10.10.2 Creating an SMP Console

PTX provides a predefined console for monitoring an SMP system. But in this example, we will create our own console to monitor CPU statistics on all of the SMP processors (four-way SMP).

Start PTX by using xmperf from the command line. You will then be presented with an initial window similar to Figure 159 on page 317.

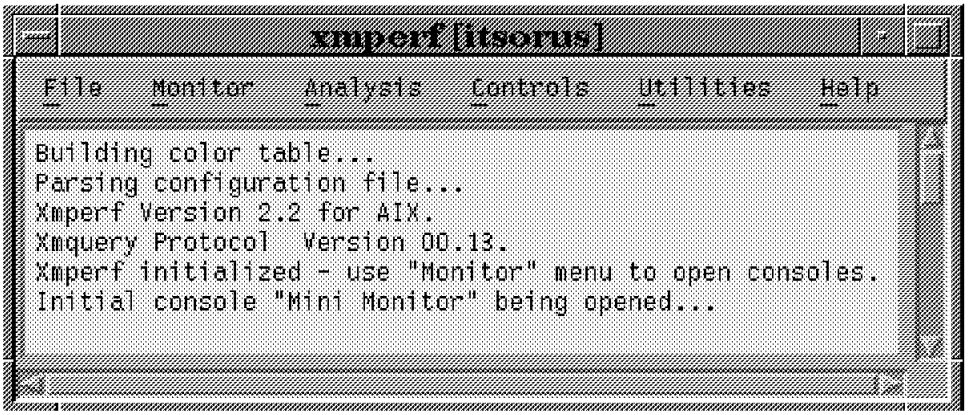


Figure 159. xmperf Initial Screen

Use the Monitor pull-down menu to select **Add New Console**. You will then get a subwindow inviting you to enter a console name. Give a meaningful name to your console instead of using the unique, default name.

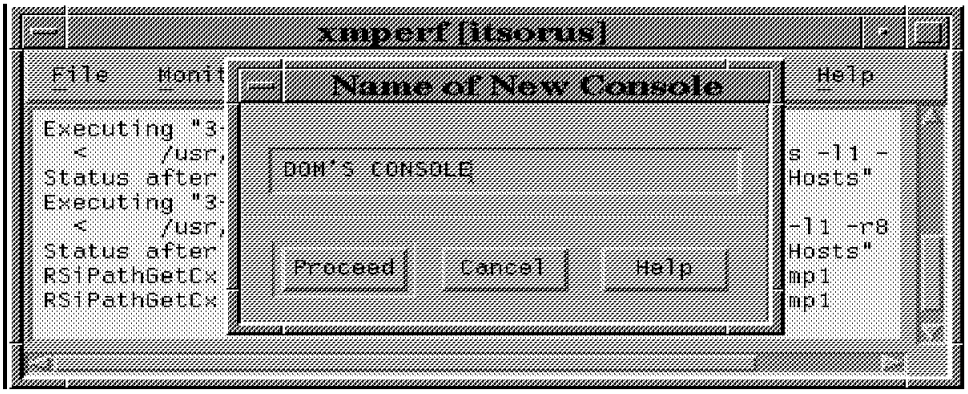


Figure 160. Creating a New Console

Note: In the Monitor pull-down menu, you could select the Instantiate Skeleton option. The Instantiate Skeleton option contains console skeletons of the most common values used for performance monitoring, grouped by categories. The last entry (MP) in the Instantiate Skeleton gives an SMP console with eight instruments. Editing these instruments might be faster than building an SMP console from scratch since each of these predefined consoles can be modified. They provide an easy and fast way to build a console.

Click on the **Proceed** button to continue; you will then get another blank window. Use the Edit Console pull-down menu and select **Add Local Instrument**.

Note: Using Add Remote Instrument means you want to monitor a remote host. If so, the program will ask you to provide the hostname. This remote host must have the performance agent installed. The remote host can be any UP or SMP system.

If you selected Add Local Instrument, you will get the following screen that will invite you to select the statistics you want to monitor within your console.

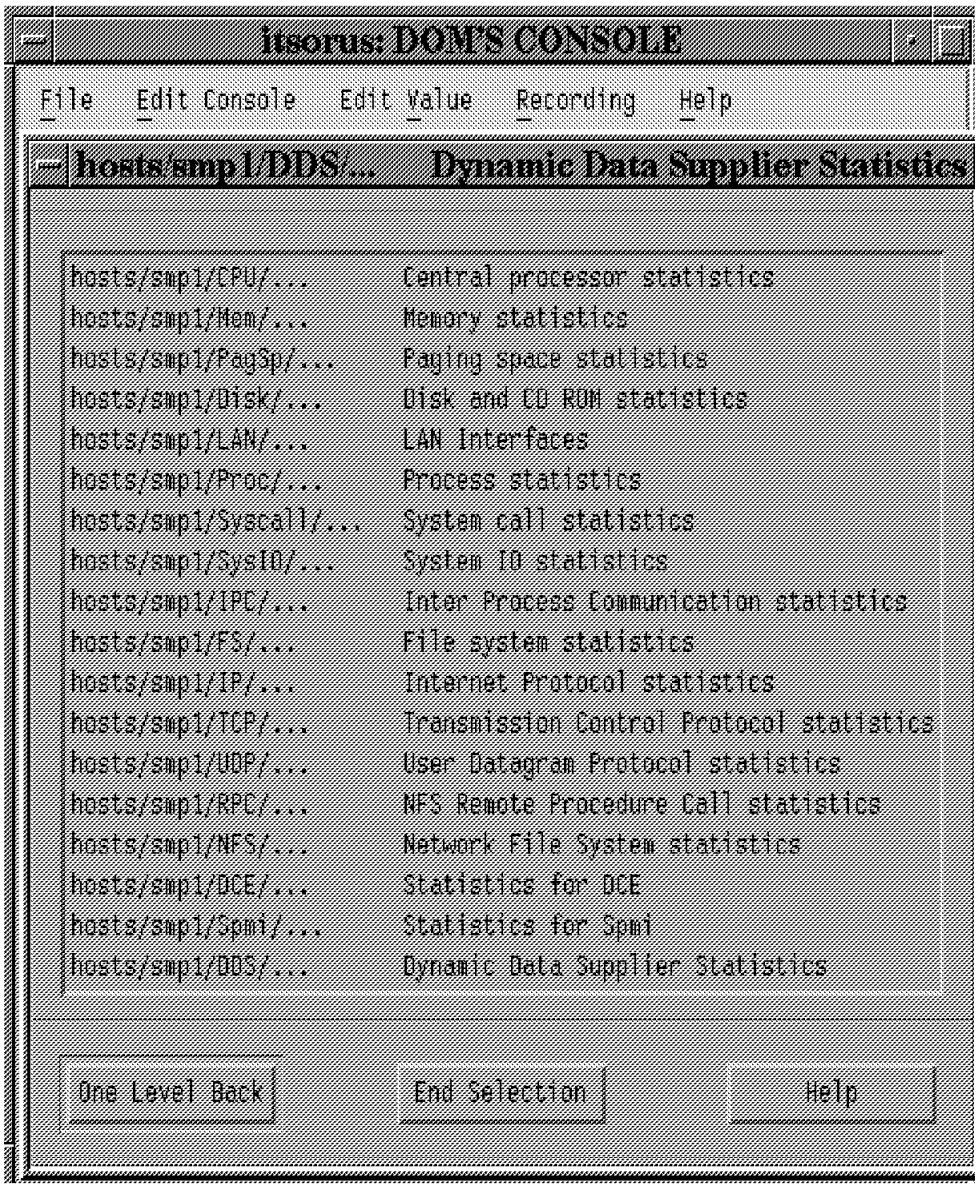


Figure 161. Selecting Central Processor Statistics

For this example, we want to monitor CPU statistics for our SMP; so we selected **Central Processor Statistics.**

You will then get a new window that looks like Figure 162 on page 319.

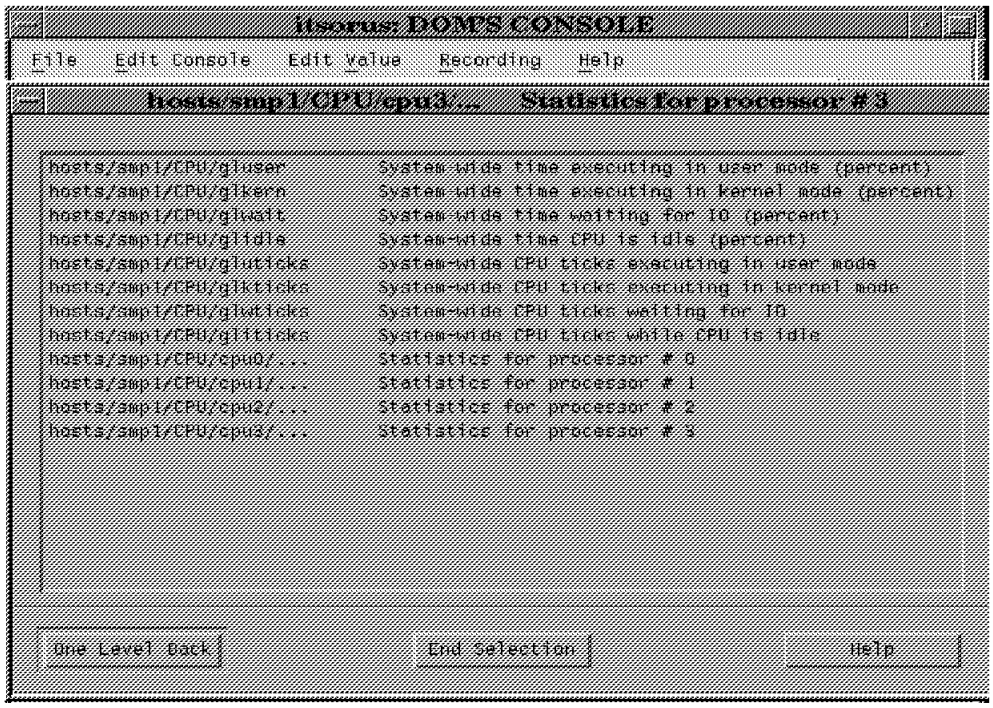


Figure 162. Selecting Statistics

At this point, you can choose global statistics or statistics for a specific processor. You cannot do multiple selections at the same time. You can choose, for example, **Statistics for processor # 0**, and you will then get the following screen:

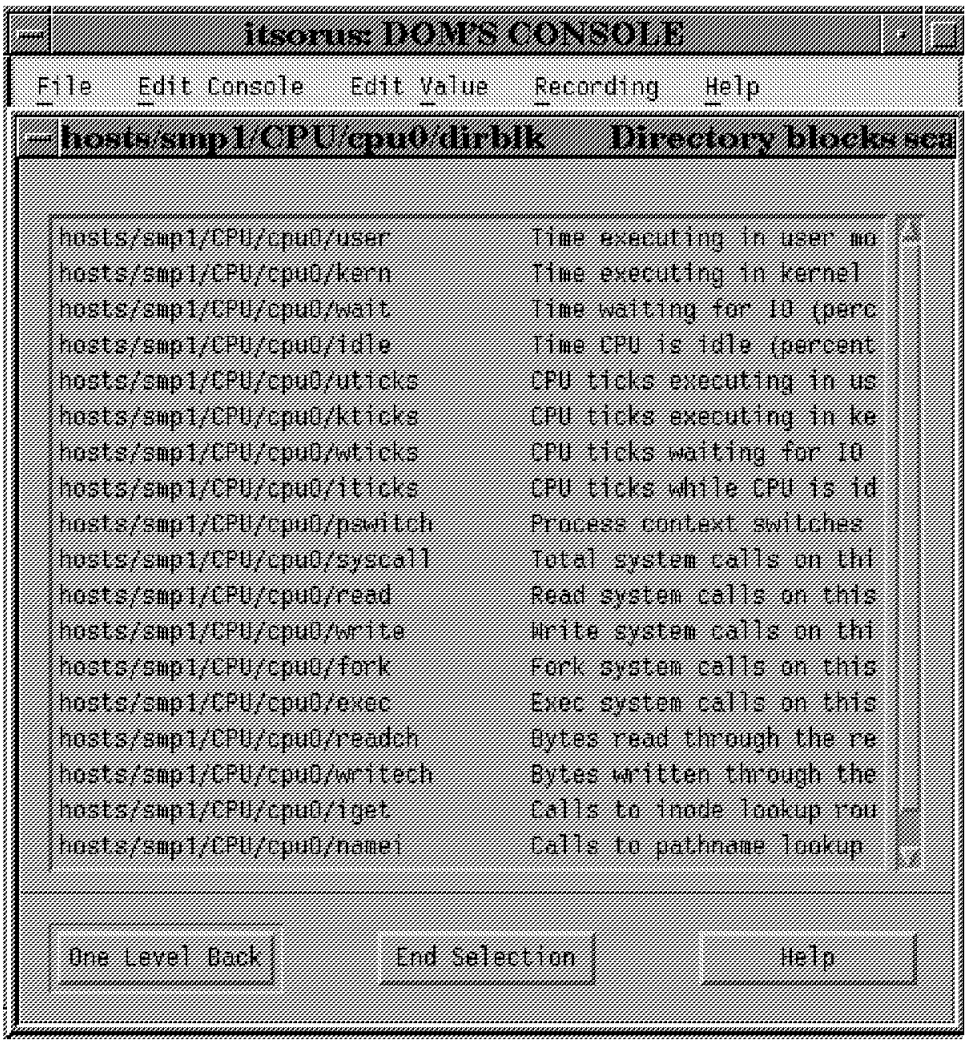


Figure 163. Selecting Statistics for Processor 0

You can then customize the properties of the values you have selected, such as the color and the type of graph you want (line, area, bars). You will be able to set upper and lower limits, set a threshold, and set an alarm when this threshold is reached. Once you are satisfied with the property values, select **Ok**.
Figure 164 on page 321 shows how to customize the properties of the values you want for your instrument.

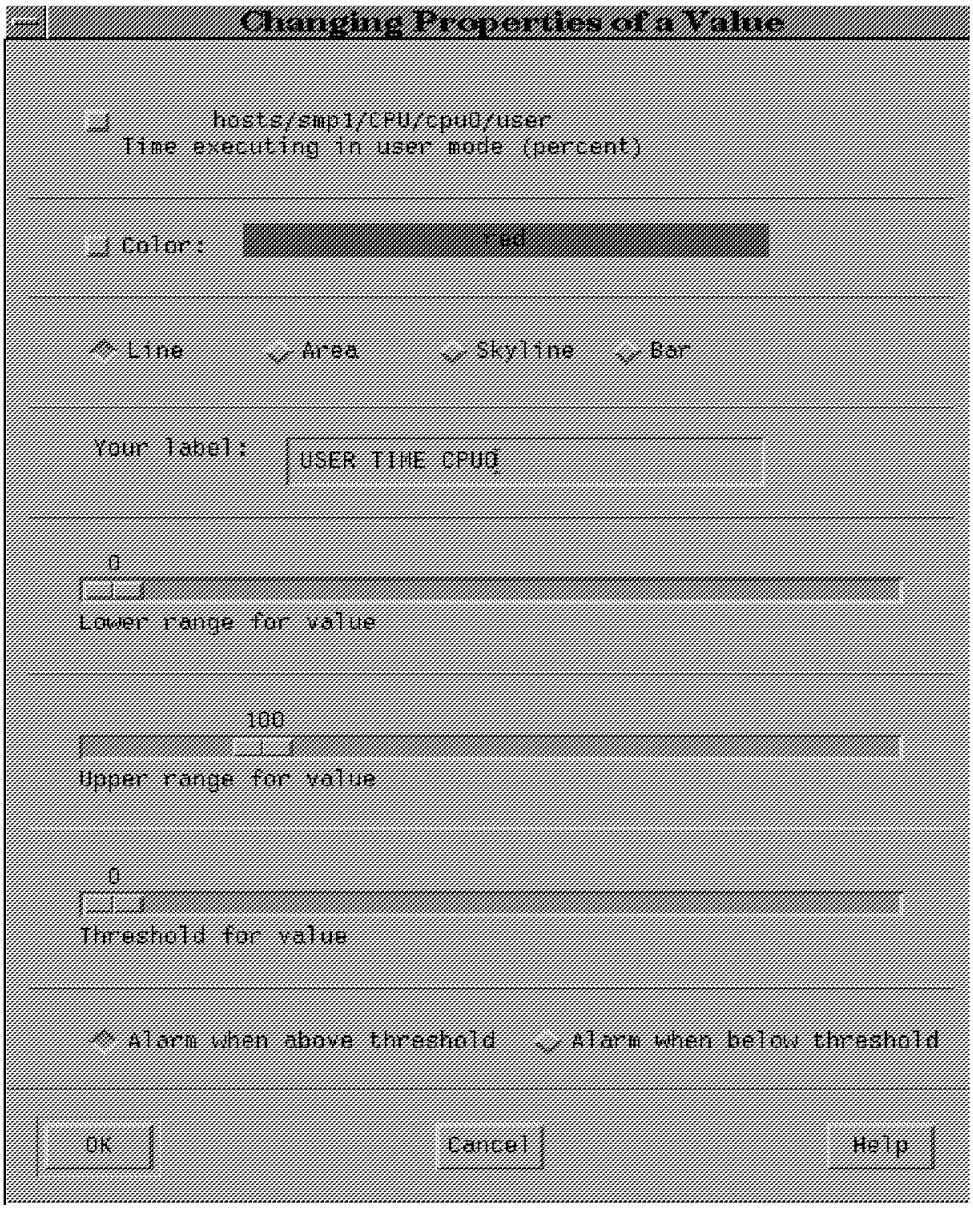


Figure 164. Changing Properties of a Value

Continue selecting values you want to monitor for Processor 0. When you have finished with Processor 0, click on **End of Selection**. You can then edit your console again and add a new local instrument for processor 1, and so on.

Note that you can save your console at any time by selecting the File pull-down menu. This will present you with the option to Save Changes should you wish to keep this console definition to refer to at a later time. Close Console will shut that console down. You can open it again from the initial xmperf menu by using the Monitor pull-down menu. Figure 165 on page 322 shows an example of a customized SMP console.

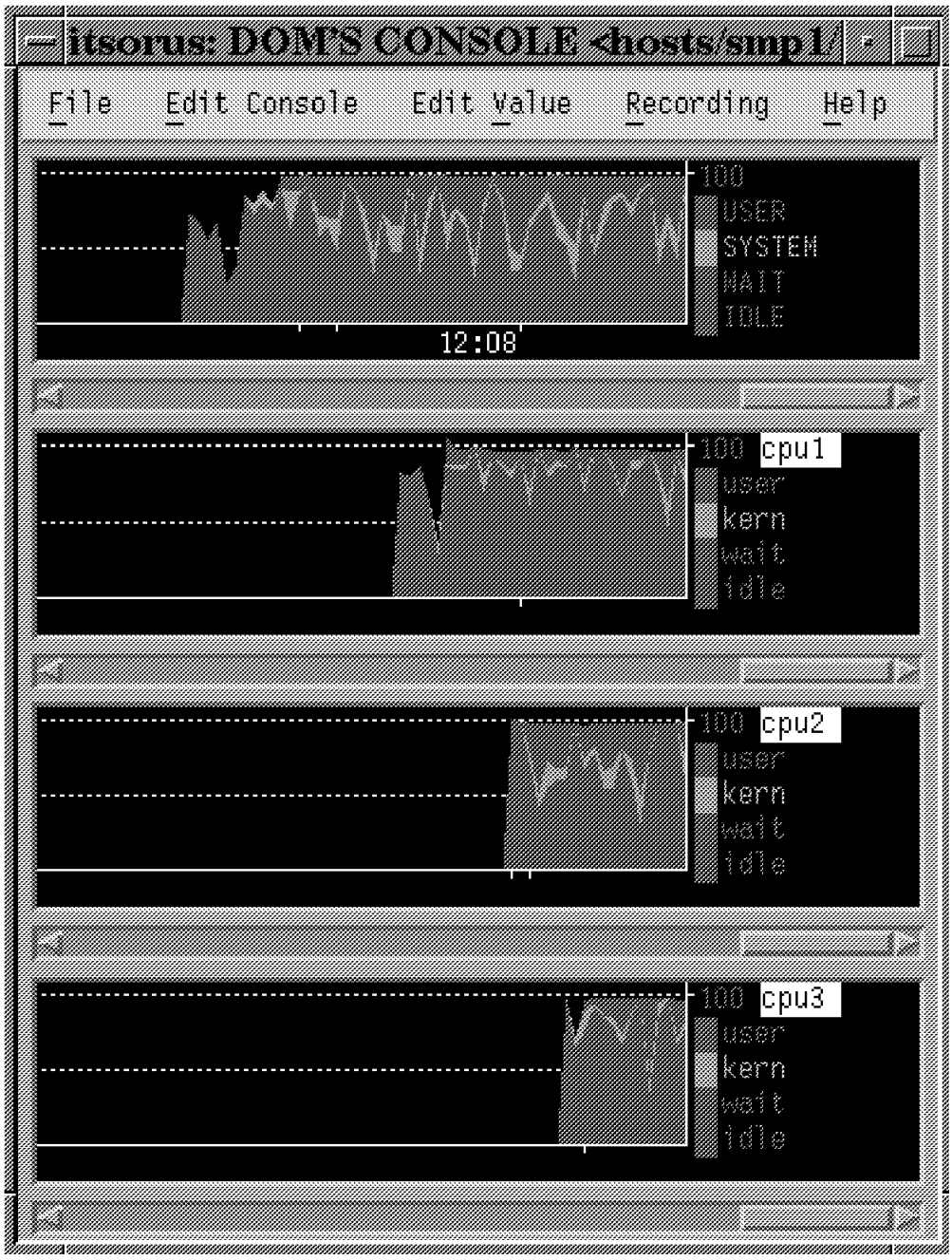


Figure 165. SMP Console Example

10.10.3 Monitoring an SMP with 3dmon

The 3dmon monitor provides a quick method of producing the same results in a three-dimensional view for important performance values.

This monitor may be invoked by going to the utilities menu in the main xmperf window. Inside this menu, you will find both the 3d Monitor Local and 3d Monitor Remote submenus. Choosing the **Local Processors (CPUs)** option will give you a screen where you can choose which CPUs you want to monitor, and when complete, your screen is displayed. The performance values you will be

monitoring are: user, kern, wait, pswitch, syscall, read, write, fork, exec, readch, writech, iget, namei, and dirblk.

The 3dmon monitor may also be invoked from the command line, by typing:

```
# 3dmon -h <hostname>
```

Once you have selected **3D-Monitor** from the Utilities pull-down menu, you will see the following screen:

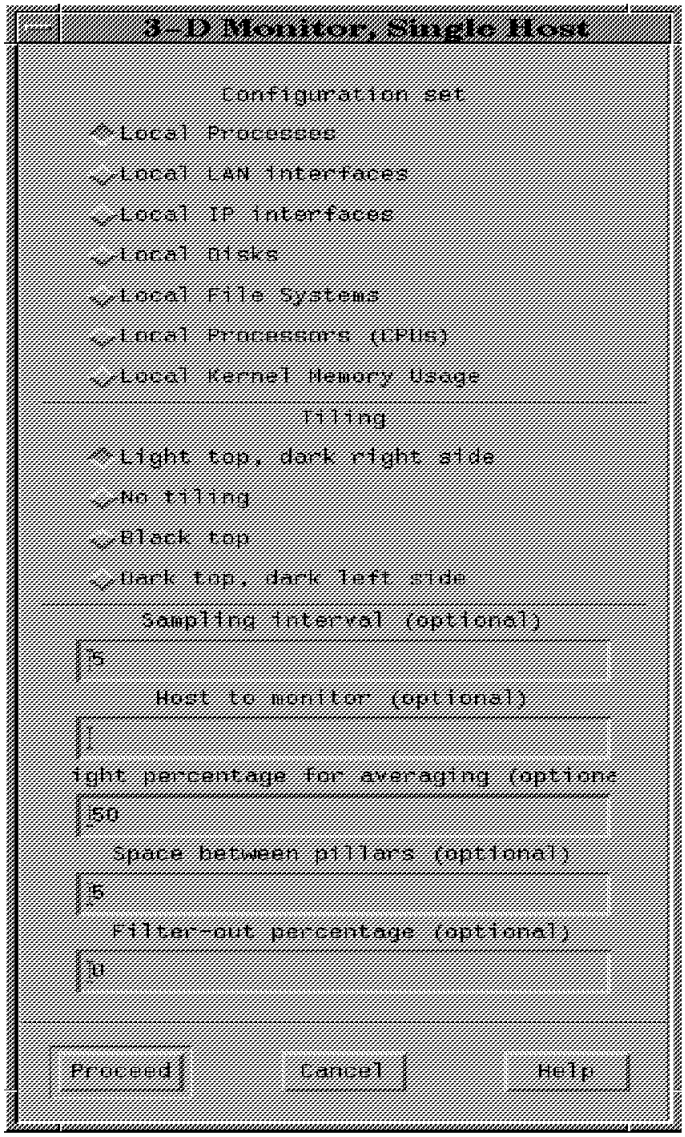


Figure 166. Selecting Local Processors

At this step, you can select resources you want to monitor and change the sampling interval. If you select **Local Processors (CPUs)**, you will then see the following screen:

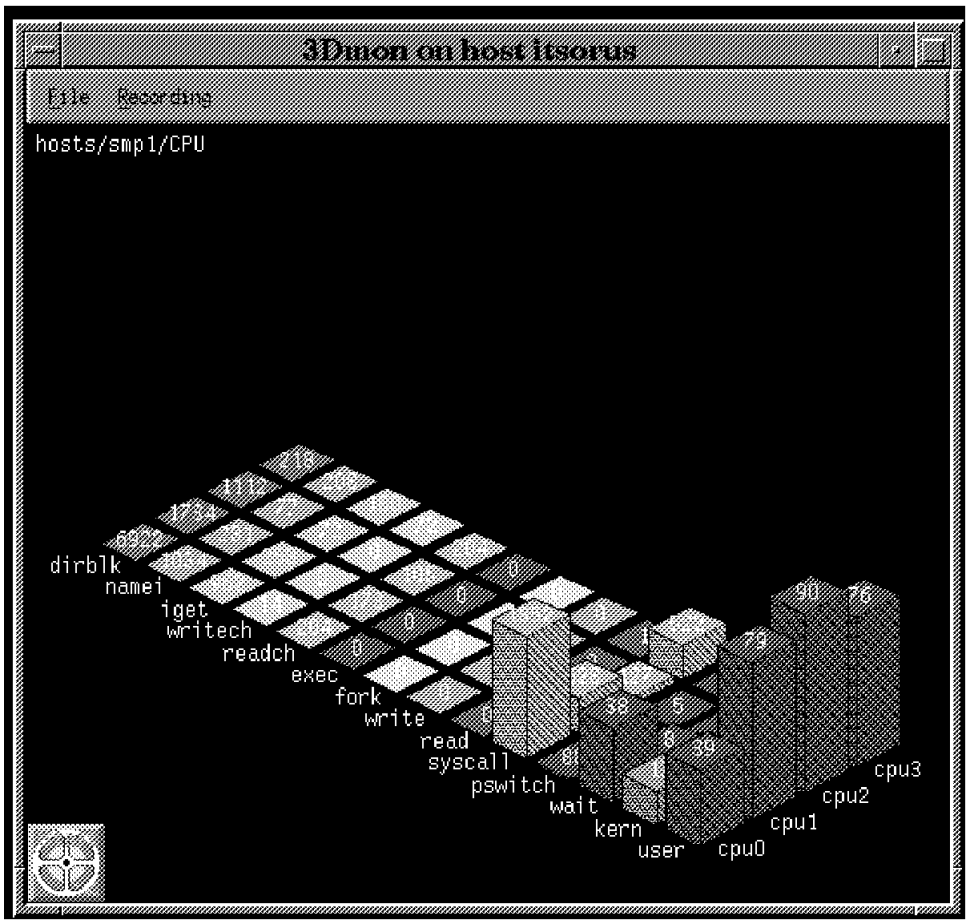


Figure 167. 3dmon Output on a Four-Way SMP

Note: If you cannot read the values behind the first towers corresponding to the user activity, you can move any monitored value to the front by double clicking on the name of that value. For example, if you want to read the kern values for all the processors, you can double click on kern. It will then move to the first position.

Hints

When you modify the xmperf setup, the changes get saved in the xmperf.cf file in your home directory.
If you have created a set of consoles that you wish to use again, it is always advisable to take a copy of this file.
You may also edit this file using a text editor to change console parameters. This is sometimes quicker than using the xmperf menus.
For more details on configuring and using Performance Toolbox, please refer to the manual, *"Performance Toolbox Version 2 for AIX: Guide and Reference",* SC23-2625.

Figure 168. Selecting Central Processor Statistics

Appendix A. SystemGuard Remote Operation Configuration

In order to utilize the remote capabilities of SystemGuard, including console mirroring, you need to have flags, parameters and TTY configurations properly set. Below, you will find tty0 and tty1 settings, sample modem files and all the parameters that are necessary to allow remote operations.

A.1 Terminal Configuration

This is the tty0 configuration for the S1 port. It assumes the tty will have an AIX login and will serve as the AIX system console.

```
[TOP]                                          [Entry Fields]
  TTY                                         tty0
  TTY type                                    tty
  TTY interface                               rs232
  Description                                 Asynchronous Terminal
  Status                                      Available
  Location                                    00-00-S1-00
  Parent adapter                              sa0
  PORT number                                [s1]
  Enable LOGIN                                enable
  BAUD rate                                  [9600]
  PARITY                                     [none]
  BITS per character                         [8]
  Number of STOP BITS                        [1]
  TIME before advancing to next port setting [0]
  TERMINAL type                              [dumb]
  FLOW CONTROL to be used                    [xon]
  OPEN DISCIPLINE to be used                 [dtropen]
  STTY attributes for RUN time               [hupcl,cread,brkint,
                                              icrnl,opost,tab3,
                                              onlcr,isig,icanon,
                                              echo,echoe,echok,
                                              echoctl,echoke,
                                              imaxbel,iexten]
  STTY attributes for LOGIN                  [hupcl,cread,echoe,cs8,
                                              ixon,ixoff]
  LOGGER name                                []
  STATUS of device at BOOT time              [available]
  TRANSMIT buffer count                      [16]
  RECEIVE trigger level                      [3]
  STREAMS modules to be pushed at OPEN time  [ldterm,tioc]
  INPUT map file                             [none]
  OUTPUT map file                            [none]
  CODESET map file                           [sbcs]
```

Figure 169. S1 Port Configuration

This is the tty1 configuration for the S2 port. It also assumes that tty1 will have a login when AIX is running.

```
[TOP]                                          [Entry Fields]
  TTY                                         tty1
  TTY type                                    tty
  TTY interface                               rs232
  Description                                 Asynchronous Terminal
  Status                                      Available
  Location                                    00-00-S2-00
  Parent adapter                              sa1
  PORT number                                [s2]
  Enable LOGIN                                enable
  BAUD rate                                  [9600]
  PARITY                                     [none]
  BITS per character                         [8]
  Number of STOP BITS                        [1]
  TIME before advancing to next port setting [0]
  TERMINAL type                              [dumb]
  FLOW CONTROL to be used                    [xon]
  OPEN DISCIPLINE to be used                 [dtropen]
  STTY attributes for RUN time               [hupcl,cread,brkint,
                                              icrnl,opost,tab3,
                                              onlcr,isig,icanon,
                                              echo,echoe,echok,
                                              echoctl,echoke,
                                              imaxbel,iexten]
  STTY attributes for LOGIN                  [hupcl,cread,echoe,cs8,
                                              ixon,ixoff]
  LOGGER name                                []
  STATUS of device at BOOT time              [available]
  TRANSMIT buffer count                      [16]
  RECEIVE trigger level                      [3]
  STREAMS modules to be pushed at OPEN time  [ldterm,tioc]
  INPUT map file                             [none]
  OUTPUT map file                            [none]
  CODESET map file                           [sbcs]
```

Figure 170. S2 Port Configuration

A.2 Flags and Parameters Settings

These are the minimum SystemGuard parameters/flags settings required for remote support. These parameters can be displayed and changed through AIX with the mpcfg command.

- Modem configuration:

```
mpcfg -dm

Index   Name                                Value

1       Modem Parameters File Name          /usr/share/modems/mir_modem
2       Service Line Speed                  9600
3       Protocol Inter Data Block Delay     5
4       Protocol Time Out                   60
5       Retry Number                        2
6       Customer ID                         9999999
7       Login ID
8       Password ID
```

Figure 171. Modem and Site Configuration Flags

The Modem Parameter File Name value should be set to /usr/share/modems/mir_modem, and that file should be a copy of or linked to your modem description file. Refer to A.4, “Initializing a Modem” on page 331 for one method of doing this. The service line speed must match your modem and tty capabilities (9600 is recommended). If you are enabling dial-out to IBM Service, be sure a valid IBM Customer Number appears in field 6.

- Service flags:

```
mpcfg -dS

Index   Name                                    Value

1       Remote Service Support                  1
2       Quick On Call Service                   0
3       Service Contract Validity               32767
4       Service Support Type
```

Figure 172. Service Support Flags

These are default values provided at the factory. They can only be changed by IBM Service. They are provided here so you’ll know what they are supposed to look like. If they don’t, contact IBM Service.

- Diagnostics flags

```
mpcfg -df

Index   Name                                    Value
1       Remote Authorization                    1
2       Autoservice IPL                         1
3       BUMP Console                            1
4       Dial-Out Authorization                  1
5       Set Mode to Normal When Booting         0
6       Electronic Mode Switch from Service Line  0
7       Boot Multi-user AIX in Service          0
8       Extended Tests                          0
9       Power On Tests in Trace Mode            0
10      Power On Tests in Loop Mode             0
11      Fast IPL                                0
```

Figure 173. Diagnostics Flags

Note that Remote Authorization and Dial-Out Authorization are turned on, assuming the system will attempt to connect to IBM Service if it can’t boot.

- Phone numbers:

```
mpcfg -dp

Index   Name                                    Value

1       Service Center Dial-Out (1)             918008301041
2       Service Center Dial-Out (2)
3       Customer Hub Dial-Out (1)
4       Customer Hub Dial-Out (2)
5       System Dial-In                          5128681234
6       System Operator Voice                   5128685678
```

Figure 174. Phone Number Flags

The phone number in the Service Center Dial-Out field contains the 800 number used in the United States and Canada. Users in other countries would substitute the appropriate number provided by local service representatives. Other phone numbers should be provided based on account-related information. Be sure the dial-out string contains all characters needed to make an outside call, including any number needed to access an outside line. So IBM Service will know whom to call, be sure to fill in field 6, System Operator Voice.

A.3 Modem Configuration Files

If you want to attach a modem to the S2 port to allow remote access, console mirroring and automatic problem reporting from SystemGuard, you will have to provide a configuration file for the modem you will be using. This file is also necessary to utilize the mirroring capabilities supported by the AIX mirrord daemon.

Three modems have been tested, the IBM 7851, IBM 7852 and the USRobotics Sportster 14.4. Below, you will find the corresponding configuration files. You will see that these files have a very specific format. You can use either of these files as a template to build a configuration file for another modem.

If you have a directly attached tty on S2, you will still need a description file. An example of a no-modem file is provided below. It can be found in /usr/share/modems in AIX 4.1 and AIX 4.2, along with sample files for the IBM and USRobotics modems listed above.

This is /usr/share/modems/mir_modem.without.modem for console mirroring using a directly attached tty on S2.

```
ICDelay 1
DefaultTO 10
condout:    done
connect:    done
retry:      done
disconnect: done
condin:     done
condwait:   done
waitcall:   done
page:       done
```

Figure 175. Mirror Without a Modem File

Below is a sample /usr/share/modems/mir_modem file for console mirroring using an IBM 7851 or an IBM 7852 modem.

```
# Tested at 9600bps.

ICDelay 5
DefaultTO 10
CallDelay 120
#  AT  Attention Code           Q0    Enable result codes to screen
# &F1  Set factory profile 1    Q1    Disable result codes to screen
#  E0  Turn echo off            S0=0  Automatic answer inhibit
#  V0  Use numeric responses    S0=2  Answer on second ring
# +++  Escape to command mode  &W0    Save configuration to profile 0
#  H0  Hang-up
# 17=38.4bps; 16=19.2bps; 12=9600bps; 11=4800bps; 10=2400bps; 7=busy
condout:    send "AT&F1E0V0Q0S0=0\r"
            expect "0\r" or "OK\r"
        done

connect:    send "ATDT%N\r"  # Tone dialing command
 expect "17\r" or "16\r" or "12\r" or "11\r" or "10\r" busy "7\r"
 timeout 60
            done

retry:      send "A/"        # Redo command
 expect "17\r" or "16\r" or "12\r" or "11\r" or "10\r" busy "7\r"
 timeout 60
            done

disconnect: send "+++ATH0\r"
            delay 2
            send "ATQ1V0E0\r"
            delay 2
        done

condin:         send "AT&F1E0V0Q0S0=2\r"
            expect "0\r" or "OK\r\n"
            send "ATQ1&W0\r"    # (there can be no reply)
        done

condwait:   send "AT&F1V0E0Q0S0=2&W0\r"
            expect "0\r" or "OK\r\n"
            done

waitcall:   ignore "2\r" timeout none
            expect "2\r" timeout 10
 expect "17\r" or "16\r" or "12\r" or "11\r" or "10\r" busy "7\r"
 timeout 60
            done

page:       send "ATDT%N;\r"  # ; = go back to command mode
            expect "0\r" or "OK\r\n" timeout 60
            delay 2
            send "ATH0\r"
            expect "0\r" or "OK\r\n"
            done
```

Figure 176. IBM 7851 and 7852 Modem Configuration File

This is a sample /usr/share/modems/mir_modem file for console mirroring using a USRobotics Sportster 14.4 modem.

```
#Tested at 9600bps.
# Physical switch settings on modem should be: 1-2 up; 3 down; 4-7 up;
#8 down.

ICDelay 5
DefaultTO 10
CallDelay 120
#  AT  Attention Code
# &F1  Set factory profile 1    Q0   Turn on responses
#  E0  Turn echo off            Q1   Turn off responses
#  V0  Use numeric responses    S0=0 Automatic answer inhibit
# +++  Escape to command mode   S0=1 Answer on first ring
#  H0  Hang-up                 &W0 Save configuration to profile 0
# 37=9600/ARQ/V32; 26=14.4ARQ; 25=14.4bps; 19=4800ARQ; 18=4800bps;
# 17=9600ARQ; 13=9600bps;7=busy
condout:    send "AT&F1E0V0Q0S0=0\r"
            expect "0\r"
        done

connect:    send "ATDT%N\r"  # Tone dialing command
 expect "37\r" or "17\r" or "13\r" or "19\r" or "18\r" busy "7\r"
 timeout 60
            done

retry:      send "A/"        # Redo command
 expect "37\r" or "17\r" or "13\r" or "19\r" or "18\r" busy "7\r"
 timeout 60
            done

disconnect:
        send "+++ATH0"
        delay 2
        send "ATQ1V0E0\r"
            delay 2
        done

condin:         send "AT&F1E0V0Q0S0=1\r"
            expect "0\r" or "OK\r\n"
            send "ATQ1&W0\r"    # (there can be no reply)
        done

condwait:   send "AT&F1E0V0Q0S0=1&W0\r"
        expect "0\r" or "OK\r\n"
        done

waitcall: ignore "2\r" timeout none
          expect "2\r" timeout 10
expect "37\r" or "17\r" or "13\r" or "19\r" or "18\r" busy "7\r"
# timeout 60
            done

page:       send "ATDT%N;\r"  # ; = go back to command mode
            expect "0\r" or "OK\r\n" timeout 60
            delay 2
            send "ATH0\r"
            expect "0\r" or "OK\r\n"
            done
```

Figure 177. USRobotics 14.4 Sportster Modem Configuration File

A.4 Initializing a Modem

Once flags, parameters and configurations have been set, the modem can be initialized to accept incoming calls and dial-out. This can be done in the following manner:

- Place the System Key to Normal.
- Change directory to /usr/share/modems
- Issue a ps -ef|grep mirrord command.
- Obtain mirrord process ID.
- Issue a kill -9 <mirrord_pid>.
- Link /usr/share/modems/mir_modem to the correct description file, for example by issuing the command:

```
ln -f mir_modem.7851 mir_modem
```

- Enter /usr/sbin/mirrord mir_modem
- Place the System Key into the Service position.

Placing the System Key in Service will awaken the mirrord process. The disconnect and condin parameters will be read from the mir_modem file, and the modem will be initialized for dial-in activity. You should see the message

```
mirrord: Wait connection...
```

After the mirrord daemon is activated, the System Key should be placed in the Normal position, stopping mirroring.

A.5 Testing Dial-Out

Dial-out or automatic problem reporting can be tested from the SystemGuard Maintenance Menu using the Off-line Test for Dial-out. During the test, the modem configuration file will be read, the modem initialized and data transmitted.

Follow these procedures to test the modem setup:

1. Go the SystemGuard Maintenance Menu and select option 7, OFF-LINE TESTS
2. From the OFF-LINE TESTS Menu, first select A, Verbose mode, and then select 0, BUILD TEST LIST
3. From the BUILD TEST LIST Menu, select Group 01, BUMP QUICK IO
4. From the BUILD TEST LIST, select 53, AUTODIAL
5. On the next screen, select E to enter the test, then go back to the OFF-LINE TESTS Menu
6. Confirm that you've selected the AUTODIAL for testing by entering 1, MODIFY/DISPLAY TEST LIST
7. After you've confirmed that you're going to test AUTODIAL, enter 3 to execute the test

You should see a >>> WARNING <<< message advising you that executing off-line tests will modify the memory/CPU hardware setup. It will ask you if you want to proceed, and you enter **y** for yes.

Then, if all goes well, you'll see the following screen:

```
                    EXECUTE TEST LIST
OPTION LIST: Verbose
PROCESSOR LIST: Test Parameter Value

     >>> ENTER (CTRL_X) TO INTERRUPT TEST EXECUTION <<<

TEST 0153 RUNNING
     .. BUMP [01.53.00] DIAL-OUT TEST             OK

PRESS RETURN TO EXIT
```

Figure 178. Successful Off-line Modem Test

A.6 Sample Problem Record Opened by SystemGuard

If your system can't boot because it has no processors configured, no available memory or a problem with one of the buses, and if it's set up for dial-out, SystemGuard can notify IBM Service that you have a problem.

The problem record created is queued to IBM Service personnel who will then contact the customer to find out if they need assistance. Below is a sample problem record generated automatically when the system attempted a normal boot with no available processors. Automatic problem reporting is in place in the United States and Canada. Since the process depends on a "catcher" system with connections to RETAIN, you should check with your country or geography Service personnel to see if SystemGuard error reporting has been implemented locally.

```
947X4 CL1L3 SV__ <-PROB-STAT-SV-S1 B004 C00 A99 R <-B/O H * 11:10
SERVDR 3 DIA p3_ <-QUE-LVL-CAT-PPG 86908__________<-GP1 *
L050 000 USA ___ <-CTR-CTRY-BU-PRS 86926__________<-CP2 *
D/T7013J30 0045067<-DEV-SER<-CON
__________________<-ALIAST___ NET/_________<-TERR-NET
_________ _______<-T/D-ECT0000 0000000<-CPU
_____ ____9999999 <-NQ/C-C# IBM INTERNAL__________<-CUST XCL
SUBMITTED BY SYSGUARD_____________________________

MPI RECORD FOUND, STATUS = IU
FATAL ERROR IN POWER_ON TESTS
PROBLEM REPORTED BY SYSGUARD SYSTEM
SRN: 888-103-409-020 00-0P 00-0Q
Login-ID:
Password general:
Time Stamp: 15:20:38
Primary Error Code: 0102
Uname:00045067
```

Figure 179. Sample Problem Record

Appendix B. Sample Programs

This appendix includes a number of sample programs that can be used for becoming familiar with the performance tools in an SMP environment and for observing multithreaded applications.

100unbound	A process with four unbound threads.
100bound	A process with four threads, each one bound to a processor.
100boundon1	A process with four threads all bound to the same processor.
100boundon2	A process with four threads bound to two processors.
big_copy	An I/O intensive program.
cpubound	A CPU intensive program bound to a single processor.
4everunbound	A process with four unbound threads runnning forever.
3everunbound	A process with three unbound threads running forever.
4everboundon4	A process with four threads running forever, each one bound to a processor.
4everboundon2	A process with four threads running forever, bound to two processors.
4everboundon1	A process with four threads running forever, bound to the same processor.
pstat_disp	A simple script to repeatedly execute the pstat command.
Makefile	Required to build the sample programs.

For IBM employees, the source code of these samples can be obtained by entering the following command on a VM system:

```
TOOLS SENDTO WTSCPOK TOOLS AIXDISK GET AIXV4SMP PACKAGE
```

B.1 100unbound

```
/* This process has 4 threads */
/* Threads are unbound */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>


int counter;
int loop;
pthread_mutex_t m;

void *Thread(void *string)
{
    int l;
    int r=0;

    while (loop<100)
    {
            l=0;
```

```
            while (l < 1000000)
                        l++;

            pthread_mutex_lock(&m);
                    counter++;
/*                  printf("%s %d\n", (char*) string, counter);  */
                    loop++;
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);

}


int main()
{
    char *e_str = "Thread One   !";
    char *f_str = "Thread Two   !";
    char *g_str = "Thread Three !";
    char *h_str = "Thread Four  !";

    pthread_t e_th;
    pthread_t f_th;
    pthread_t g_th;
    pthread_t h_th;
    int rc;

    pthread_mutex_init(&m, NULL);

    rc = pthread_create(&e_th, NULL, Thread, e_str);
    if (rc)
    {
            printf("Error 1\n");
            exit(-1);
    }

    rc = pthread_create(&f_th, NULL, Thread, f_str);
    if (rc) {
            printf("Error 2\n");
            exit(-1);
    }

    rc = pthread_create(&g_th, NULL, Thread, g_str);
    if (rc) {
            printf("Error 3\n");
            exit(-1);
    }

    rc = pthread_create(&h_th, NULL, Thread, h_str);
    if (rc)
    {
            printf("Error 4\n");
            exit(-1);
    }

    pthread_exit(0);
}
```

B.2 100bound

```
/* This process has 4 threads */
/* Each thread is bound to a processor */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>


int counter;
int loop;
pthread_mutex_t m;

struct arg {
    char *string;
    int     id;
};
typedef struct arg arg_t;

void *Thread(void *x)
{
    int l;
    int r=0;
    int ret;

    ret =bindprocessor(BINDTHREAD, thread_self(), ((arg_t*) x)->id);

/*  printf("Processor No = %d: Thread id= %d: Return value from bind=%d\n",
((arg_t*) x)->id, thread_self(), ret); */

    if(ret==-1){
            perror("bind");
            pthread_exit( (void*)-1);
    }


    while (loop < 100)
    {
            l=0;
            while (l < 1000000)
                            l++;

            pthread_mutex_lock(&m);
                    counter++;
/*                  printf("%s %d\n", (char*) x->string, counter);  */
                    loop++;
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);

}


int main()
{
    char *e_str = "Thread One   !";
    char *f_str = "Thread Two   !";
```

```
    char *g_str = "Thread Three !";
    char *h_str = "Thread Four  !";
    struct arg arg1, arg2, arg3, arg4;

    pthread_t e_th;
    pthread_t f_th;
    pthread_t g_th;
    pthread_t h_th;
    int rc;

    printf("\n");

    pthread_mutex_init(&m, NULL);

    arg1.string = e_str;
    arg1.id = 0;
    rc = pthread_create(&e_th, NULL, Thread, &arg1);
    if (rc)
    {
            printf("Error 1\n");
            exit(-1);
    }

    arg2.string = f_str;
    arg2.id = 1;
    rc = pthread_create(&f_th, NULL, Thread, &arg2);
    if (rc) {
            printf("Error 2\n");
            exit(-1);
    }

    arg3.string = g_str;
    arg3.id = 0;
    rc = pthread_create(&g_th, NULL, Thread, &arg3);
    if (rc) {
            printf("Error 3\n");
            exit(-1);
    }

    arg4.string = h_str;
    arg4.id = 1;
    rc = pthread_create(&h_th, NULL, Thread, &arg4);
    if (rc) {
            printf("Error 4\n");
            exit(-1);
    }

    pthread_exit(0);
}
```

B.3 100boundon1

```
/* This process has 4 threads */
/* All threads are bound to the same processor */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>
```

```
int counter;
int loop;
pthread_mutex_t m;

struct arg {
    char *string;
    int     id;
};
typedef struct arg arg_t;

void *Thread(void *x)
{
    int l;
    int r=0;
    int ret;

    ret =bindprocessor(BINDTHREAD, thread_self(), ((arg_t *) x)->id);

/*  printf("Processor No = %d: Thread id= %d: Return value from bind=%d\n",
((arg_t *) x)->id, thread_self(), ret); */

    if(ret==-1){
            perror("bind");
            pthread_exit( (void*)-1);
    }


    while (loop < 100)
    {
            l=0;
            while (l < 1000000)
                            l++;

            pthread_mutex_lock(&m);
                    counter++;
/*                  printf("%s %d\n", (char*) x->string, counter);  */
                    loop++;
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);

}


int main()
{
    char *e_str = "Thread One   !";
    char *f_str = "Thread Two   !";
    char *g_str = "Thread Three !";
    char *h_str = "Thread Four  !";
    struct arg arg1, arg2, arg3, arg4;

    pthread_t e_th;
    pthread_t f_th;
    pthread_t g_th;
    pthread_t h_th;
```

```
    int rc;

    printf("\n");

    pthread_mutex_init(&m, NULL);

    arg1.string = e_str;
    arg1.id = 2;
    rc = pthread_create(&e_th, NULL, Thread,  &arg1);
    if (rc)
    {
            printf("Error 1\n");
            exit(-1);
    }

    arg2.string = f_str;
    arg2.id = 2;
    rc = pthread_create(&f_th, NULL, Thread,  &arg2);
    if (rc) {
            printf("Error 2\n");
            exit(-1);
    }

    arg3.string = g_str;
    arg3.id = 2;
    rc = pthread_create(&g_th, NULL, Thread,  &arg3);
    if (rc) {
            printf("Error 3\n");
            exit(-1);
    }

    arg4.string = h_str;
    arg4.id = 2;
    rc = pthread_create(&h_th, NULL, Thread,  &arg4);
    if (rc) {
            printf("Error 4\n");
            exit(-1);
    }

    pthread_exit(0);
}
```

B.4 100boundon2

```
/* This process has 4 threads */
/* Threads are bound to 2 processors */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>


int counter;
int loop;
pthread_mutex_t m;

struct arg {
    char *string;
```

```
    int     id;
};
typedef struct arg arg_t;

void *Thread(void *x)
{
    int l;
    int r=0;
    int ret;

    ret =bindprocessor(BINDTHREAD, thread_self(), ((arg_t*) x)->id);

/*  printf("Processor No = %d: Thread id= %d: Return value from bind=%d\n",
((arg_t*) x)->id, thread_self(), ret); */

    if(ret==-1){
            perror("bind");
            pthread_exit( (void*)-1);
    }


    while (loop < 100)
    {
            l=0;
            while (l < 1000000)
                        l++;

            pthread_mutex_lock(&m);
                    counter++;
/*                  printf("%s %d\n", (char*) x->string, counter);  */
                    loop++;
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);

}


int main()
{
    char *e_str = "Thread One   !";
    char *f_str = "Thread Two   !";
    char *g_str = "Thread Three !";
    char *h_str = "Thread Four  !";
    struct arg arg1, arg2, arg3, arg4;

    pthread_t e_th;
    pthread_t f_th;
    pthread_t g_th;
    pthread_t h_th;
    int rc;

    printf("\n");

    pthread_mutex_init(&m, NULL);

    arg1.string = e_str;
    arg1.id = 0;
```

```
    rc = pthread_create(&e_th, NULL, Thread, &arg1);
    if (rc)
    {
            printf("Error 1\n");
            exit(-1);
    }

    arg2.string = f_str;
    arg2.id = 1;
    rc = pthread_create(&f_th, NULL, Thread, &arg2);
    if (rc) {
            printf("Error 2\n");
            exit(-1);
    }

    arg3.string = g_str;
    arg3.id = 0;
    rc = pthread_create(&g_th, NULL, Thread, &arg3);
    if (rc) {
            printf("Error 3\n");
            exit(-1);
    }

    arg4.string = h_str;
    arg4.id = 1;
    rc = pthread_create(&h_th, NULL, Thread, &arg4);
    if (rc) {
            printf("Error 4\n");
            exit(-1);
    }

    pthread_exit(0);
}
```

B.5 cpubound

```
/* This process has one thread bound to one processor */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>

int counter;
pthread_mutex_t m;

struct arg {
    char *string;
    int     id;
};
typedef struct arg arg_t;

void *Thread(void *x)
{
    int l;
    int r=0;
    int ret;

    ret =bindprocessor(BINDTHREAD, thread_self(), ((arg_t*)x)->id);
```

```
/*  printf("Processor ID = %d: Thread id= %d: Return value from bind=%d\n",
((arg_t*)x)->id, thread_self(), ret); */

    if(ret==-1){
            perror("bind");
            pthread_exit( (void *)-1);
    }

    while (1)
    {
            l=0;
            while (l < 1000000)
                            l++;

            pthread_mutex_lock(&m);
                    counter++;
    /*              printf("%s %d\n", (char*) x->string, counter);  */
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);
}

int main()
{
    char *e_str = "Thread One !";
    struct arg arg1;

    pthread_t e_th;
    int rc;

    pthread_mutex_init(&m, NULL);

    arg1.string = e_str;
    arg1.id = 0;
    rc = pthread_create(&e_th, NULL, Thread, &arg1);
    if (rc)
    {
            printf("Error 1\n");
            exit(-1);
    }

    pthread_exit(0);
}
```

B.6 4everunbound

```
/* This process has 4 threads running forever */
/* All threads are unbound */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>

int counter;
pthread_mutex_t m;
```

```
void *Thread(void *string)
{
    int l;
    int r=0;

    while (1)
    {
            l=0;
            while (l < 3000000)
                        l++;

            pthread_mutex_lock(&m);
                    counter++;
/*                  printf("%s %d\n", (char*) string, counter);   */
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);
}

int main()
{
    char *e_str = "Thread One   !";
    char *f_str = "Thread Two   !";
    char *g_str = "Thread Three !";
    char *h_str = "Thread Four  !";

    pthread_t e_th;
    pthread_t f_th;
    pthread_t g_th;
    pthread_t h_th;
    int rc;

    pthread_mutex_init(&m, NULL);

    rc = pthread_create(&e_th, NULL, Thread, e_str);
    if (rc)
    {
            printf("Error 1\n");
            exit(-1);
    }

    rc = pthread_create(&f_th, NULL, Thread, f_str);
    if (rc) {
            printf("Error 2\n");
            exit(-1);
    }

    rc = pthread_create(&g_th, NULL, Thread, g_str);
    if (rc) {
            printf("Error 3\n");
            exit(-1);
    }

    rc = pthread_create(&h_th, NULL, Thread, h_str);
    if (rc)
    {
            printf("Error 4\n");
            exit(-1);
```

```
    }

    pthread_exit(0);
}
```

B.7 3everunbound

```
/* This process has 3 threads running forever */
/* All threads are unbound */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>


int counter;
pthread_mutex_t m;

void *Thread(void *string)
{
    int l;
    int r=0;

    while (1)
    {
            l=0;
            while (l < 3000000)
                        l++;

            pthread_mutex_lock(&m);
                    counter++;
/*                  printf("%s %d\n", (char*) string, counter);   */
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);

}


int main()
{
    char *e_str = "Thread One   !";
    char *f_str = "Thread Two   !";
    char *g_str = "Thread Three !";

    pthread_t e_th;
    pthread_t f_th;
    pthread_t g_th;
    int rc;

    pthread_mutex_init(&m, NULL);

    rc = pthread_create(&e_th, NULL, Thread, e_str);
    if (rc) {
            printf("Error 1\n");
            exit(-1);
    }
```

```
    rc = pthread_create(&f_th, NULL, Thread, f_str);
    if (rc) {
            printf("Error 2\n");
            exit(-1);
    }

    rc = pthread_create(&g_th, NULL, Thread, g_str);
    if (rc)
    {
            printf("Error 3\n");
            exit(-1);
    }

    pthread_exit(0);
}
```

B.8 4everboundon4

```
/* This process has 4 threads running forever */
/* Each thread is bound to a processor */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>


int counter;
pthread_mutex_t m;

struct arg {
    char *string;
    int     id;
};
typedef struct arg arg_t;

void *Thread(void *x)
{
    int l;
    int r=0;
    int ret;

    ret =bindprocessor(BINDTHREAD,thread_self(), ((arg_t*) x)->id);

/*  printf("Processor No = %d: Thread ID = %d: Return value from bind=%d \n"
,((arg_t*)x)->id, thread_self(), ret);*/

    if(ret==-1){
            perror("bind");
            pthread_exit( (void*)-1);
    }


    while (1)
    {
            l=0;
            while (l < 1000000)
                            l++;
```

```
            pthread_mutex_lock(&m);
                    counter++;
/*                  printf("%d Thread ID = %d \n", counter, ((arg_t*)x)->id)
;  */
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);

}


int main()
{
    char *e_str = "Thread One   !";
    char *f_str = "Thread Two   !";
    char *g_str = "Thread Three !";
    char *h_str = "Thread Four  !";
    struct arg arg1, arg2, arg3, arg4;

    pthread_t e_th;
    pthread_t f_th;
    pthread_t g_th;
    pthread_t h_th;
    int rc;

    printf("\n");

    pthread_mutex_init(&m, NULL);

    arg1.string = e_str;
    arg1.id = 0;
    rc = pthread_create(&e_th, NULL, Thread, &arg1);
    if (rc)
    {
            printf("Error 1\n");
            exit(-1);
    }

    arg2.string = f_str;
    arg2.id = 1;
    rc = pthread_create(&f_th, NULL, Thread, &arg2);
    if (rc) {
            printf("Error 2\n");
            exit(-1);
    }

    arg3.string = g_str;
    arg3.id = 2;
    rc = pthread_create(&g_th, NULL, Thread, &arg3);
    if (rc) {
            printf("Error 3\n");
            exit(-1);
    }

    arg4.string = h_str;
    arg4.id = 3;
    rc = pthread_create(&h_th, NULL, Thread, &arg4);
```

```
    if (rc) {
            printf("Error 4\n");
            exit(-1);
    }

    pthread_exit(0);
}
```

B.9 4everboundon2

```
/* This process has 4 threads running forever */
/* Threads are bound to 2 processors */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>


int counter;
pthread_mutex_t m;

struct arg {
    char *string;
    int     id;
};
typedef struct arg arg_t;

void *Thread(void *x)
{
    int l;
    int r=0;
    int ret;

    ret =bindprocessor(BINDTHREAD, thread_self(), ((arg_t*) x)->id);

/*  printf("Processor No = %d, Thread id = %d: Return value from bind=%d\n"
, ((arg_t*) x)->id, thread_self(), ret);*/

    if(ret==-1){
            perror("bind");
            pthread_exit( (void*)-1);
    }

    while (1)
    {
            l=0;
            while (l < 1000000)
                            l++;

            pthread_mutex_lock(&m);
                    counter++;
/*                  printf("%s %d\n", (char*) x->string, counter);  */
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);

}
```

```
int main()
{
    char *e_str = "Thread One   !";
    char *f_str = "Thread Two   !";
    char *g_str = "Thread Three !";
    char *h_str = "Thread Four  !";
    struct arg arg1, arg2, arg3, arg4;

    pthread_t e_th;
    pthread_t f_th;
    pthread_t g_th;
    pthread_t h_th;
    int rc;

    pthread_mutex_init(&m, NULL);

    arg1.string = e_str;
    arg1.id = 2;
    rc = pthread_create(&e_th, NULL, Thread, &arg1);
    if (rc)
    {
            printf("Error 1\n");
            exit(-1);
    }

    arg2.string = f_str;
    arg2.id = 3;
    rc = pthread_create(&f_th, NULL, Thread, &arg2);
    if (rc) {
            printf("Error 2\n");
            exit(-1);
    }

    arg3.string = g_str;
    arg3.id = 2;
    rc = pthread_create(&g_th, NULL, Thread, &arg3);
    if (rc) {
            printf("Error 3\n");
            exit(-1);
    }

    arg4.string = h_str;
    arg4.id = 3;
    rc = pthread_create(&h_th, NULL, Thread, &arg4);
    if (rc) {
            printf("Error 4\n");
            exit(-1);
    }

    pthread_exit(0);
}
```

B.10 4everboundon1

```
/* This process has four threads running forever */
/* Threads are bound to one processor */

# include <pthread.h>
# include <stdio.h>
# include <unistd.h>


int counter;
int loop;
pthread_mutex_t m;

struct arg {
    char *string;
    int     id;
};
typedef struct arg arg_t;

void *Thread(void *x)
{
    int l;
    int r=0;
    int ret;

    ret =bindprocessor(BINDTHREAD, thread_self(), ((arg_t*) x)->id);

/*  printf("Processor No = %d: Thread id= %d: Return value from bind=%d\n",
((arg_t*) x)->id, thread_self(), ret); */

    if(ret==-1){
            perror("bind");
            pthread_exit( (void*)-1);
    }


    while (1)
    {
            l=0;
            while (l < 1000000)
                            l++;

            pthread_mutex_lock(&m);
                    counter++;
/*                  printf("%s %d\n", (char*) x->string, counter);  */
                    loop++;
            pthread_mutex_unlock(&m);

    }
    pthread_exit( (void *)1);

}


int main()
{
    char *e_str = "Thread One   !";
    char *f_str = "Thread Two   !";
```

```
    char *g_str = "Thread Three !";
    char *h_str = "Thread Four  !";
    struct arg arg1, arg2, arg3, arg4;

    pthread_t e_th;
    pthread_t f_th;
    pthread_t g_th;
    pthread_t h_th;
    int rc;

    printf("\n");

    pthread_mutex_init(&m, NULL);

    arg1.string = e_str;
    arg1.id = 2;
    rc = pthread_create(&e_th, NULL, Thread, &arg1);
    if (rc)
    {
            printf("Error 1\n");
            exit(-1);
    }

    arg2.string = f_str;
    arg2.id = 2;
    rc = pthread_create(&f_th, NULL, Thread, &arg2);
    if (rc) {
            printf("Error 2\n");
            exit(-1);
    }

    arg3.string = g_str;
    arg3.id = 2;
    rc = pthread_create(&g_th, NULL, Thread, &arg3);
    if (rc) {
            printf("Error 3\n");
            exit(-1);
    }

    arg4.string = h_str;
    arg4.id = 2;
    rc = pthread_create(&h_th, NULL, Thread, &arg4);
    if (rc) {
            printf("Error 4\n");
            exit(-1);
    }

    pthread_exit(0);
}
```

B.11 big_copy

```
/* This process creates intensive I/O activity */

main ()
{
while (1)
{
system ("cp /unix /perf/bigfile");
system ("cp /perf/bigfile /perf/bigfile1");
system ("rm /perf/bigfile1");
system ("rm /perf/bigfile");
sleep (1);
}
}
```

B.12 pstat_disp

```
#! /bin/ksh

while true
do
   pstat -S
   sleep 1
done
```

B.13 Makefile

```
# ################################## Macros ###################################

CC      = cc_r            # call to C compiler
CFLAGS      =
LIBS        = -lc_r -lpthreads -lm -lXm -lXt -lX11

# ################################## Targets ##################################


LabProgs : 100bound 100boundon1 100boundon2 100unbound 3everunbound  \
4everboundon1 4everboundon2 4everboundon4 4everunbound big_copy cpubound
```

Appendix C. Special Notices

This publication is intended to help customers and systems engineers to understand the IBM RS/6000 SMP servers hardware, software design and technology in order to plan for, install and use an SMP system running IBM AIX Version 4. The information in this publication is not intended as the specification for any programming interfaces that are provided by IBM AIX Version 4.1. See the PUBLICATIONS section of the IBM Programming Announcement for IBM AIX Version 4.1.1 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AT
DRDA	ESCON
GXT150M	HACMP/6000
IBM	InfoExplorer
Language Environment	Micro Channel
Portmaster	PowerPC
PowerPC 601	PowerPC 603
PowerPC 604	RISC System/6000
RS/6000	S/370
S/390	Service Director
SP2	System/370
Ultimedia	9076 SP2

The following trademarks are trademarks of other companies.

AlphaServer	Digital Equipment Corporation
Compaq	Compaq Computer Corporation
DCE	The Open Software Foundation
DEC	Digital Equipment Corporation
Digital	Digital Equipment Corporation
Hayes	Hayes Microcomputer Products, Incorporated
HP	Hewlett-Packard Corporation
Intel	Intel Corporation
NCR	NCR Corporation
NFS	Sun Microsystems, Incorporated
OSF/1	Open Software Foundation, Incorporated
PFS	Software Publishing Company
POSIX	Institute of Electrical and Electronic Engineers
SCSI	Security Control Systems, Incorporated
Sequent	Sequent Computer Systems, Incorporated
Solaris	Sun Microsystems, Incorporated
Sportster	U.S. Robotics
Tandem	Tandem Computers, Incorporated
Teradata	AT&T Global Information Solutions
USRobotics	U.S. Robotics
VAX	Digital Equipment Corporation

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo
are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Other trademarks are trademarks of their respective companies.

Appendix D. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How To Get ITSO Redbooks” on page 357.

- *AIX Version 4 Desktop Handbook, GG24-4451*

D.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RISC System/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RISC System/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

D.3 Other Publications

These publications are also relevant as further information sources:

All AIX-related documentation is listed in the following IBM publication.

- *AIX and Related Products Documentation Overview, SC23-2456*

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *AIX Version 4.1 Quick Install Guide, SC23-2650*
- *AIX Version 4.1 Installation Guide, SC23-2550*
- *AIX Version 4.1 Files Reference, SC23-2512*
- *AIX Version 4.1 Network Installation Management Guide and Reference, SC23-2627*
- *AIX Version 4.1 iFOR/LS System Management Guide, SC23-2665*
- *AIX Version 4.1 iFOR/LS Tips and Techniques, SC23-2666*
- *AIX Version 3.2 and 4.1 Performance and Tuning Guide, SC23-2365*
- *Performance Toolbox 1.2 and 2.1 for AIX Guide and Reference, SC23-2625*
- *AIX Version 4.1 Xstation Management Guide, SC23-2713*

- *AIX Version 4.1 AIXwindows Desktop Advanced Users and System Administrators Guide, SC23-2671*
- *7012 G Series Operator Guide, SA23-2740*
- *7012 G Series Service Guide, SA23-2741*
- *7013 J Series Operator Guide, SA23-2724*
- *7013 J Series Service Guide, SA23-2725*
- *7015 Model R30 CPU Enclosure Operator Guide, SA23-2742*
- *7015 Model R30 CPU Enclosure Service Guide, SA23-2743*
- *7015 Model R00 Rack Installation and Service Guide, SA23-2744*
- *Cluster Power Control Operator and Service Guide, SA23-2766*
- *Unix Systems for Modern Architectures, Curt Schimmel, Addison Wesley, ISBN 0-201-63638-8.*
- *General Programming Concepts, SC23-2205.*
- *Programming With Threads, Kleiman, Shah, Smaalders, Prentice Hall, SR23-7473 or ISBN 0-13-172389-8*.

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269	
Canada	1-403-267-4455	
Outside North America	(+45) 48 14 2207	(long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

- **Please put me on the mailing list for updated versions of the IBM Redbook Catalog.**

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number _____
- Credit card number _____

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

List of Abbreviations

APAR Authorized Program Analysis Report

API Application Programming Interface

ASCII American Standard Code for Information Interchange

BIST Built-In Self Test

BOS Base Operating System

CCA Cache Controller Address

CCD Cache Controller for Data

CDE Common Desktop Environment

CD-ROM Compact Disk - Read Only Memory

COSE Common Open Software Environment

COP Common On-Chip Processor

CPC Cluster Power Controller

CPU Central Processing Unit

CSU Customer Setup Unit

CPI Cycles Per Instruction

DCB Data Crossbar (switch)

DCE Distributed Computing Environment

DOS Disk Operating System

EEPROM Electrically Erasable Programmable Read Only Memory

EPROM Erasable Programmable Read Only Memory

FDDI Fiber Distributed Data Interface

FRU Field Replaceable Unit

I2C Inter-Integrated Circuit

IBM International Business Machines Corporation

iFOR/LS Information For Operation Retrieval/License System

IOD Input/Output Daughter

ITSO International Technical Support Organization

IPC Inter-Process Communication

JEDEC Joint Electronic Device Engineering Council

JTAG Joint Test Action Group

LCD Liquid Crystal Display

LED Light Emitting Diode

LPP Licensed Program Product

LRU Least Recent Used

LUN Logical Unit Number

MA Memory Array

MESI Modified, Exclusive, Shared, Invalid

MIPS Millions of Instructions Per Second

MP Multiprocessor

MPB Multiprocessor Board

MPB-SysBus Multiprocessor Board System Bus

NIM Network Installation Manager

NVRAM Non Volatile Random Access Memory

NUMA Non-Uniform Memory Architectures

ODM Object Data Manager

PCI Power Control Interface

PDT Performance Diagnostic Tool

PIO Programmed Input/Output

PMR Program Modification Request

PON Power On (tests)

POSIX Portable Operating System Interface for Computer Environments

POST Power On Self Test

POWER Performance Optimized With Enhanced RISC

PLL Phase Lock Loop

PReP PowerPC Reference Platform

PSA Product Support Application

PTF Program Temporary Fix

PTX Performance Toolbox

RAM Random Access Memory

RAS Reliability Availability Serviceability

RISC Reduced Instruction Set Computer/Cycles

ROS Read-Only Storage

RPQ Request for Price Quotation

RWNITM Read With No Intent to Modify

RWITM Read With Intent to Modify

SCSI Small Computer System Interface

SIB System Interface Board

SID System Identification

SIMM Single Inline Memory Module

SMIT System Management Interface Tool

SMP Symmetrical Multiprocessor

SPOT Shared Product Object Tree

SRAM Static Random Access Memory

SSA Serial Storage Architecture

SSI Single System Image

SMC System Memory Controller

TCB Trusted Computing Base

TOD Time of Day

UP Uniprocessor

UPS Uninterruptable Power Supply

VP Virtual Processor

VPD Vital Product Data

VRMF Version Release Modification Fix

Index

M

N

O

P

R

S

T

U

V

W

Printed in U.S.A.