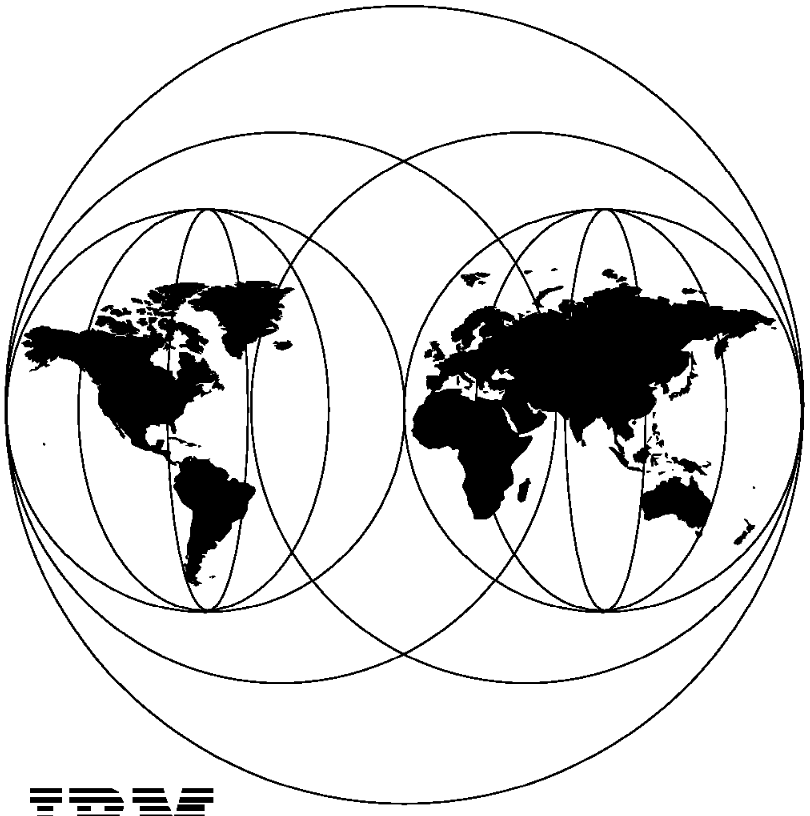
International Technical Support Organization

SG24-2583-00

IBM RISC System/6000 SMP Servers Architecture and Implementation

November 1995



IBM

International Technical Support Organization

Austin Center

International Technical Support Organization

SG24-2583-00

IBM RISC System/6000 SMP Servers Architecture and Implementation

November 1995

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Special Notices" on page xvii.

First Edition (November 1995)

This edition applies to IBM RISC System/6000 SMP servers models G30, J30 and R30 and IBM AIX Version 4.1.3.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, 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.

Abstract

This redbook describes the family of IBM RISC System/6000 Symmetric Multiprocessor (SMP) servers that run IBM AIX V4.1. It covers multiprocessing concepts, AIX V4.1 threads implementation, the SMP hardware architecture with a specific focus on the memory subsystem, and the SystemGuard service processor. It describes the SMP models G30, J30 and R30. It also highlights the Cluster Power Controller (CPC) feature, AIX V4.1 specifics to SMP systems and 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 RISC System/6000 SMP servers in terms of design and for those who plan to set up an SMP system.

A basic knowledge of AIX V4.1 is assumed.

(273 pages)

Contents

Figures

Tables

Special Notices

ADSTAR	AIX
AIXwindows	AT

C Set ++	CICS/6000
DB2/6000	DirectTalk/6000
graPHIGS	HACMP/6000
IBM	InfoExplorer
Micro Channel	POWER Architecture
PowerPC	PowerPC Architecture
PowerPC 601	PowerPC 603
RETAIN	RISC System/6000
RS/6000	SP1
SP2	Xstation Manager

The following terms are trademarks of other companies:

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.

Windows is a trademark of Microsoft Corporation.

AXP, DEC	Digital Equipment Corporation
Display Postscript	Adobe Systems, Inc.
HP	Hewlett-Packard Company
Intel	Intel Corporation
NCR	NCR Corporation
NFS	Sun Microsystems, Inc.
OpenGL	Silicon Graphics, Inc.
OSF/1, Motif	Open Software Foundation, Inc.
Sequent	Sequent Computer Systems, Inc.
Tandem	Tandem Computers Incorporated
PEX	Massachusetts Institute of Technology
POSIX	Institute of Electrical and Electronic Engineers

Other trademarks are trademarks of their respective companies.

Preface

This document is intended to assist customers and systems engineers in understanding and implementing the IBM RISC System/6000 SMP servers running AIX Version 4.1.

It contains a wide range of topics to make the reader more comfortable with the entire system, including its hardware and software.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Multiprocessing Concepts”

  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 V4.1 Threads”

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

- Chapter 3, “SMP Servers Architecture”

  This chapter describes the design rationale of the IBM RISC System/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”

  This chapter describes the three IBM RISC System/6000 SMP servers (Models G30, J30 and R30) based on the PowerPC 601 processor, and their related hardware features. It also highlights some hardware specifics to these models.

- Chapter 5, “SystemGuard”

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

- Chapter 6, “Cluster Power Controller”

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

- Chapter 7, “Installing an SMP System with AIX V4.1”

  This chapter provides an overview of AIX V4.1, some of the features that are useful in an SMP server environment. It also outlines how to install an SMP system and how to upgrade a uniprocessor to an SMP.

- Chapter 8, “SMP Performance Tools”

  This chapter shows how to use some of the performance tools to observe some characteristics of an SMP server running AIX V4.1.

- Appendix A, “SystemGuard Remote Operation Configuration”

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

- Appendix B, “Sample Programs”

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

Related Publications

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*

International Technical Support Organization Publications

- *AIX Version 4 Desktop Handbook, GG24-4451*

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

> *International Technical Support Organization Bibliography of Redbooks,* GG24-3070.

To get a catalog of ITSO technical publications (known as “redbooks”), VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Redbooks

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office. For guidance on ordering, send a PROFS note to BOOKSHOP at DKIBMVM1 or E-mail to bookshop@dk.ibm.com.

Customers may order hardcopy ITSO books individually or in customized sets, called BOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

A listing of all redbooks, sorted by category, may also be found on MKTTOOLS as ITSOCAT TXT. This package is updated monthly.

ITSO Redbooks on the World Wide Web (WWW)

Internet users may find information about redbooks on the ITSO World Wide Web home page. To access the ITSO Web pages, point your Web browser to the following URL:

```
http://www.redbooks/ibm.com/redbooks
```

IBM employees may access LIST3820s of redbooks as well. The internal Redbook home page may be found at the following URL:

```
http://w3.itsc.pok.ibm.com/redbooks/redbooks.html
```

Acknowledgments

This project was designed and managed by:

Bob Minns
International Technical Support Organization, Austin Center

Yves Bex
International Technical Support Organization, Austin Center

The authors of this document are:

Rob Hendry
IBM South Africa

Satish Sharma
IBM New Zealand

Colin Fearnley
IBM South Africa

Yves Bex
International Technical Support Organization, Austin Center

This publication is the result of a residency conducted at the International Technical Support Organization, Austin Center.

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

Jim Nicholson
IBM Austin

Mark Brown
IBM Austin

Dave Rittinger
IBM Austin

Kanti Shah
IBM Austin

Vijay Oak
IBM Austin

Marcus Brewer, Editor
ITSO, Austin Center

Chapter 1. Multiprocessing Concepts

Multiprocessing is a relatively new concept for the IBM RISC System/6000 environment although it is not at all new for IBM. There are a lot of terms, abbreviations and concepts that will be new to many readers. This chapter is an attempt to introduce some of them.

1.1.1 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 speeds 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 a number of factors to be taken into consideration, such as resource sharing.

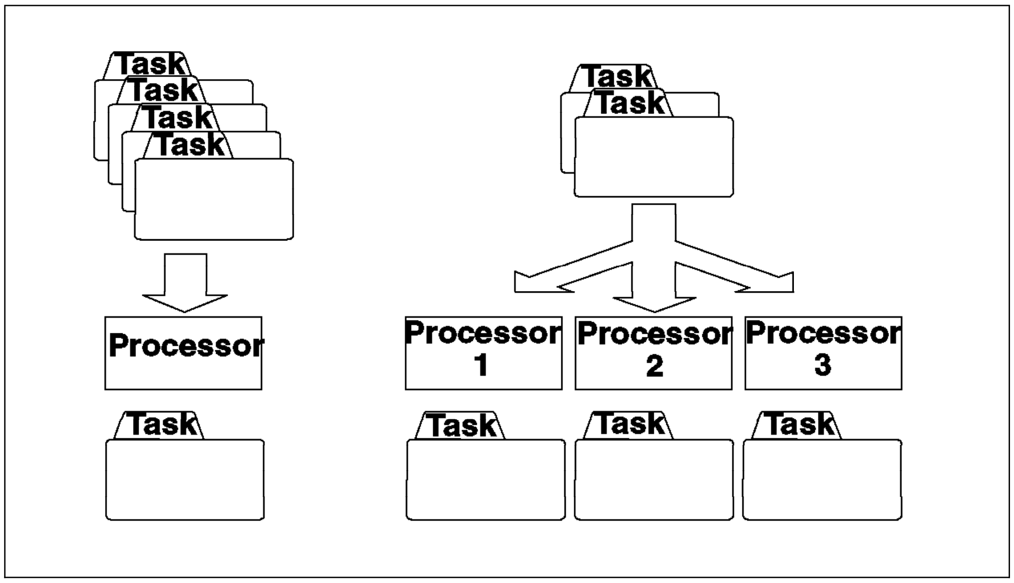


Figure 1. Uniprocessing versus Multiprocessing

1.1.2 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?

1.1.3 Sharing Resources

When resources are required to be shared, there are a number of aspects that need to be taken into account when designing SMP systems. First, the bandwidth between the processors and resources, such as memory and I/O subsystem, must be as high as possible. The inherent latency (built in delay) 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 limit scalability of the system.

1.1.4 Multiprocessor Types

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

1.1.4.1 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.

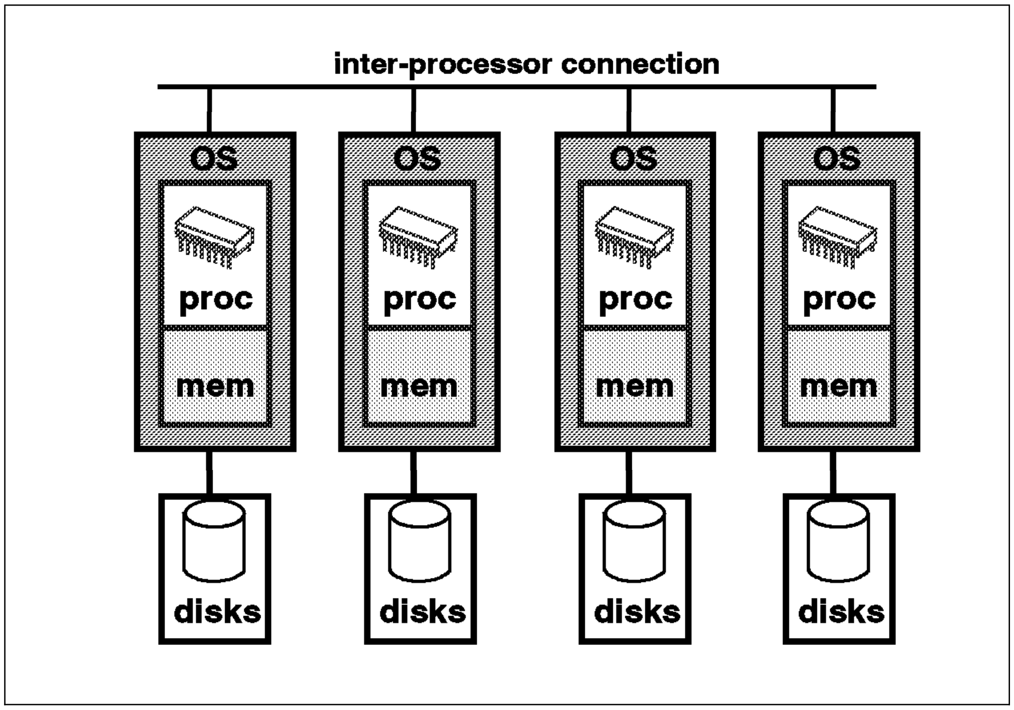


Figure 2. Shared-Nothing Multiprocessor

Examples of a shared-nothing MP are in the IBM SP1 and SP2 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; so scalability can be very high (typically several hundreds of processors).
- 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.

1.1.4.2 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.

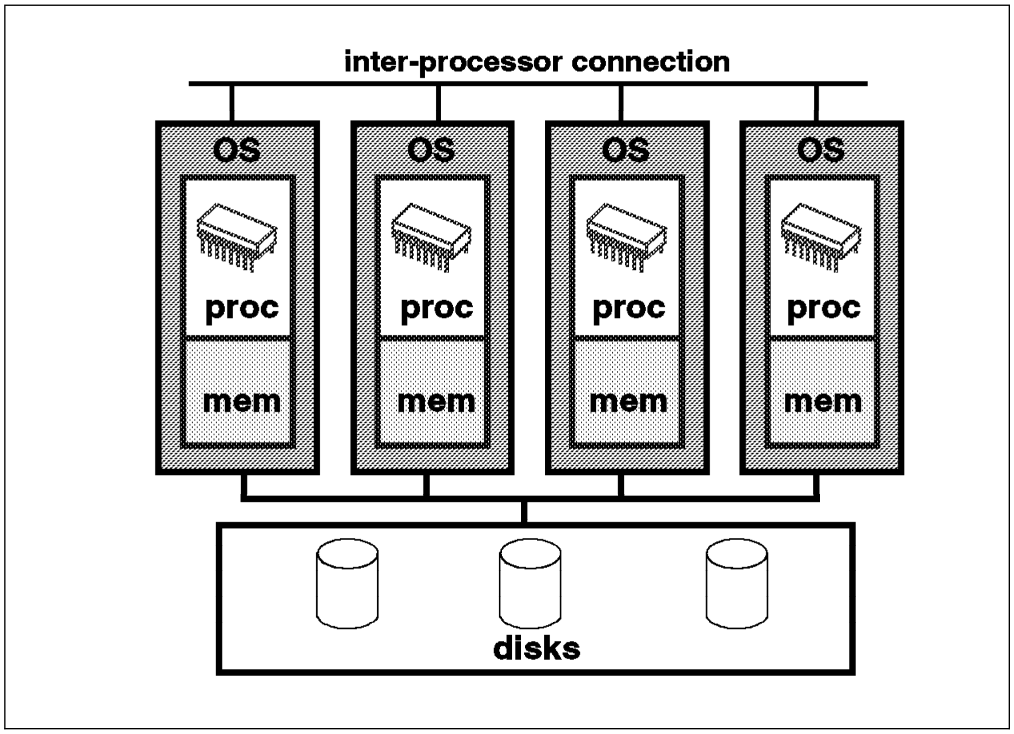


Figure 3. Shared-Disks Multiprocessor

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-nothing 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.

1.1.4.3 Shared-Memory Cluster

Here are some of the characteristics of a shared-memory cluster:

- Each processor has its own caches, memory and I/O subsystem.
- Each processor runs a copy of the operating system.
- The processors are tightly coupled through a piece of shared memory.
- Communication between processors is done via this shared memory.

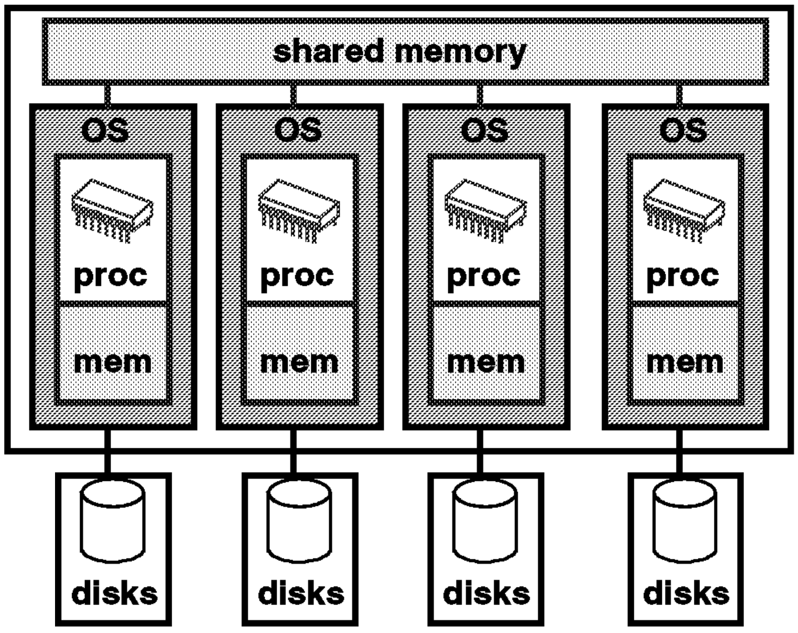


Figure 4. Shared Memory Cluster

This not a common MP implementation. IBM developed such an architecture and built a system called the Power4.

1.1.4.4 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.

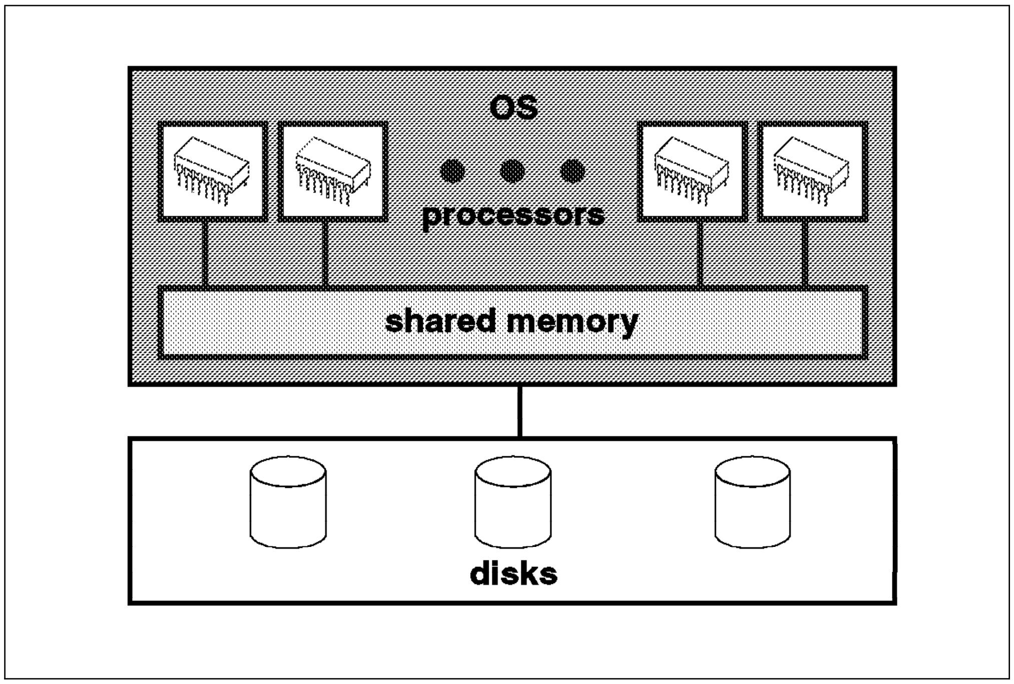


Figure 5. Shared-Memory Multiprocessor

Shared-memory MP is one of the most common multiprocessing implementations. Following are some examples of this implementation:

- IBM G30, J30, R30
- HP T500, x70 range, K series
- SUN SPARCserver 20 and 1000, SPARCcenter 2000
- DEC 2100 A500 and A600, DEC 7000 AXP
- NCR 3455 and 3525
- Sequent Systems
- SGI Power Challenge

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.
- Programming model is familiar because UP programming model can still be used.

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

- Scalability is limited due to the sharing of resources.
- In order to take advantage of multiprocessing, the use of a thread library is required. This programming model requires specific skills.

1.1.5 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. Slave processors can only perform computation. On a slave processor, all I/O operations 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 runs I/O operations. Also, failure of the master processor is catastrophic; the system cannot continue to run.

In a symmetric shared-memory multiprocessor (SMP), all of the processors are functionally equivalent; they all can perform 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.

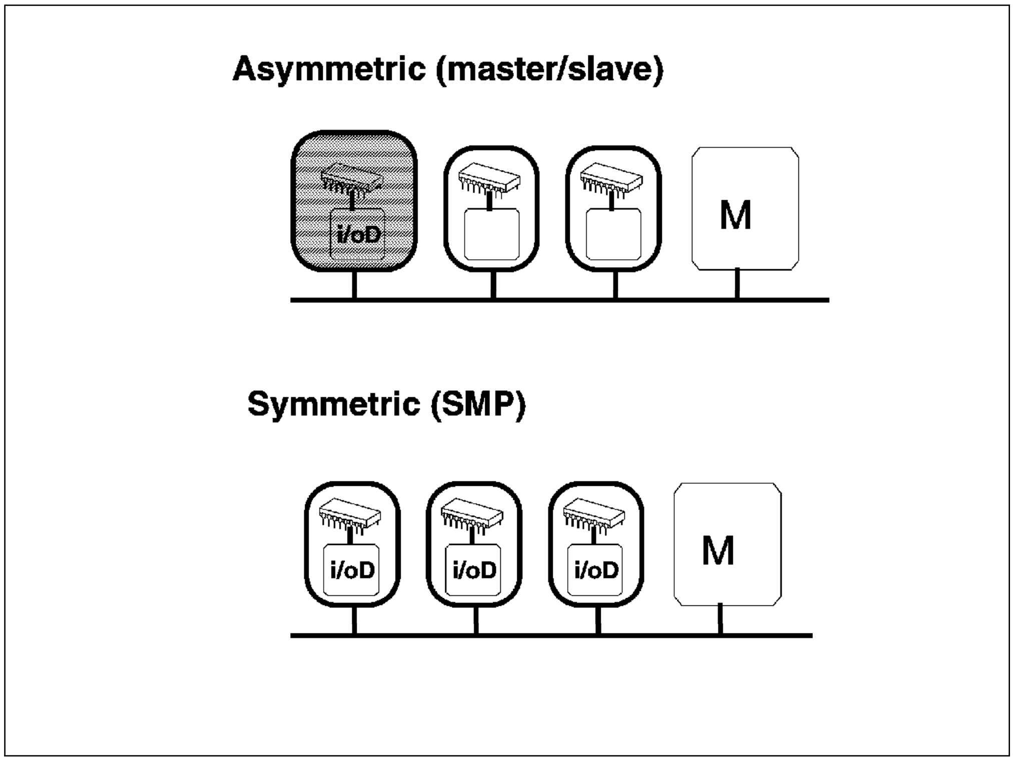


Figure 6. Symmetric versus Asymmetric Shared-Memory Multiprocessors

1.2 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 issues.

1.2.1.1 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. On the other hand, 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 and around 1 to 4 MB.

The main memory is the third level of memory. Its access time is slow in comparison to L1 but much faster than disks. The number of cycles needed to load data from the memory is usually around 20 to 50 cycles, 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 is 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.

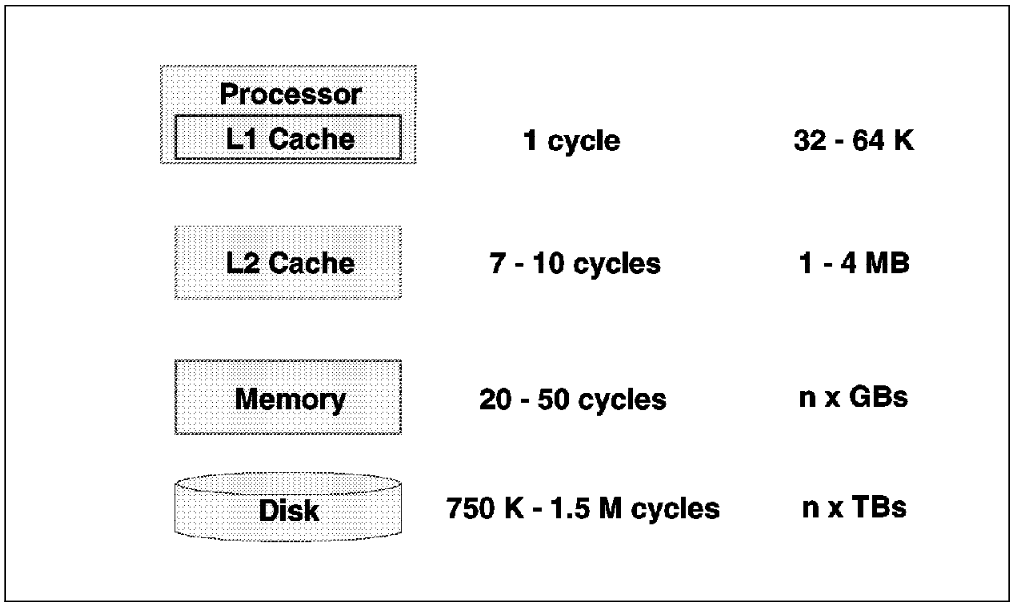


Figure 7. Memory Hierarchy

1.2.1.2 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.

1.2.1.3 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 processor's 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.

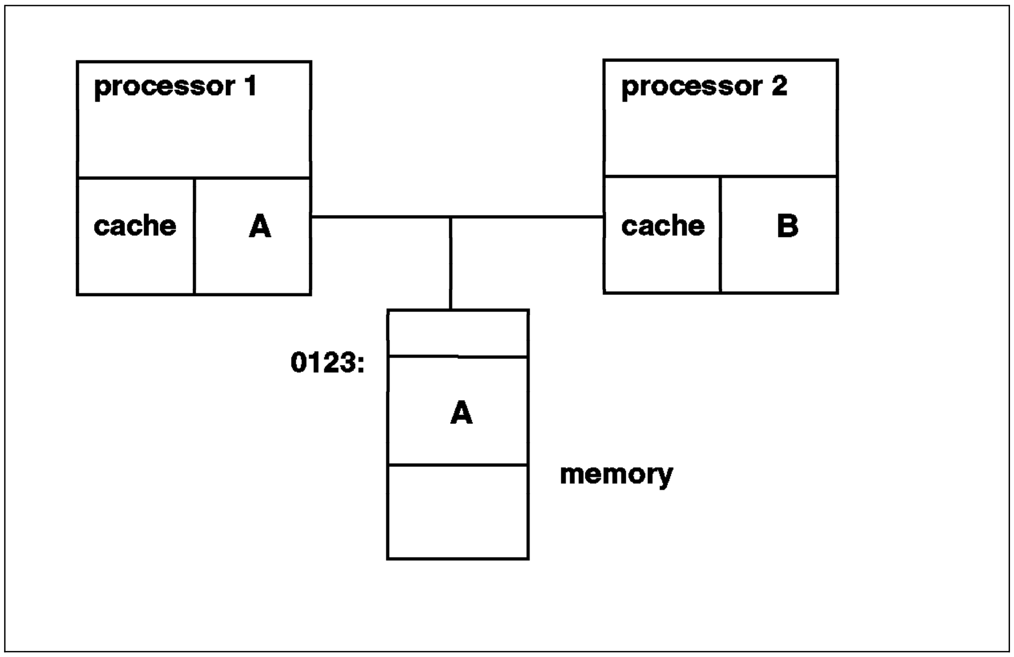


Figure 8. SMP Cache Coherency Problem

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.

1.2.1.4 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 broadcasted which causes that entry to be invalidated in all other caches. This is called 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.

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

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

1.2.1.5 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 put into effect 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.

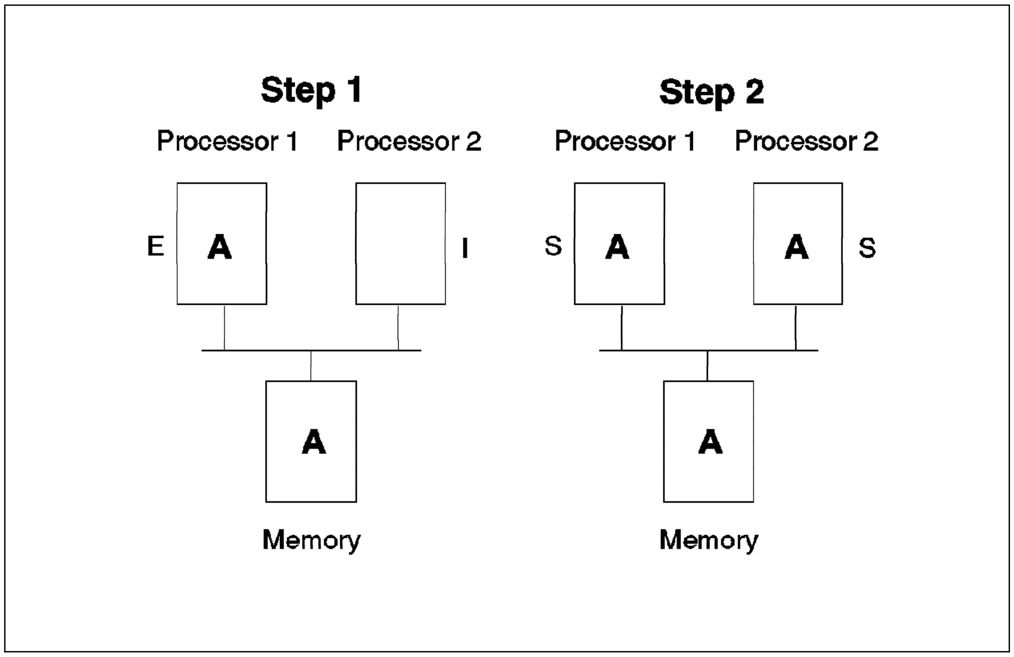


Figure 9. MESI Protocol - Steps 1 and 2

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.

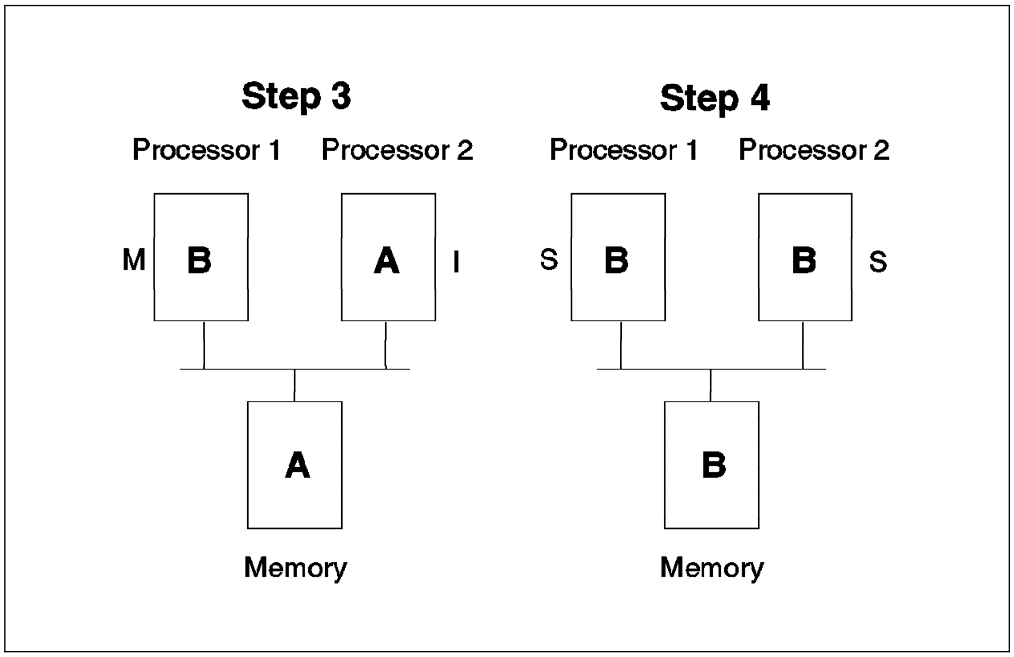


Figure 10. MESI Protocol - Steps 3 and 4

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 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.

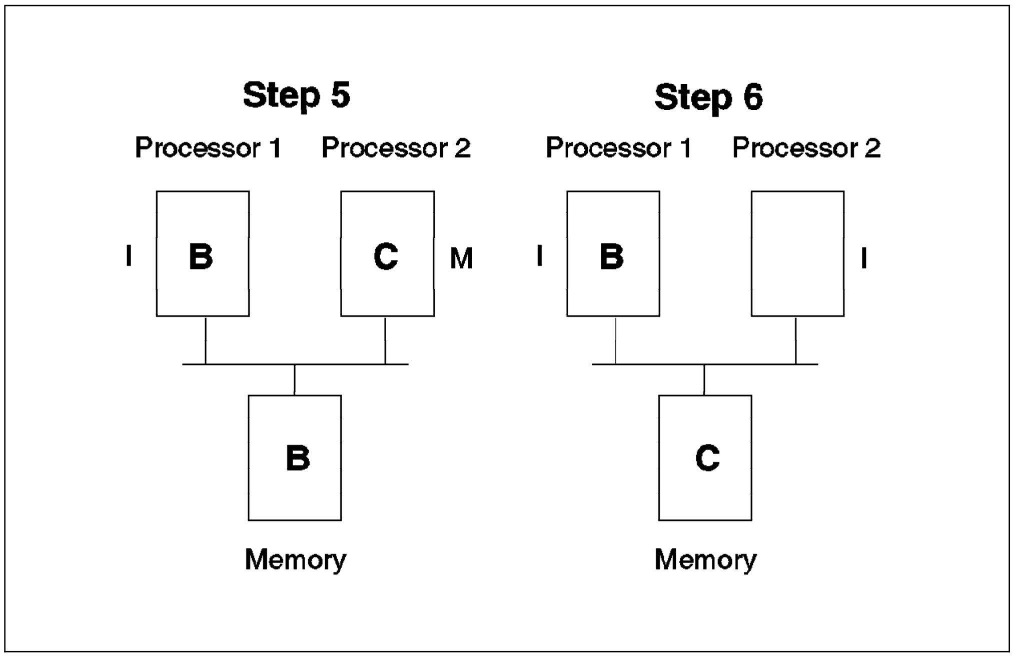


Figure 11. MESI Protocol - Steps 5 and 6

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 Recent 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.

1.2.1.6 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 Recent Used (LRU) algorithm is used, and the LRU sector is written back to the memory. The memory is also updated in case of a cache flush.

1.2.1.7 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).

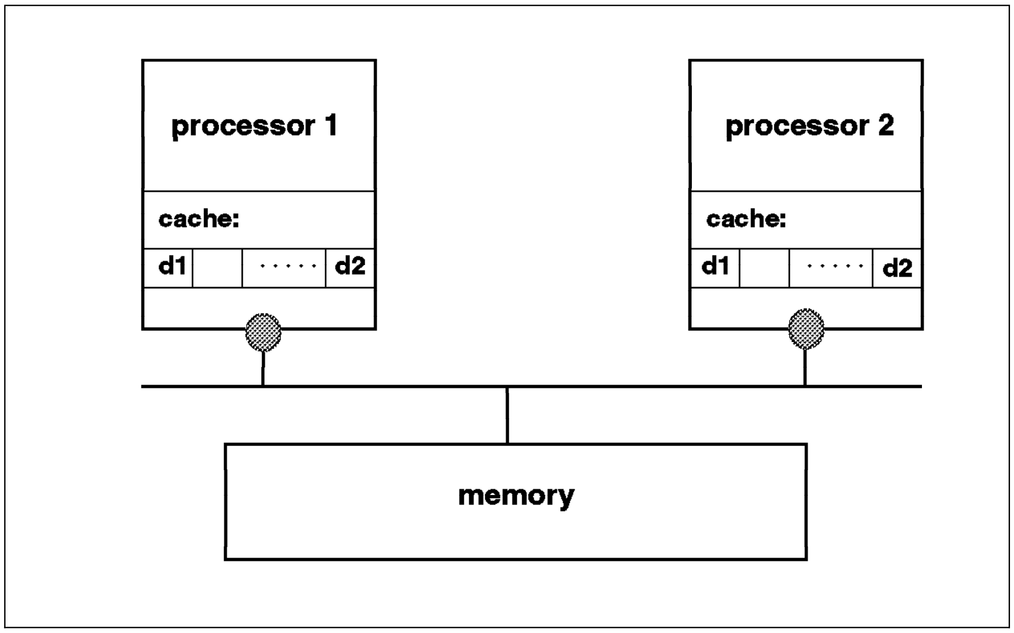


Figure 12. False Sharing

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 valid 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.

1.3 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.

1.3.1 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.

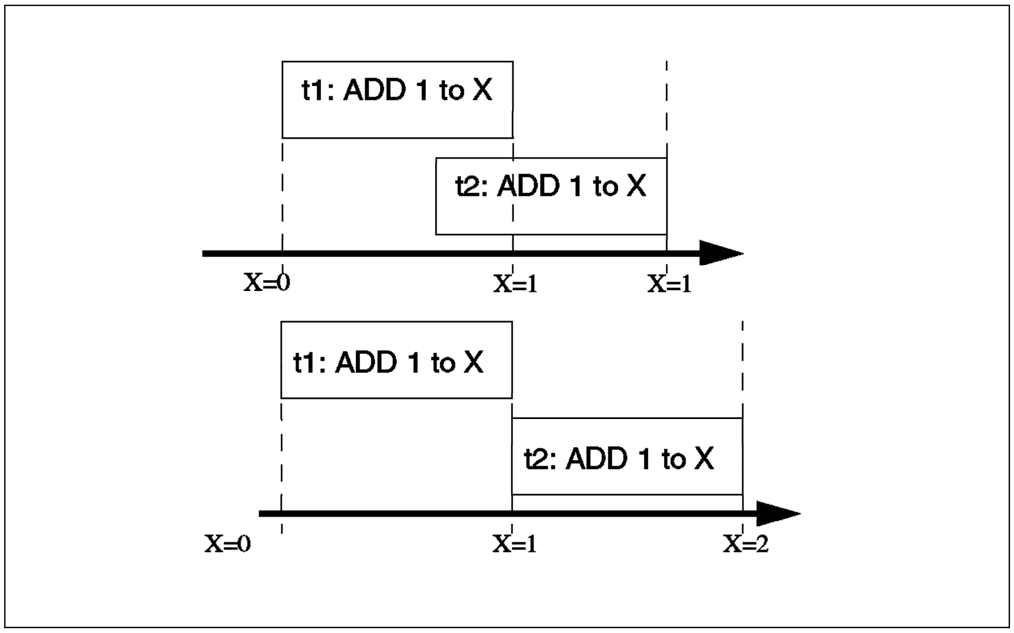


Figure 13. Synchronization Issue

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.

1.3.2 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, and 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.

1.3.3 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, 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.

1.3.4 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.

1.3.5 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, preemptable
- 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.

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.

1.3.6 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.

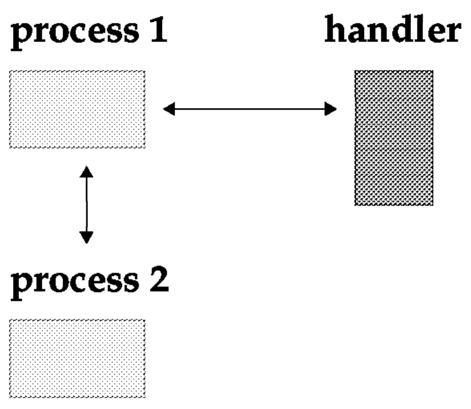


Figure 14. UP Synchronization

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.

1.3.7 SMP Synchronization

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

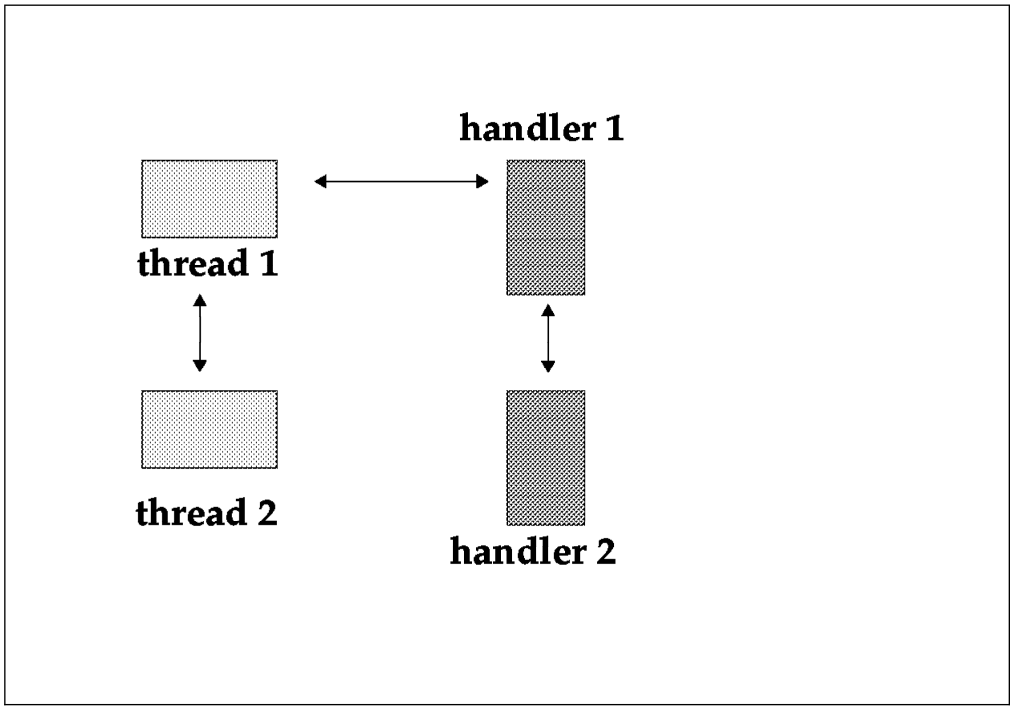


Figure 15. SMP Synchronization

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 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.
- 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.

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.

1.3.8 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, do not block and return 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

1.3.9 AIX V4.1 Lock Services Summary

In the previous sections, we covered the locking mechanism that is 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 V4.1 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.

1.3.10 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 ten 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.

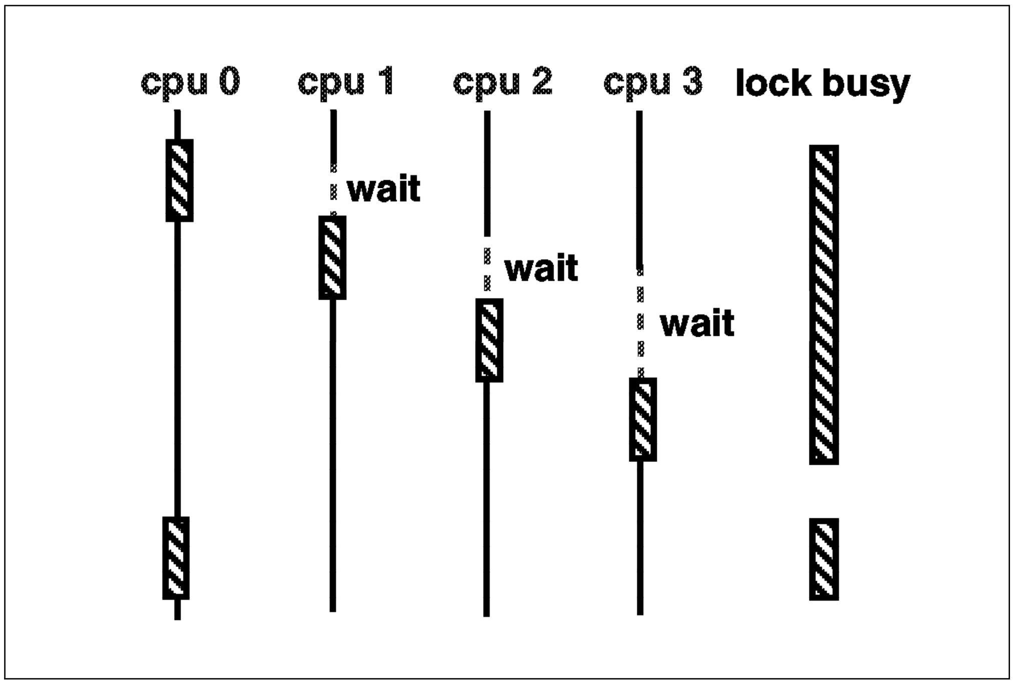


Figure 16. Lock Penalty

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.

1.3.11 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 on page 21 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.

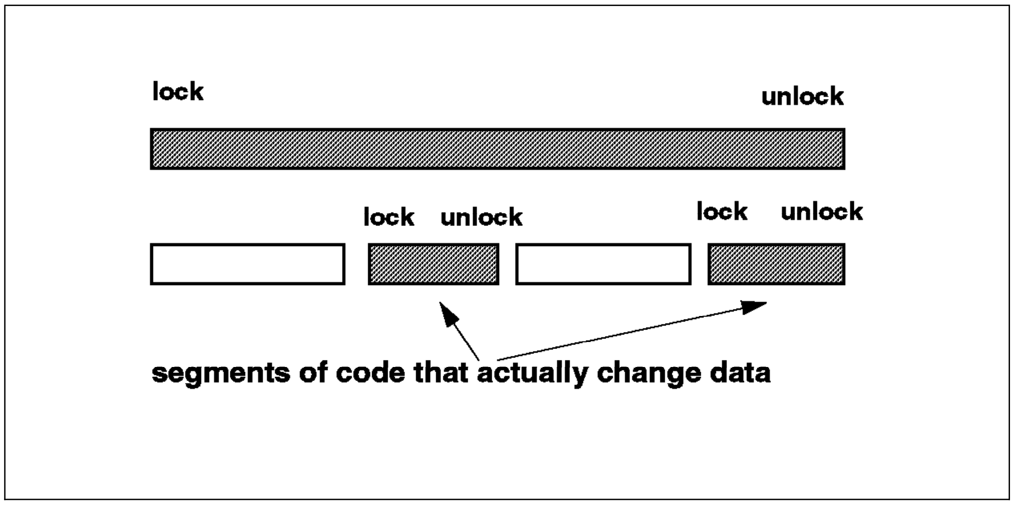


Figure 17. Lock Granularity

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.

1.3.12 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.

1.3.13 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.

The way this is accomplished is as follows: As shown in Figure 18, 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.

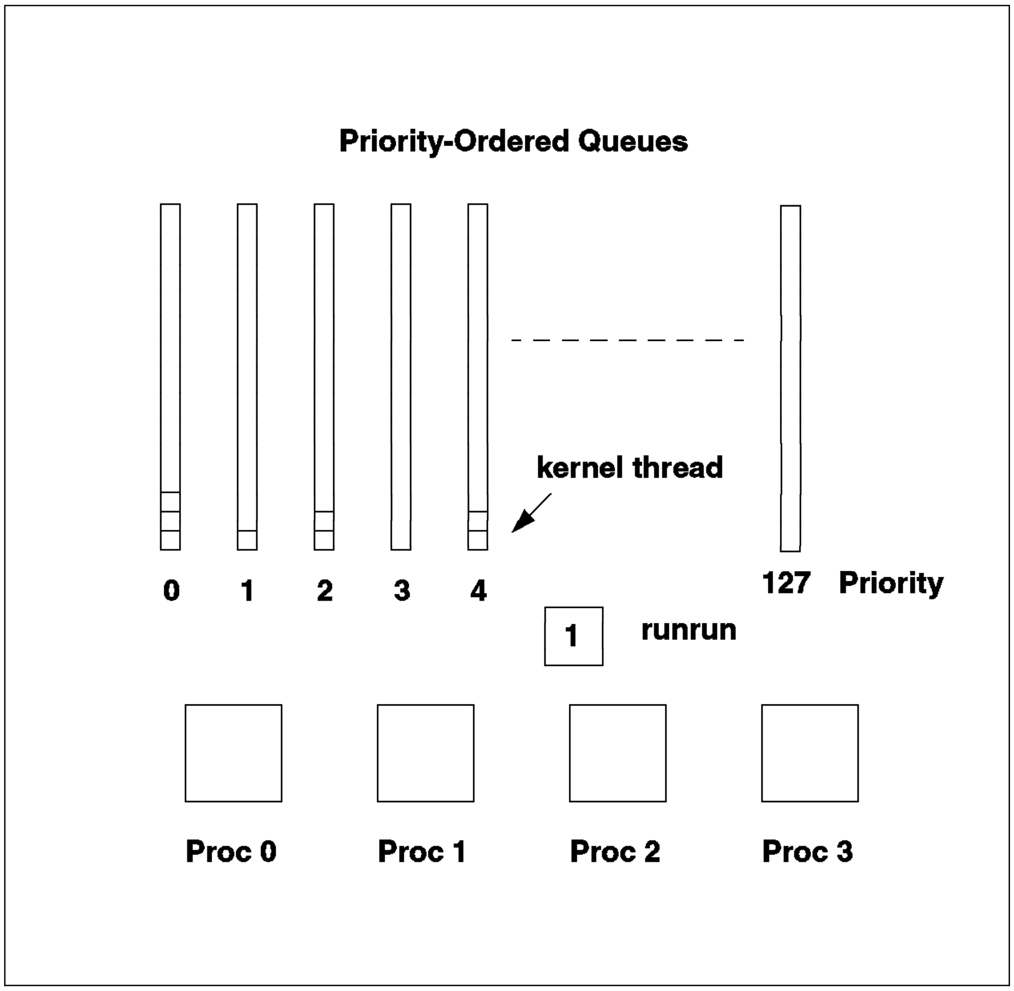


Figure 18. Threads Dispatching

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 if 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.

1.3.14 Binding

Binding is the strongest form of processor affinity, and 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.

1.3.15 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 unavailable; unavailable is indicated when a hardware failure is detected by the system.

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

Table 1. Physical and Logical Processor Numbering

ODM Name	Physical Number	Logical Number	Processor State
proc0	0	0	Enabled
proc1	2		Disabled
proc2	2		Unavailable
proc3	3	1	Enabled

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, and it can also enable or disable processors (a reboot is required for the change to take effect).

1.3.16 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 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, a porting is necessary. That 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.

1.3.17 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.

1.3.17.1 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.

1.3.17.2 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 funnelled.

Therefore, all device drivers that do not tell the kernel that they are MP-safe are funnelled. 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.

1.3.18 PowerPC Specifics
The PowerPC processor defines a *weakly ordered memory* model which allows an 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.

1.4 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.

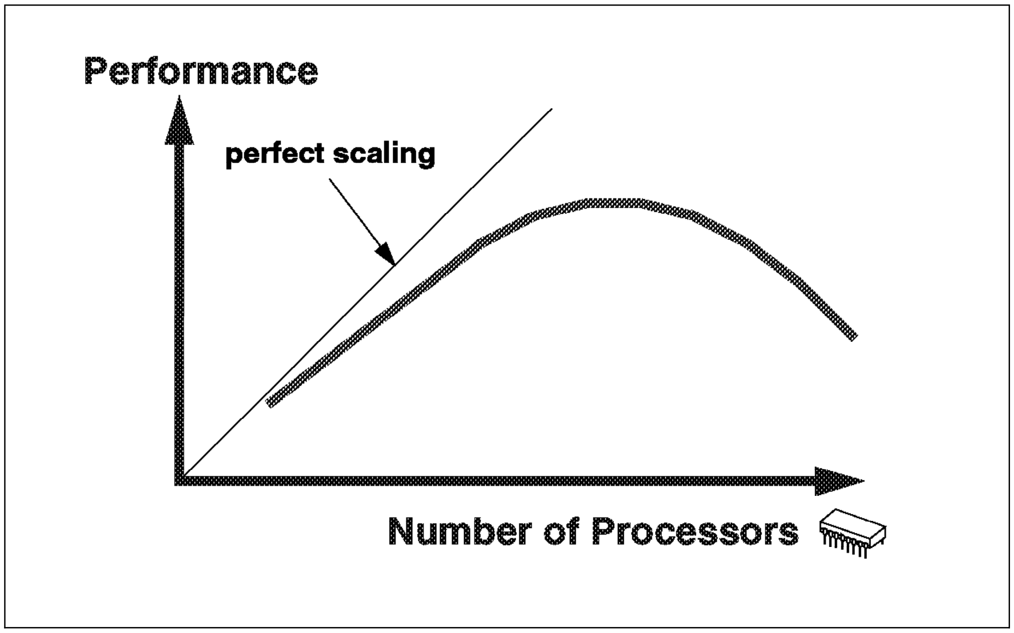


Figure 19. Scaling

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

- Bus/Switch contention increases when the number of processors increases.
- Memory contention increases 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 scale poorly on an SMP.

For example, the SPECrate workload scales very well on an SMP because it is made of several independent programs that do not interact 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.

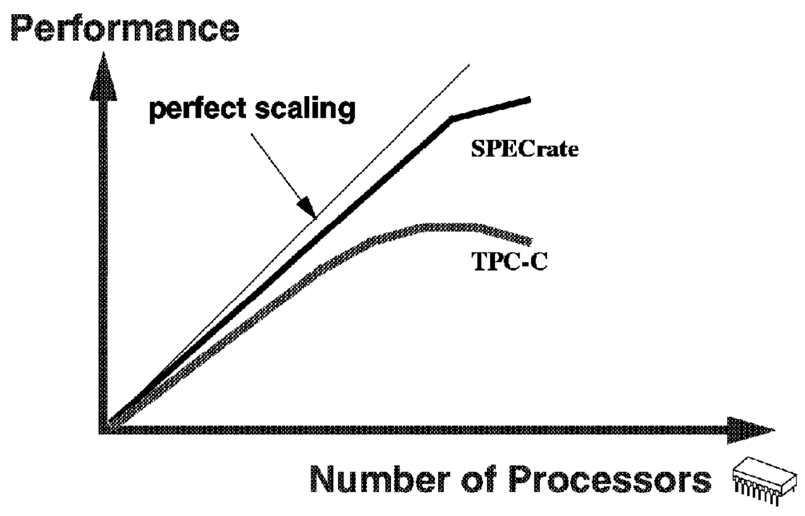


Figure 20. Scaling is Workload Dependent

1.4.1 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 an excellent result.

Table 2. Hypothetical SMP Scaling Metrics

Number of Processors	**Value of the Performance Test (Hypothetical)**	**Ratio to 1 Processor**	**Ratio to Number of Processors**
1	100		
2	180	1.8	0.90
4	300	3.0	0.75
6	410	4.1	0.68
8	480	4.8	0.60

1.4.2 Two-Dimensional Scaling

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

The IBM RISC System/6000 SMPs 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 chip 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.

1.5 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.

1.5.1 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.

1.5.2 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.

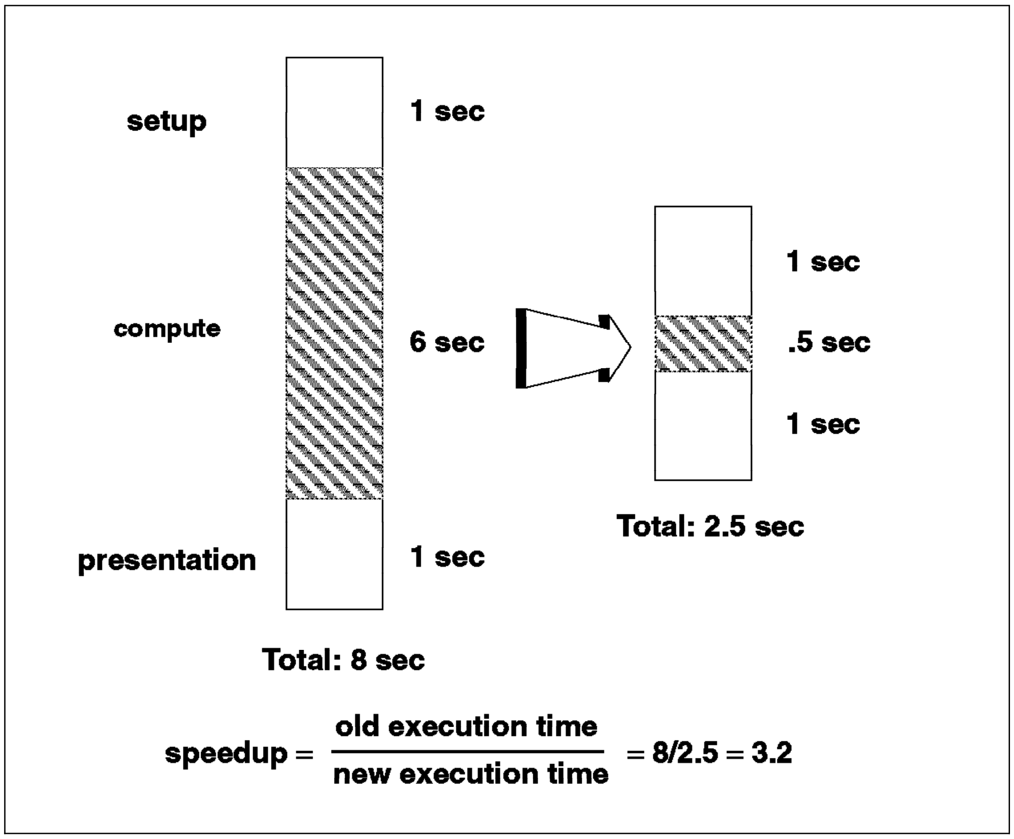


Figure 21. Amdahl's Law

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.

1.5.3 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 processors speed is key. Code is often made with short loops of instructions and may fit in the L1 cache.

1.6 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. POWER2 or 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 V4.1 Threads

This chapter discusses the AIX V4.1 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 controls 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 4.1, 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 to 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 on page 33 illustrates the differences between a traditional UNIX process and a multithreaded process.

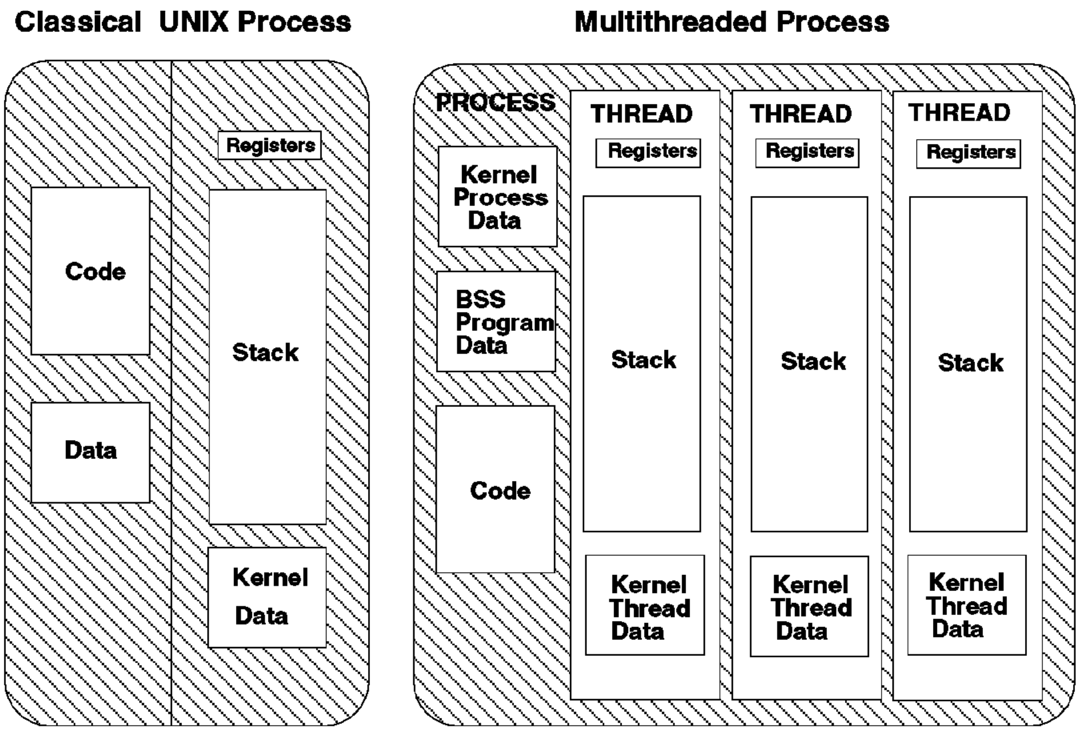


Figure 22. Multithreaded Process

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 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 on page 35 shows the structure of threads and processes.

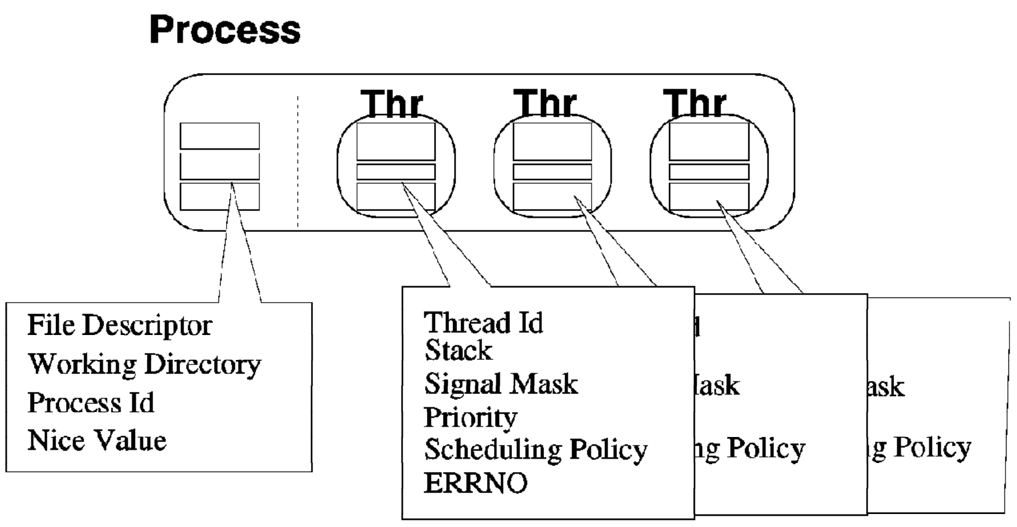


Figure 23. Threads and Processes Properties

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 V4.1, 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 controls 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 V4.1 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 a *M:1* implementation on AIX V3.2. Figure 24 is an illustration of this model.

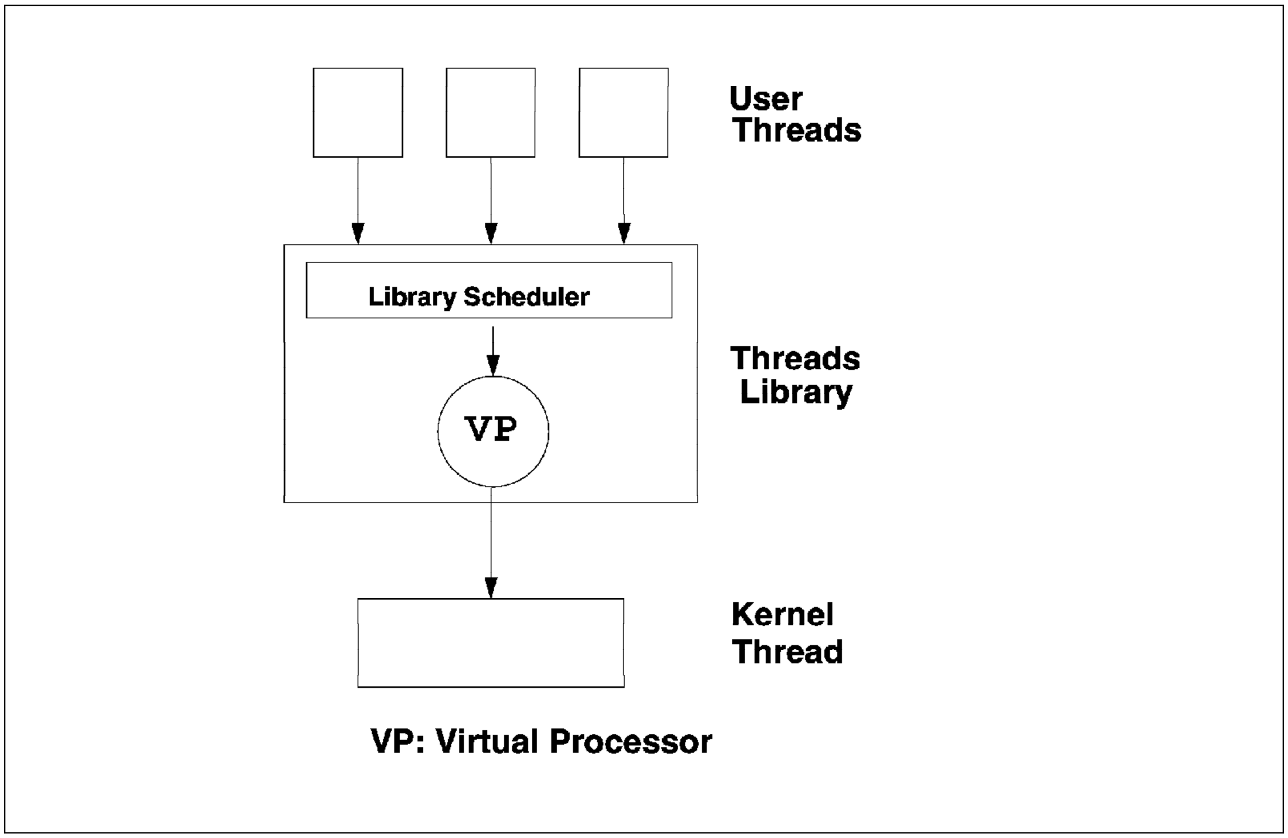


Figure 24. M:1 Model

2.4.1.2 1:1 Model

The *1:1* model is the one which is currently implemented by AIX V4.1 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 illustrates the *1:1* model.

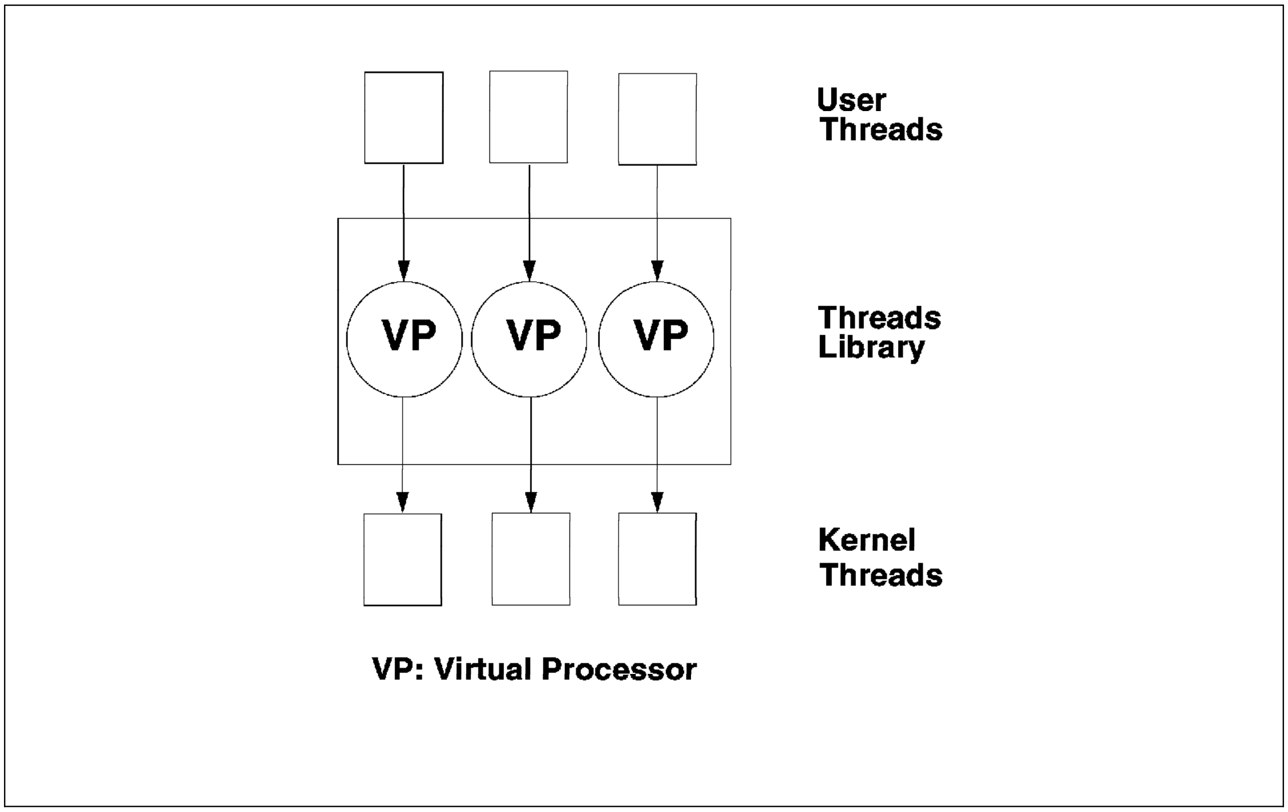


Figure 25. 1:1 Model

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 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.

When a thread needs a VP to execute a system call, it remains bound to this VP until the system call is done, even if the thread has to wait for a resource inside the kernel.

Figure 26 shows the *M:N* model implementation.

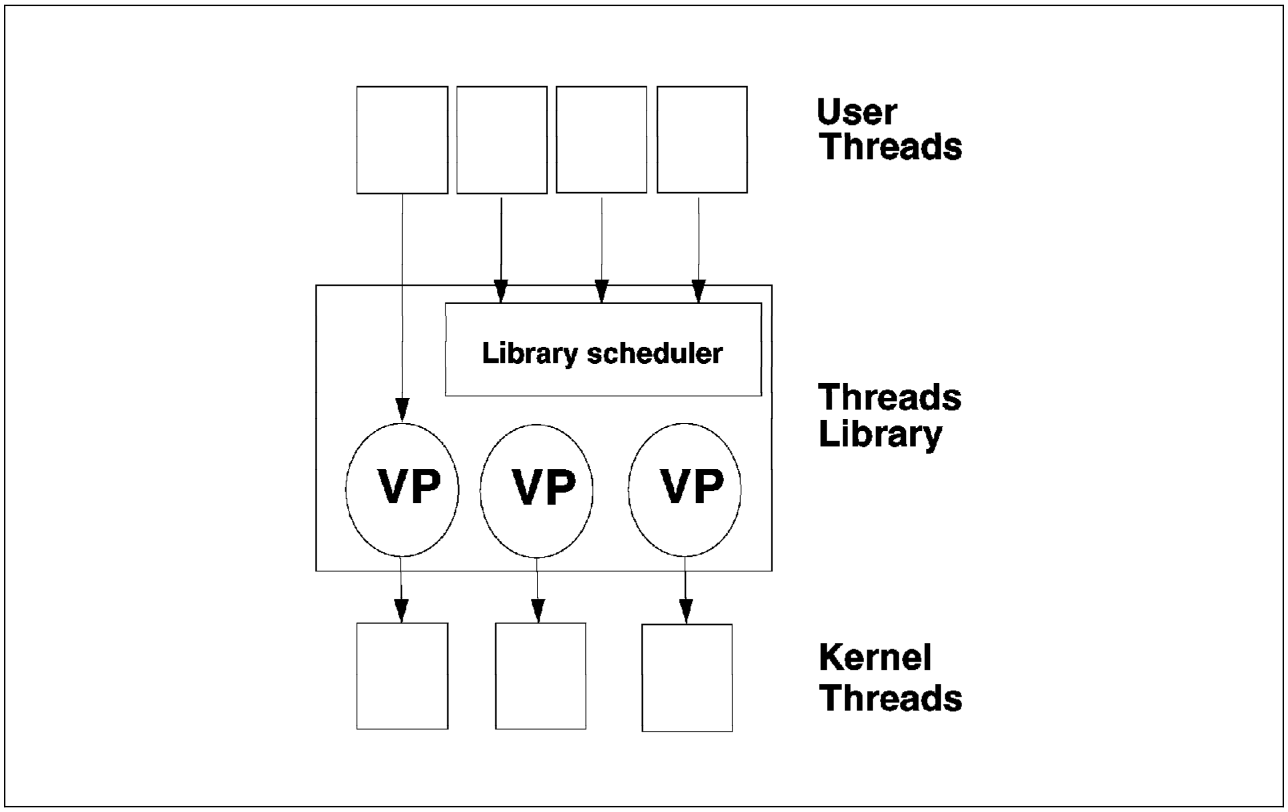


Figure 26. M:N Model

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 a *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 V4.1 Kernel Support of Threads

AIX V4.1 has a multithreaded kernel. It supports a large number of threads throughout the system (up to 256 K), 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 V4.1 Threads Library Implementation

The AIX V4.1 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 intent to implement an *M:N* model later, and also to move to the official version of the POSIX standard when it becomes 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 on page 41 shows the AIX V4 Threads Implementation.

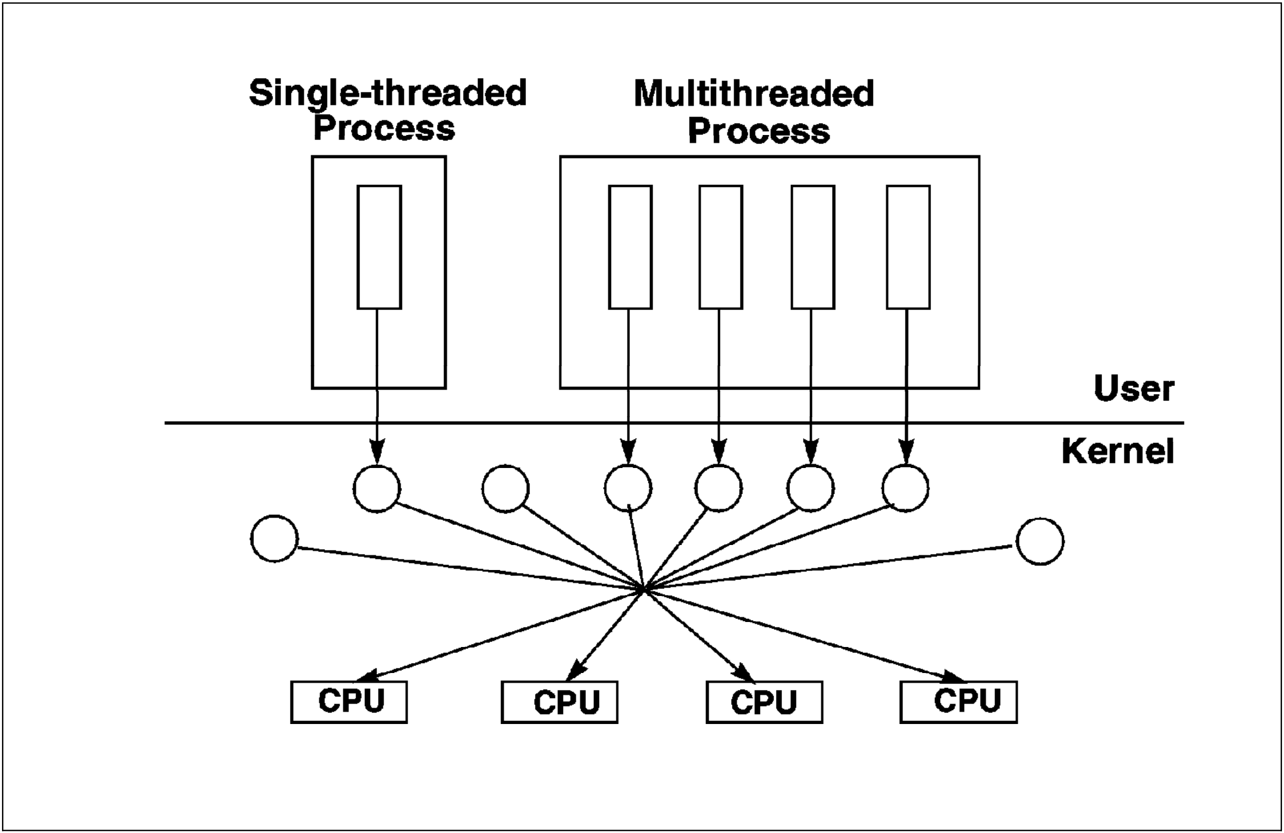


Figure 27. AIX V4.1 Thread Architecture

2.8 Threads Scheduling

In AIX Version 3.2, the scheduler dispatches processes. In AIX Version 4.1, the scheduler dispatches threads (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.

2.9 Threads Programming Considerations

Facilities are provided to 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.

2.9.1 Thread-Safe Libraries

In single-threaded processes there is only one flow of control. Therefore the code executed by these processes do 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.

2.9.2 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.

2.9.3 Thread Attributes

When creating a thread, you can pass some attributes to the threads. 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`).

2.9.4 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 (hold 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 broadcasted.

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 tries 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.

2.9.5 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.

2.9.6 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 a 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 4.1, 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` by different threads of the same process are serialized at the kernel level.

2.9.7 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.

2.9.8 Signal Management

Signals in a multithreaded environment is 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 V4.1.

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.

2.9.9 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 245.

2.9.10 Debugger Threads Support

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

2.9.11 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 programs samples are provided in Appendix B, “Sample Programs” on page 245.

```
# 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

2.9.12 AIX V4.1 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 family of products 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. But 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. That is the way processors communicate in an SMP.

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), bigger than L1 but a bit slower
- The memory, slower than L2 but which can have a high 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 750 K to 1.5 M cycles to access data.

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!

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. So, 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 L1, 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.

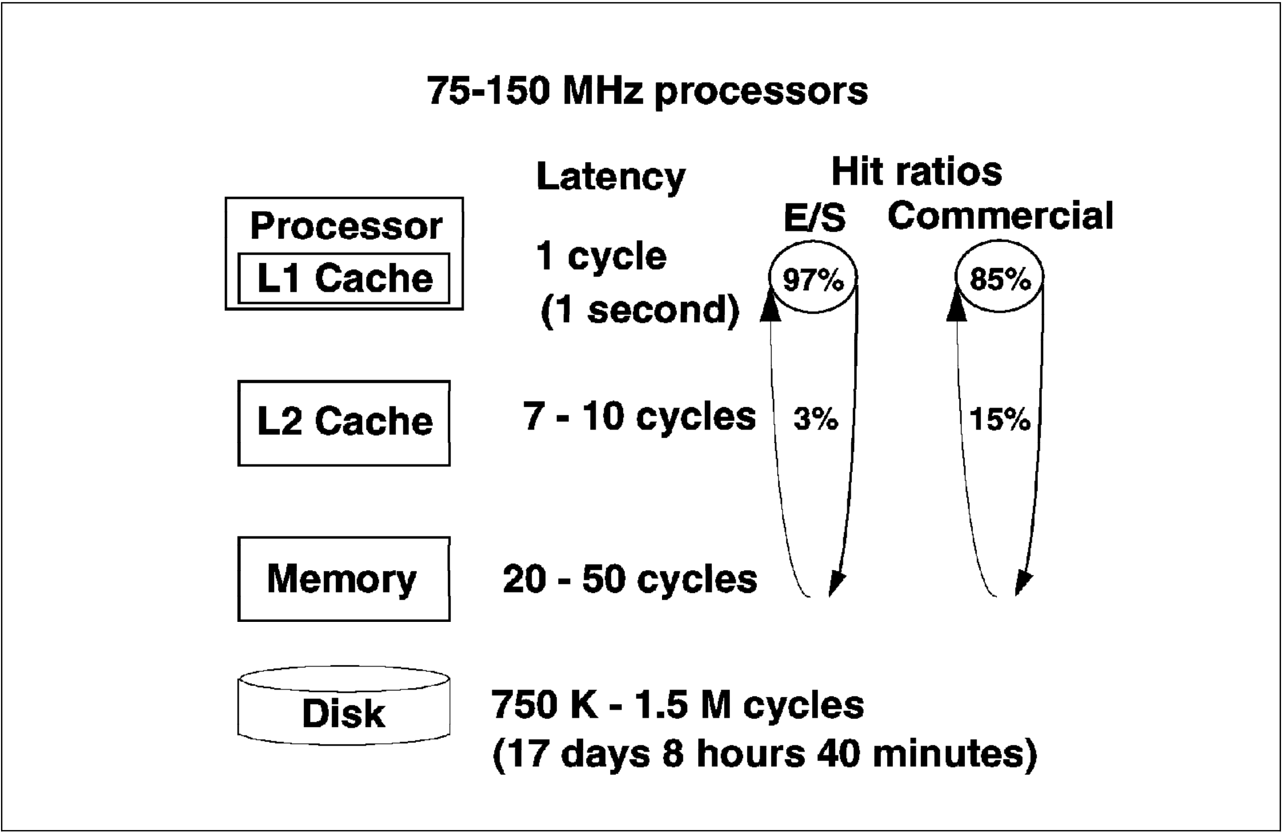


Figure 29. Scientific vs. Commercial Environment

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.

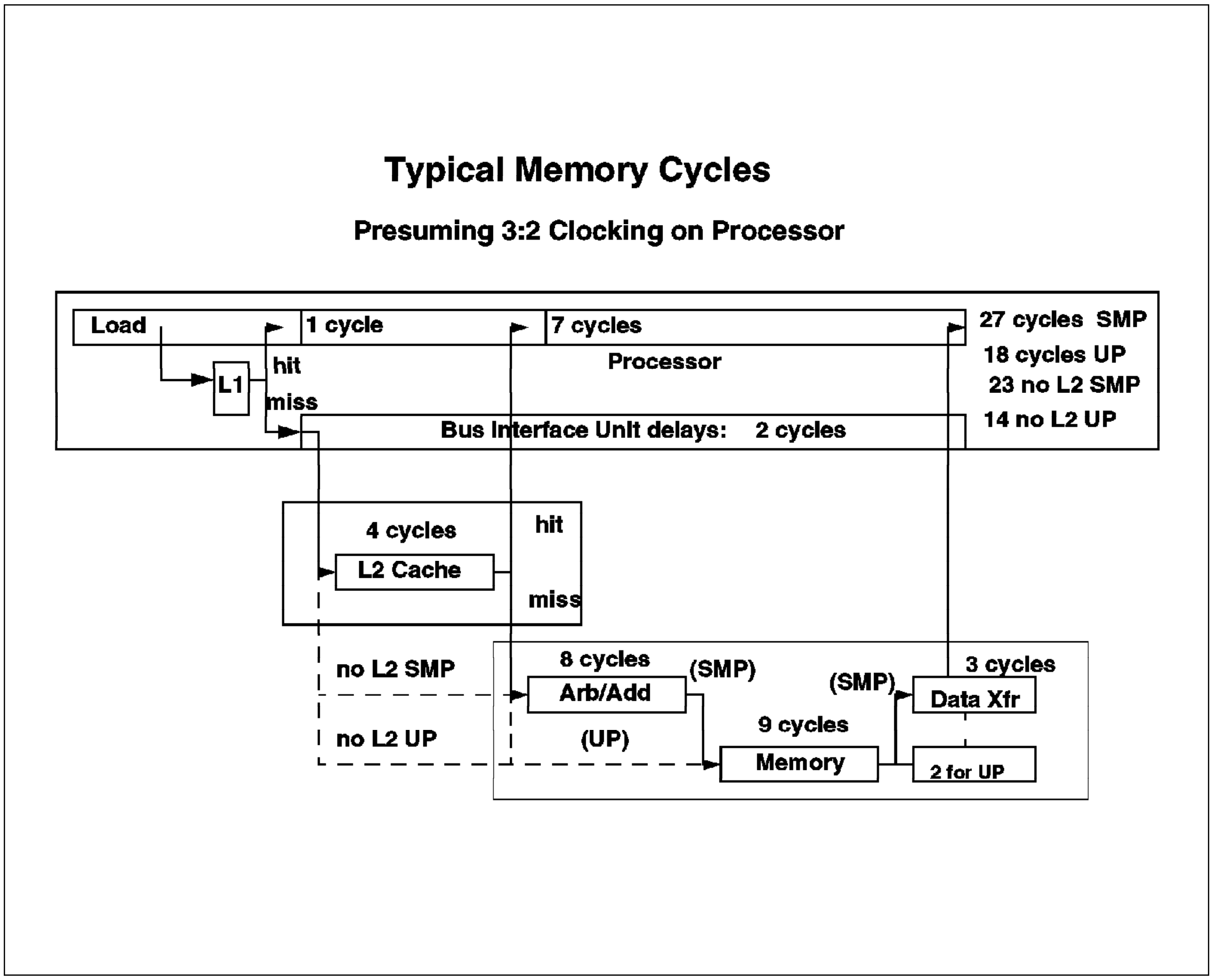


Figure 30. Typical Memory Cycles

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 (PPL) 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.

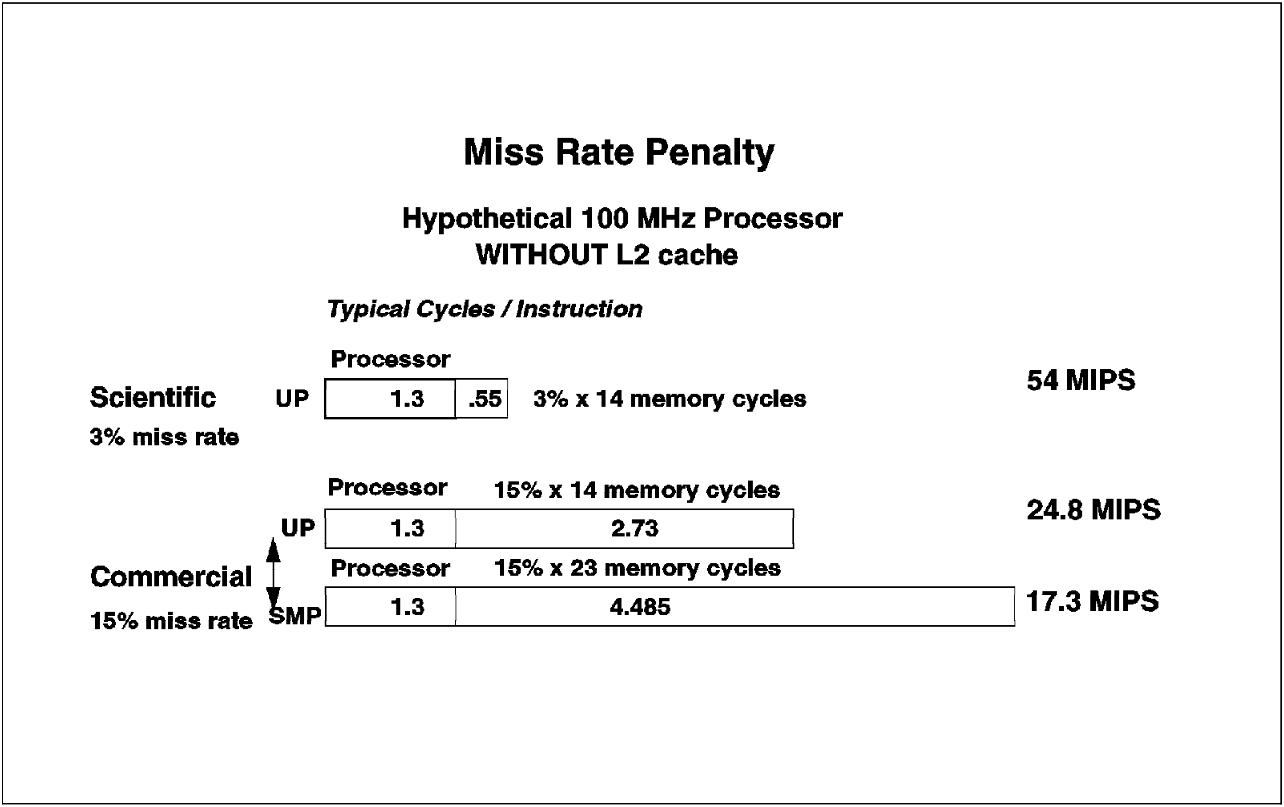


Figure 31. Miss-Rate Penalty

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.

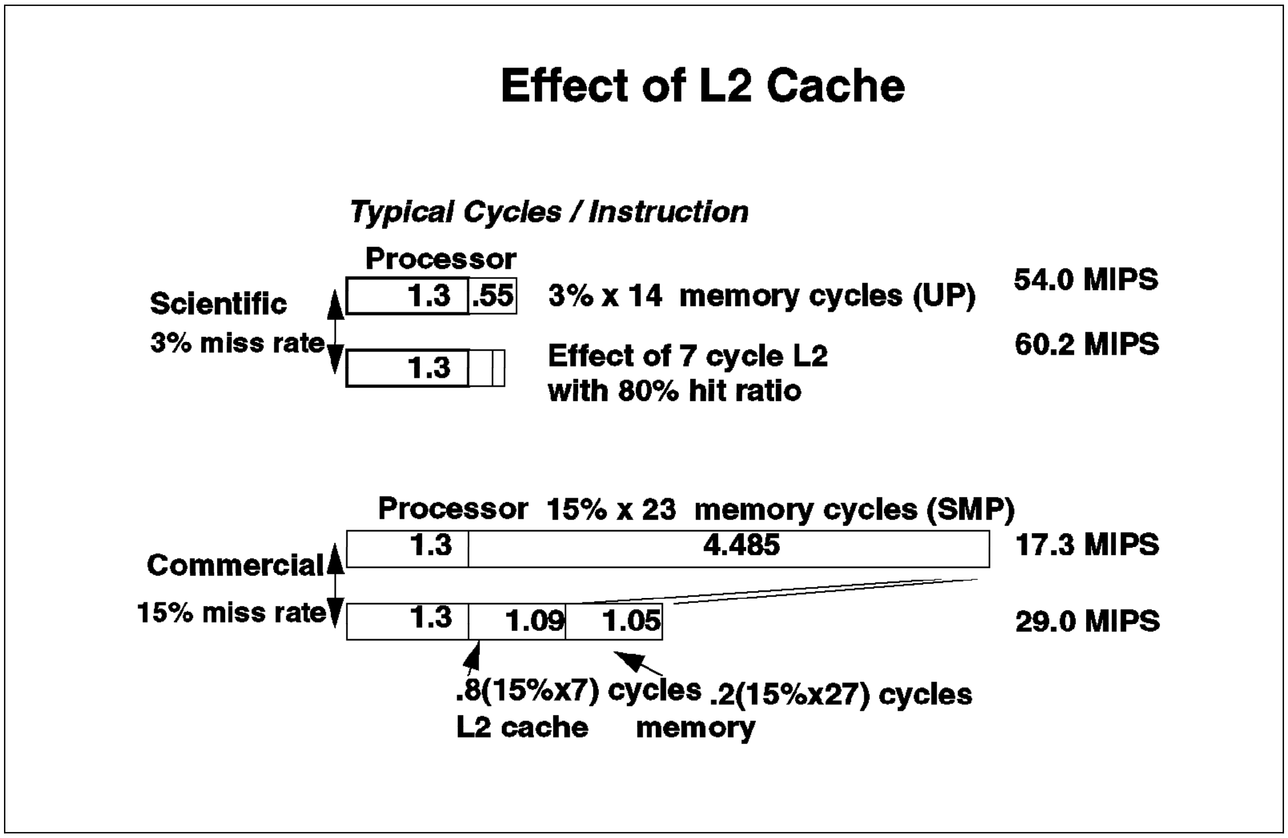


Figure 32. Effect of L2 Cache

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.

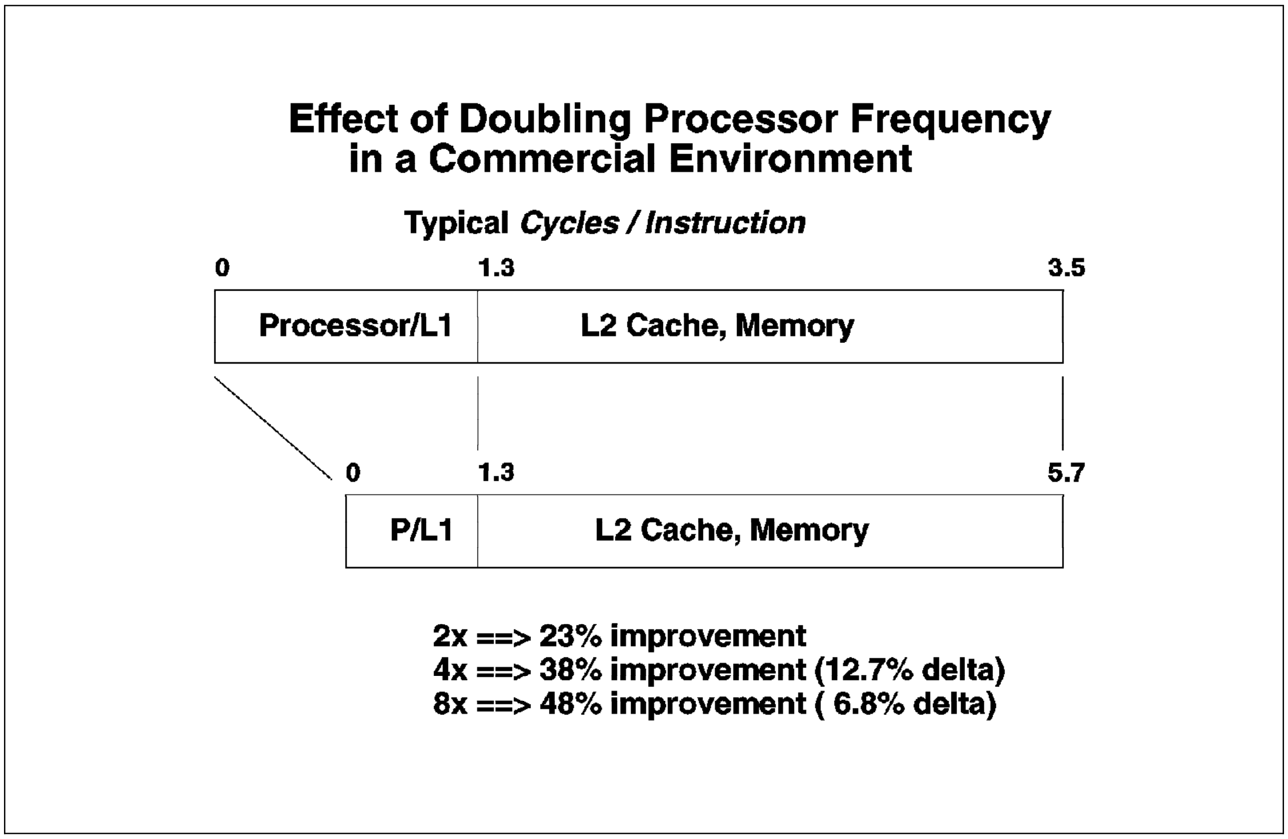

Effect of Doubling Processor Frequency in a Commercial Environment
Typical Cycles / Instruction
0
1.3
3.5
Processor/L1
L2 Cache, Memory
0
1.3
5.7
P/L1
L2 Cache, Memory
2x ==> 23% improvement
4x ==> 38% improvement (12.7% delta)
8x ==> 48% improvement (6.8% delta)

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 systems.

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 amount 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 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 cache. 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 increases as well. 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.

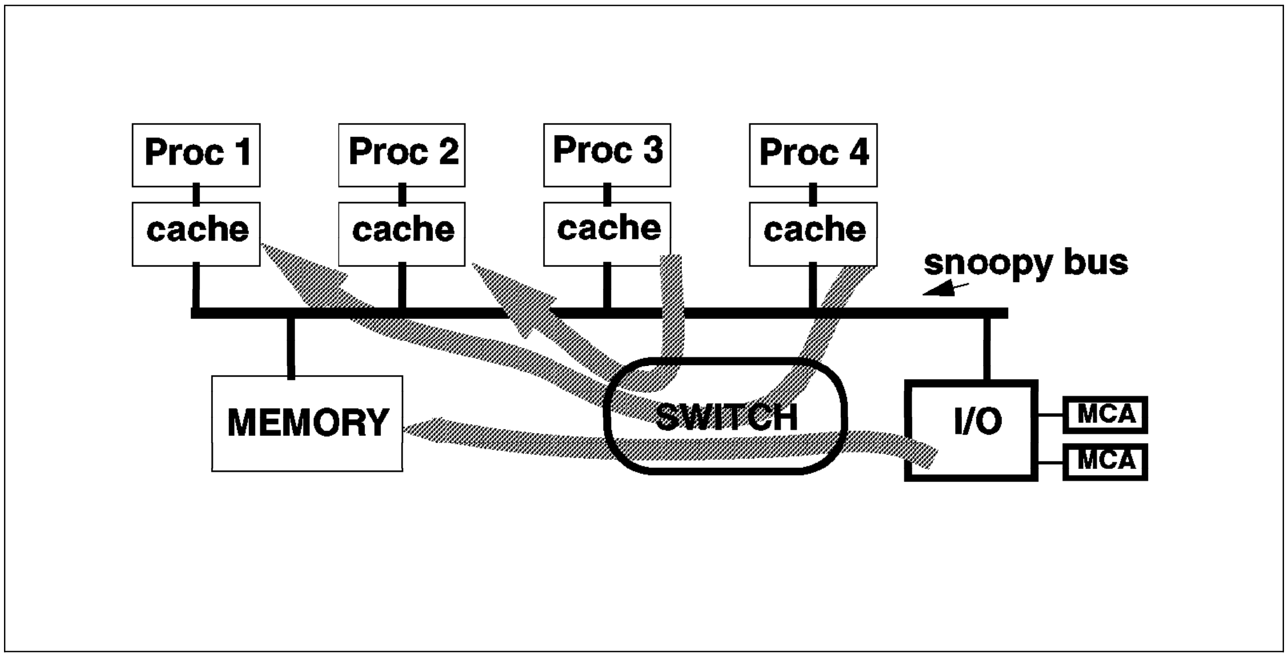


Figure 34. Using a Switch for Data Transfer

3.2.3 SMP Architecture Description

Figure 36 on page 60 describes the SMP architecture. This figures shows an SMP system with four CPU boards (each CPU board having two processors), the Data Crossbar switch, the sixteen 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 112 MB/s). The memory is totally shared by all of the processors. Each processor can access any memory modules through the Data Crossbar switch.

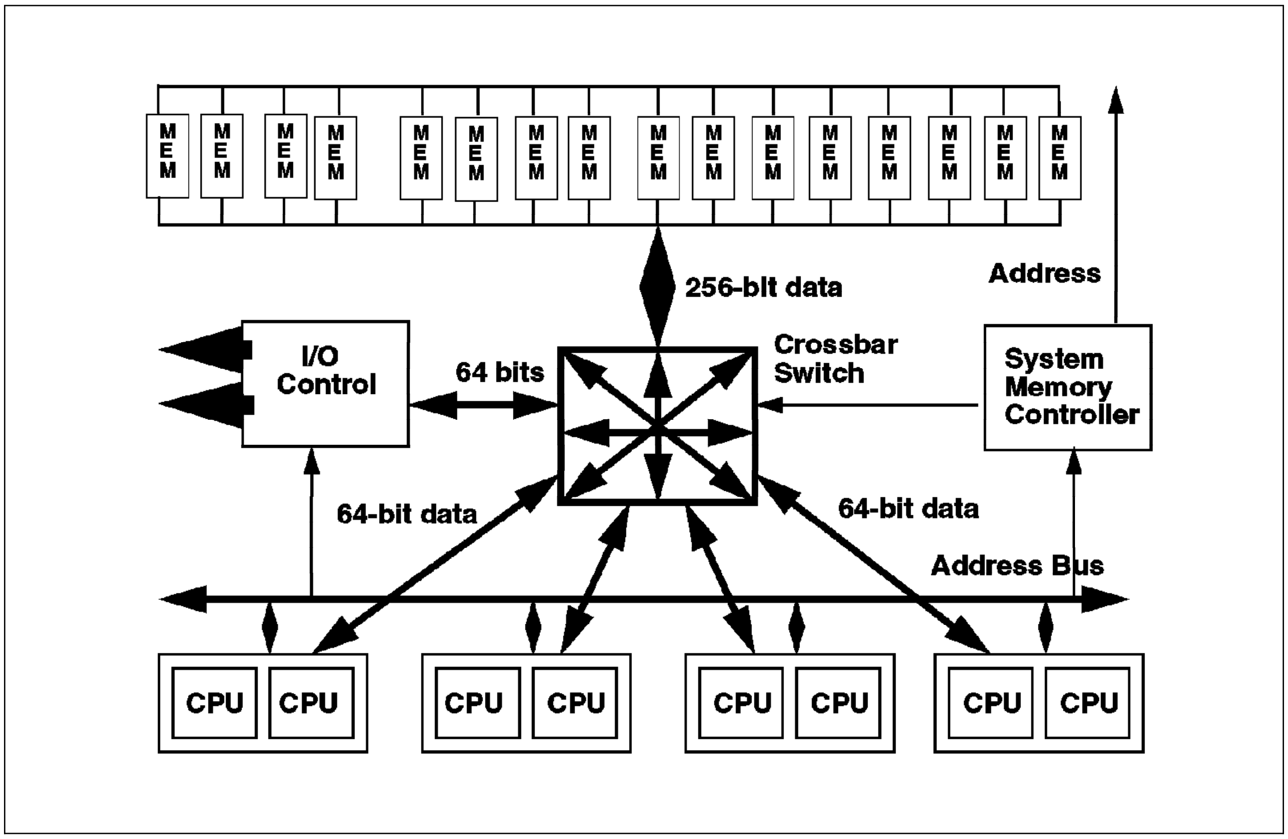


Figure 35. SMP Architecture

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* on 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 sixteen-way.

The most important point here is that interleaving is automatically optimized by the system at the 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, sixteen-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 sixteen banks. The size of a bank can be 32 MB, 64 MB or 128 MB. Also, the architecture already provides support for 256 MB and 512 MB banks that will use future 64 Mb memory technology.

A system can also have banks with different sizes. In this case, the system will automatically optimize the interleaving scheme.

Figure 36 on page 60 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.

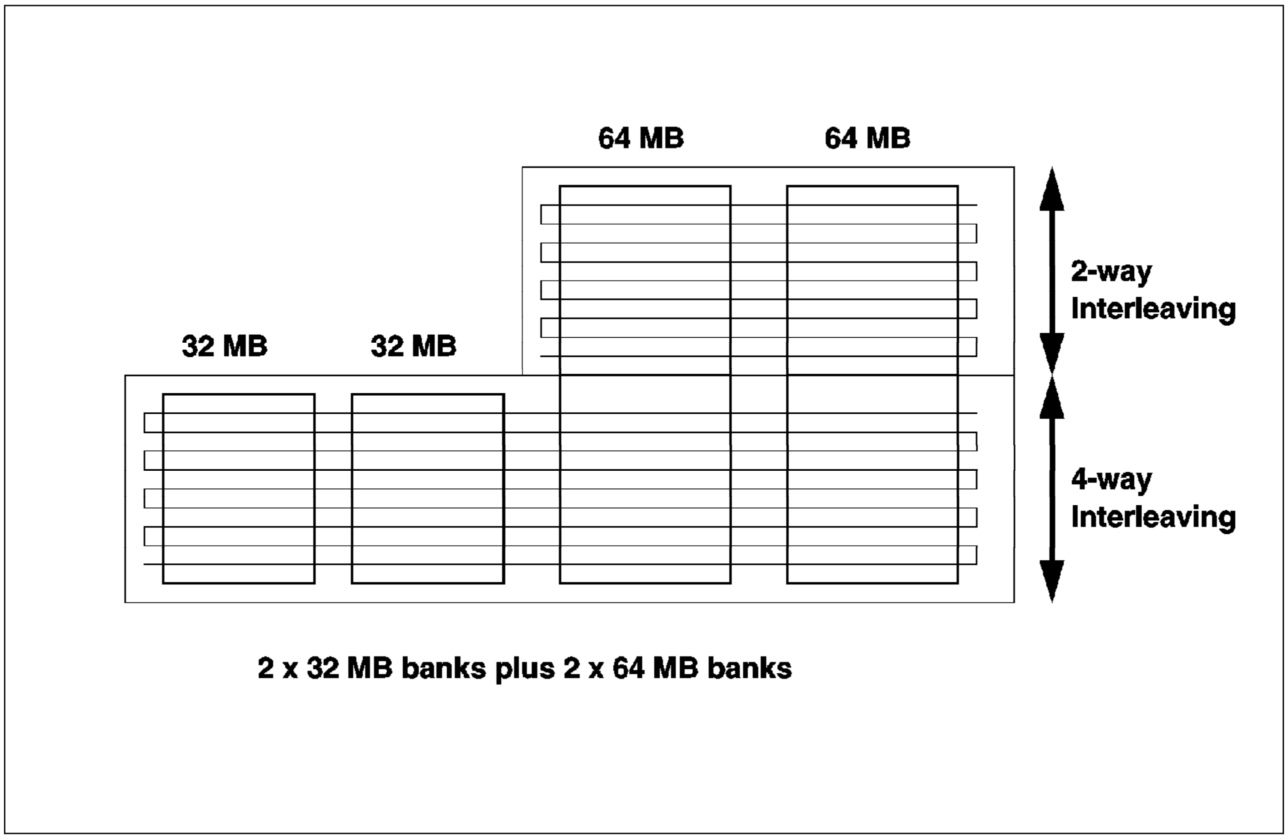


Figure 36. Interleaving Optimization

Again, the interleaving technique on the IBM SMP system is extremely advanced.

3.2.6 Crossbar Main Characteristics

The Non-blocking Data Crossbar switch provides four CPU ports in the J30 and R30 and two CPU ports in the G30. 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.

3.2.7 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.

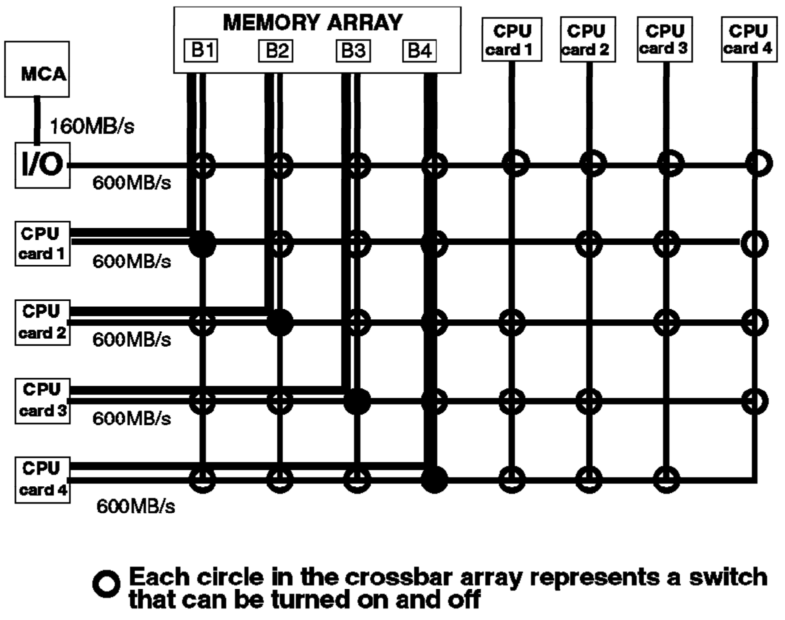


Figure 37. Crossbar Switch Interconnection

Figure 38 on page 62 shows the interconnection between the I/O subsystem and the memory, CPU cards 1 and 4 and CPU cards 2 and 3.

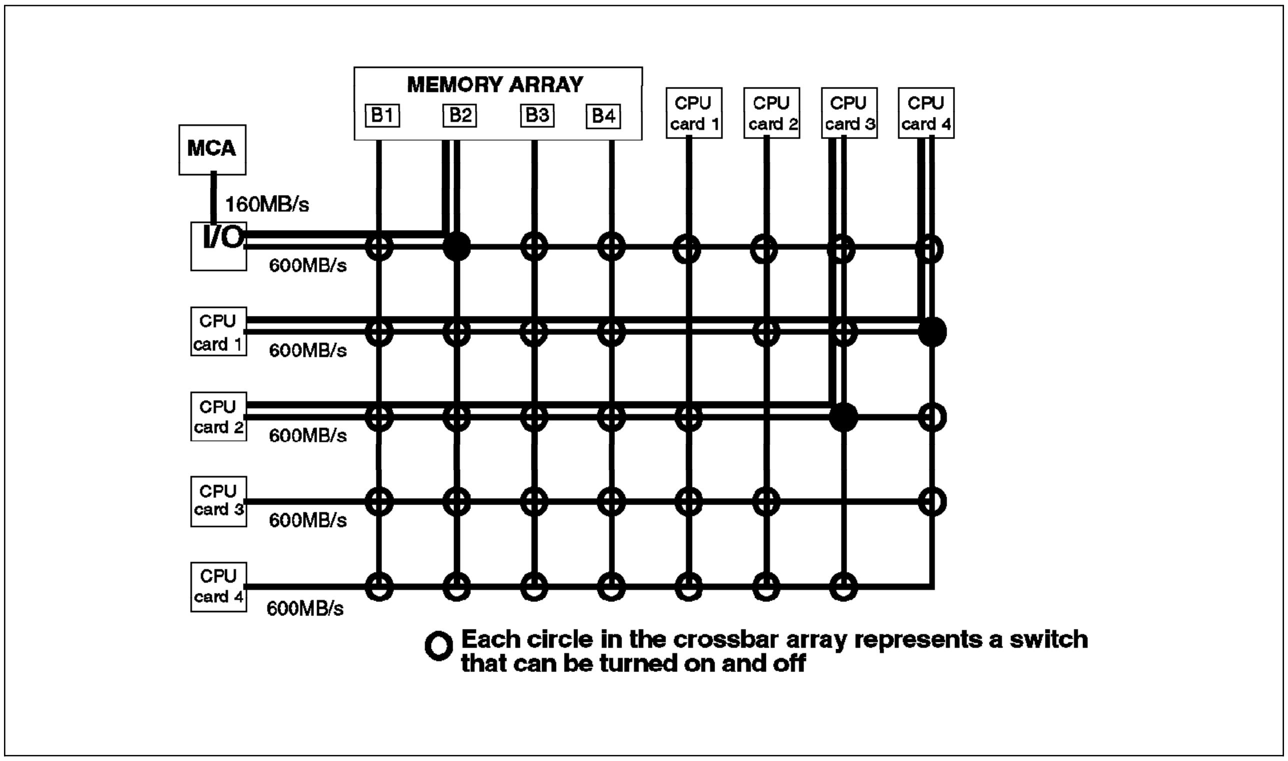


Figure 38. Crossbar Switch Interconnection

3.2.8 Crossbar Architecture

Figure 39 on page 63 shows the architecture of the crossbar in a configuration with four CPU cards (eight processors).

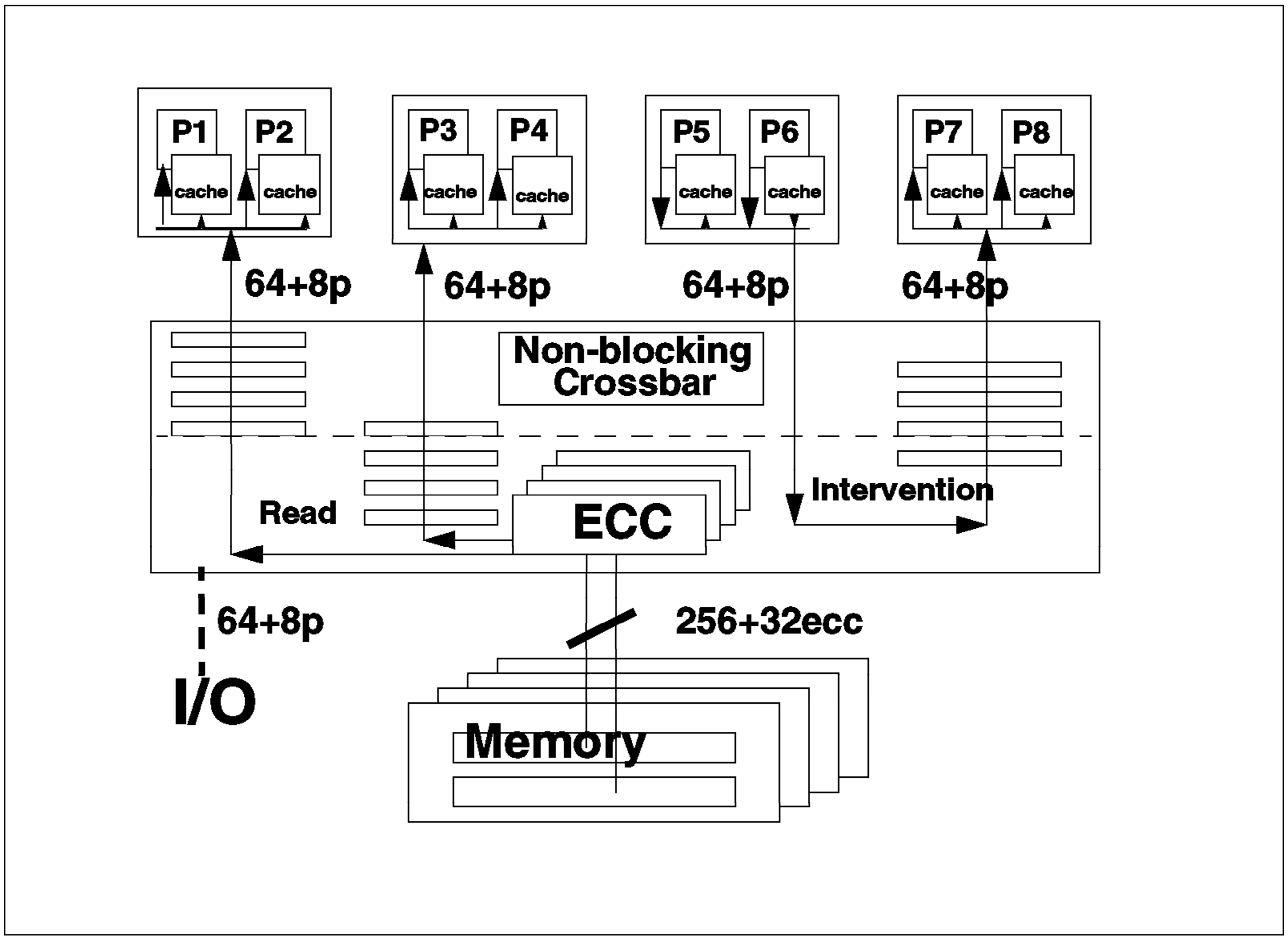


Figure 39. Crossbar Architecture

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 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 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 operations and operates in conjunction with the data crossbar.

3.2.9 Crossbar Performance Characteristics

The internal clock used throughout the system is 75 MHz (in the actual G30, J30 and R30 implementation).

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.

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.

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 on page 63. 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.

Note: This peak rate can be obtained when the system is equipped with four CPU cards. It will not be reached on a G30, which is limited to four processors (two CPU cards).

3.2.10 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.

3.2.11 Crossbar Operations

This is a description of the different crossbar operations. Figure 40 on page 66 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.

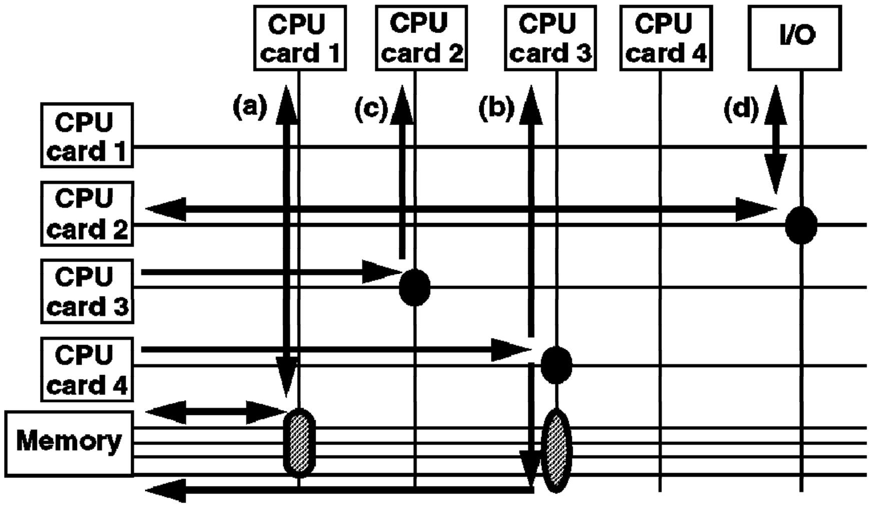


Figure 40. Crossbar Operations

3.2.12 Crossbar Advantages Summary

This is a summary of the crossbar advantages.

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 one port and the memory array, all the bandwidth is available. Several transfers occur at the same. For example, two CPU-to-memory transfers can occur at the same time as two CPU-to-CPU transfers.

While several transfers occur at the same time, the crossbar provides a high degree of overlap; this means that the instantaneous bandwidth can be very high (1800 MB/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.

Since the bandwidth is very high and latency is fixed, the crossbar provides the IBM SMP systems with a very high scalability factor.

3.3 Architecture Implementation

Figure 41 on page 68 shows the physical implementation of the SMP architecture. Following is a description of the different components:

- **Multiprocessor Board (MPB):**

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

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

  Each PowerPC 601-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 601 processors to one based on 604 processors by substituting the CPU boards.

  Note: Mixed 601 and 604 configurations are not supported.

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

  The IOD acts as the bridge between the MCA busses 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 is made with 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 to 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 th MPB_SysBus (Multiprocessor Board System Bus) data path to the memory subsystem, to the CPU boards and to the I/O board.

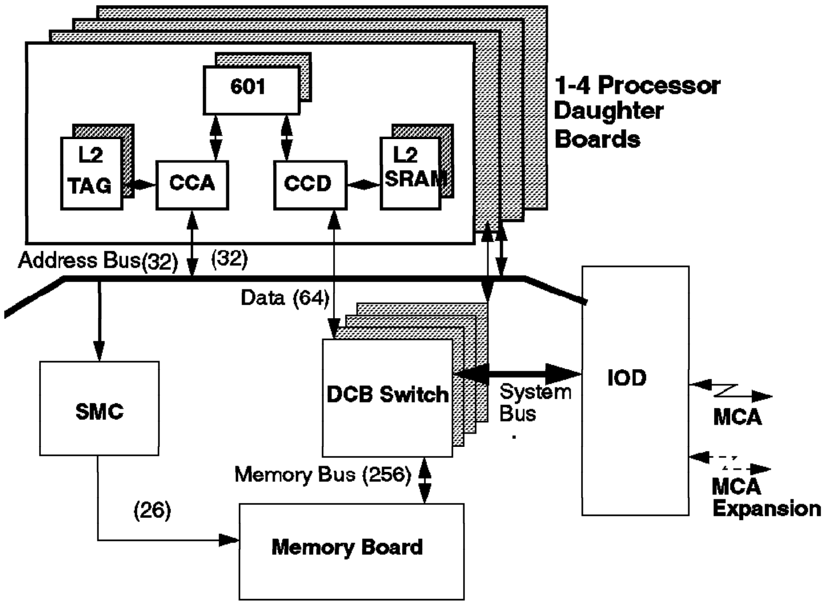


Figure 41. SMP Architecture Implementation

3.4 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) 1M, 2M and 4M x 72 bit ECC SIMMs (Single 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 SIMMs memory used on the SMP systems are 72-bit SIMMs, whereas the 128/256MB 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 a 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 G30. Each board is considered as a row of the global memory array. Memory boards are increments of one.

A row is made with one, two or four banks of SIMM memory. Up to 16 banks (from four memory boards) are supported in the system.

Different memory boards have a different number of memory banks:

In the G30:

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

In the J30 and R30:

- 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 G30 that has only one memory slot).

If a memory bank is 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 memory board's configuration. The best interleaving scheme is adopted when there are two, four, eight, or 16 banks of homogenous DRAMS.

Chapter 4. SMP Servers Hardware Features

This chapter introduces the main hardware features of the IBM family of SMP servers. It will highlight some of the hardware specifics that might help you in configuring or implementing these systems.

4.1 IBM RISC System/6000 SMP Servers Family

The IBM RISC System/6000 SMP servers family consists of three models: G30, J30 and R30. The G30 is a mini-tower model, the J30 a deskside model and the R30 a rack drawer. All these servers are based on the PowerPC technology. They are supported by AIX V4.1.2 or later.

The three SMP models incorporate a technology which is new for the IBM RISC System/6000 product line: the switch or Data Crossbar (DCB). This technology provides a dedicated high-performance communication path between the processors and the main memory. It also allows a very high bandwidth between the processors and the main memory, and this results in a very high scalability factor.

The SMP family also introduces a service processor called SystemGuard. This service processor has its own firmware. It is responsible for controlling the SMP hardware at boot time or 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 99 in this redbook.

Another feature offered with the SMP family is the Cluster Power Controller (CPC). Even though the CPC has been introduced with the 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 systems are intended to be commercial servers. Only the G30 can support a graphic display, a keyboard and a mouse. The J30 and the R30 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. One port can be dedicated to a modem connection, allowing remote support. The G30, like the C10, has only two physical ports; so the first port can be split into serial port S1 and S2 through a splitter cable.

4.2 Model G30 Server

The IBM RISC System/6000 Model G30 is the smallest SMP system in the SMP family. It comes in a mini-tower cabinet.

The G30 comes standard with two 32-bit PowerPC 601 processors and 32 MB of memory. Despite its small size, the G30 can be made more powerful by adding two more CPUs (one more processor card), for a total of four CPUs, and by increasing the memory size, for a total of 512 MB of memory. The system has one memory slot and two processor card slots (each processor card having two processors).

It has a total of six microchannel slots, five of which are available for optional adapters.

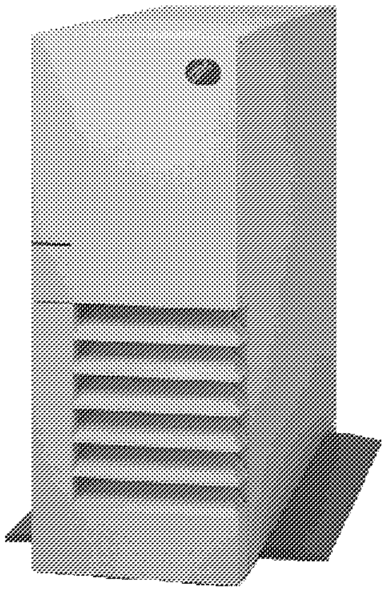


Figure 42. IBM RISC System/6000 Model G30 SMP Server

Because of its small size and compactness, the G30 can be configured as a stand-alone desktop system or as a high-performance server in a commercial environment. The G30 model is a type 7012, which is the same type as the desktop models, such as the 3xx series.

4.2.1 Standard Configuration

The standard configuration for the IBM RISC System/6000 Model G30 is the following:

- Two 75 MHz PowerPC 601 processors
- 0.5 MB L2 cache per processor
- 32 MB of memory, but a 64 MB card can be selected at no charge instead of the 32 MB card
- One SCSI-2 Fast/Wide SE controller
- Five available microchannel slots
- 1.1 GB SCSI-2 disk, but a 2.2 GB disk can be selected at no charge instead of the standard 1.1 GB
- Quad-speed SCSI-2 CD-ROM drive
- 1.44 MB, 3.5 inch diskette drive

4.2.2 Hardware Features

Following are the hardware features and expansion capabilities of the IBM RISC System/6000 Model G30.

- Two or four 75 MHz PowerPC 601 processors
- Caches:
  - Level 1 cache (instructions/data): 32 KB per processor
  - Level 2 cache (instructions/data): 0.5 MB per processor
- Memory:
  - One slot of memory
  - Memory bus width: 256 bits
  - 72-bit JEDEC SIMMs
  - 32 MB, 64 MB, 128 MB, 256 MB and 512 MB cards available
  - Up to 512 MB
- Standard SCSI-2 Fast/Wide controller
- Internal storage capacity: 1.1 GB to 17.9 GB

  Note: This internal storage capacity can be reached when the second processor card is replaced by a disk tray supporting two disks. A four-way G30 supports up to 13.5 GB internally.
- Storage/media bays:
  - Two full-height, 3.5 inch disk bays
  - Two half-height, 5.25 inch media bays
- Disk drives available:
  - 1.1 GB SCSI-2
  - 2.2 GB SCSI-2
  - 4.5 GB SCSI-2
- Tape drives:
  - 5 GB, 8 mm
  - 4 GB, 4 mm
  - 1.2 GB, 1/4-inch cartridge
- Internal diskette drive: 1.44 MB, 3.5 inch
- One microchannel bus at 80 MB/s with six slots
- Five available microchannel slots (the sixth slot is taken over by the standard SCSI-2 F/W controller).
- One SCSI-2 F/W for external devices
- Three serial ports (S1/S2 are physically in one 25-pin port which is split by a splitter cable, part number 31F4126). This splitter cable is normally shipped with the system.
- One parallel port
- One power control port (RS-485 for connecting the G02 expansion unit)

4.2.3 Additional Information on the G30 Server

Figure 43 is a picture of the G30.

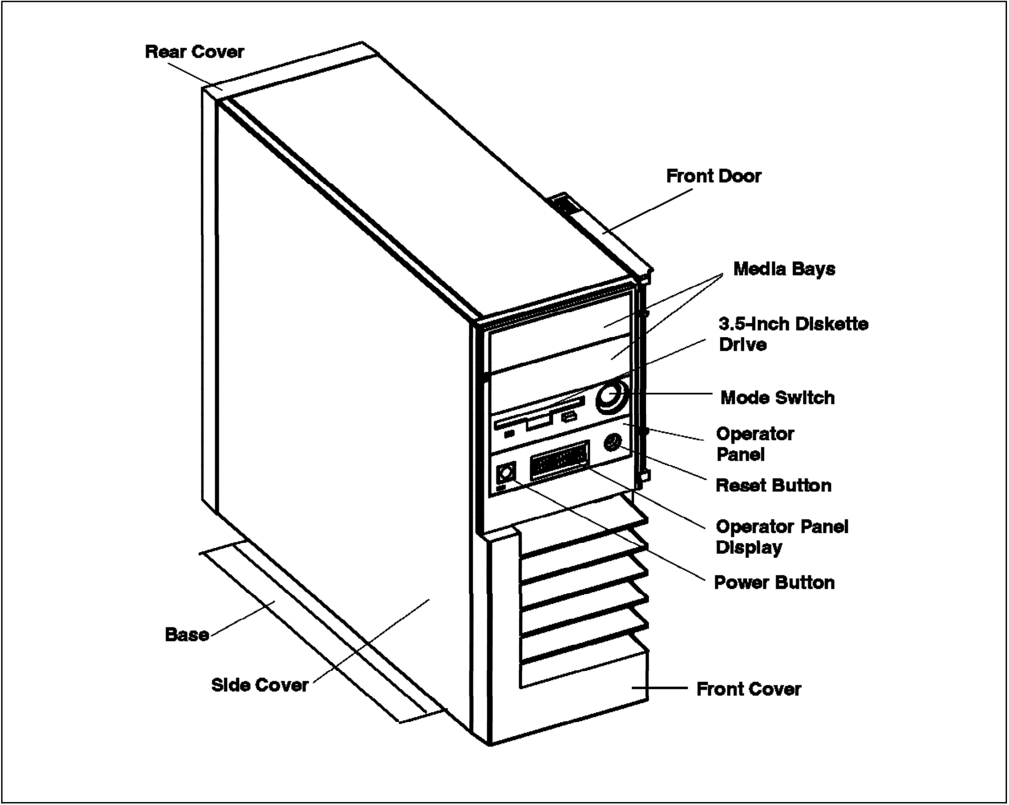


Figure 43. Model G30 SMP Server

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

- Event indications and problems reporting during Power-On tests and configuration methods
- 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
- Runtime problem reporting

The G30 has the following dimensions:

- Height: 450 mm (17.5 inches)
- Depth: 613 mm (28.2 inches)
- Width: 173 mm (6.9 inches)

Figure 44 on page 75 gives an internal view of the G30 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.

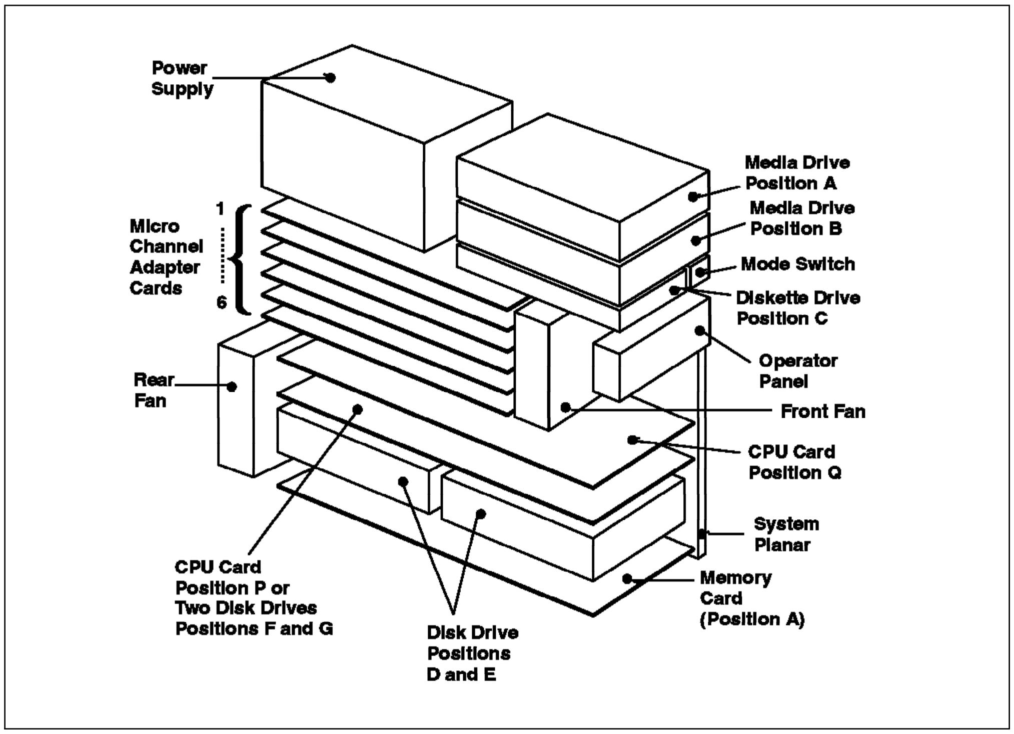


Figure 44. Internal View of the Model G30 Server

You can see in this picture that the G30 has two physical locations for processor cards (position Q and P). Note that an additional two-disk bay can be added in lieu of the second processor card (card P) using FC #6510.

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 FC #6511. Note that this feature code is added automatically when more then two disks are requested.

Thus, by using FC #6510 and #6511, you can have up to five disks installed on the machine.

Figure 45 on page 76 shows the rear of the G30.

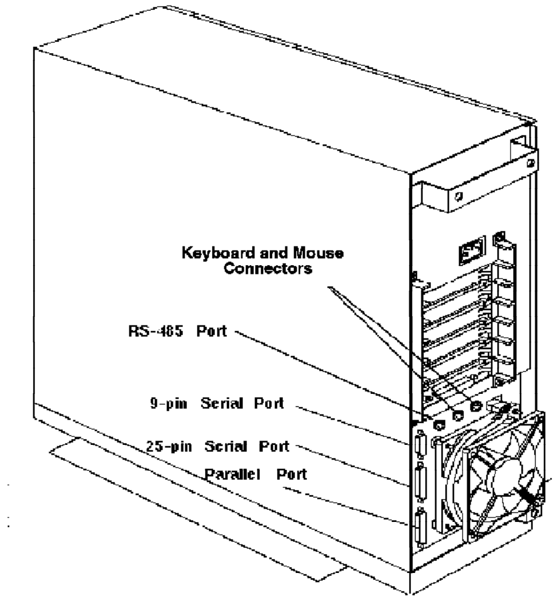


Figure 45. Rear View of the Model G30 Server

You can see that the G30 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 (FC #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 99 in this redbook.

In the G30, all the internal media and disk devices have fixed SCSI addresses. Following are the SCSI device locations and their addresses.

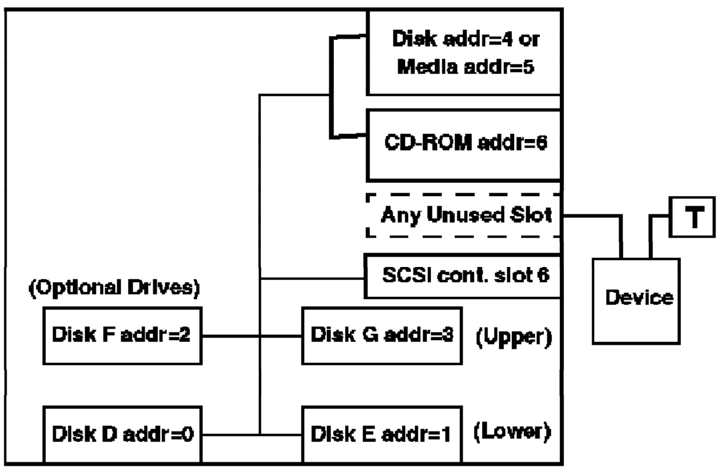


Figure 46. SCSI Device Address and Location for the G30 Server

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

On the G30, the G02 expansion unit is used to increase the number of media devices and/or hard disks in the G30.

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 G30. A G30 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 FC #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 G30.

The 6-pin RS-485 port present in the rear of the G30 must be connected to the G02 for power control.

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.

4.4 Model J30 Server

The IBM RISC System/6000 Model J30 is a deskside system which is more expandable than the G30.

It features four slots for processor cards and four slots for memory cards. As a result, the J30 is designed to support up to eight CPUs and 2048 MB of memory.

The system comes standard with two PowerPC 601 CPUs, 64 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 J30 has hot-pluggable disks. Disks in the base cabinet, as well as disks in the expansion cabinet, are hot-pluggable. This means that if a disk were to fail, it can 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 model J30 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 IBM RISC System/6000 systems, an upgrade path is offered to the model J30.

Figure 47 on page 79 is a picture of the J30 base unit. The J30 model is very different from the existing 5xx models as far as the industrial design, packaging and expansion capabilities are concerned. The J30 is a deskside model so it has the type 7013, just like the 5xx models.

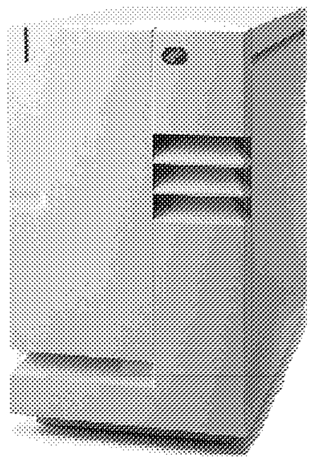


Figure 47. IBM RISC System/6000 Model J30 SMP Server

4.4.1 Standard Configuration

The standard configuration of the IBM RISC System/6000 Model J30 is the following:

- Dual 75 MHz PowerPC 601
- 1 MB L2 cache per processor
- 64 MB of memory, but a 128 MB card can be selected instead at no additional charge
- Six available microchannel slots
- SCSI-2 Differential Fast/Wide controller
- 2.2 GB SCSI-2 disk drive, but a 4.5 GB SCSI-2 disk can be selected instead at no additional charge.
- CD-ROM drive
- 1.44 MB, 3.5 inch diskette drive

4.4.2 Hardware Features

Following are the hardware features and expansion capabilities of the J30:

- Four CPU slots
- Two, four, six, or eight-way 75 MHz PowerPC 601
- Caches:
  - Level 1 (instructions/data): 32 KB per processor
  - Level 2 (instructions/data): 1 MB per processor
- Memory:
  - Four slots
  - Bus width: 256-bit
  - 72-bit JEDEC SIMMs modules
  - 64 MB, 128 MB, 256 MB and 512 MB cards available

    - Up to 2 GB of memory

- SCSI-2 Differential Fast/Wide controller
- Internal storage capacity: 2.2 GB to 40.5 GB

  Note: This internal storage can be reached if two of the media bays are used for installing disks.

- Disk drives:
    - 1.1 GB SCSI-2
    - 2.2 GB SCSI-2
    - 4.5 GB SCSI-2
- Tape drives:
    - 5 GB 8 mm
    - 4 GB 4 mm
    - 1.2 GB 1/4-inch cartridge
- Storage/media bays:
    - Seven half-height, 3.5 inch disk bays
    - Three half-height, 5.25 inch media bays
- Internal diskette drive: 1.44 MB, 3.5 inches
- Six available microchannel slots (the seventh slot is taken over by the SCSI-2 Differential F/W controller)
- A System Interface Board (SIB) providing the following I/O ports:
    - Two 68-pin SCSI-2 Differential F/W connectors marked A and B (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 and OUT 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 Uninterruptible Power Supply (UPS)
    - One parallel port

Figure 48 on page 81 shows the System Interface Board (SIB) which is located at the rear of the J30. You can see the two Differential SCSI-2 F/W connectors that terminate the internal SCSI-2 Differential F/W 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.

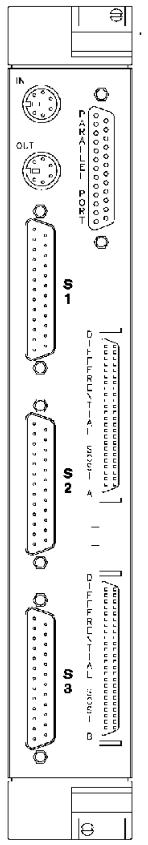


Figure 48. Rear View of the System Interface Board on J30 Server

4.4.3 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 often 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 82 is an internal front view of the J30. 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.

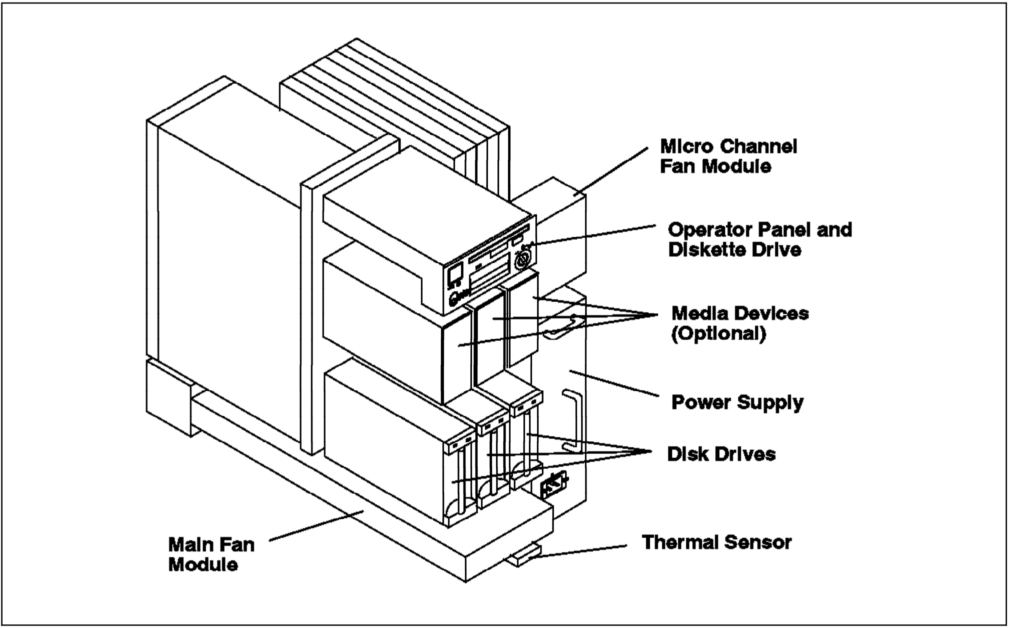


Figure 49. J30 Front Internal View

There is a thermal sensor located at the bottom, front of the system. If the thermal sensor senses the temperature inside the system to be above the operating limit (+50 C), it will send a warning message and then shut down the system after a few minutes. Figure 50 shows the rear of the J30. Four disks can be installed at the rear of the system. You can also see the SIB located at the back of the system.

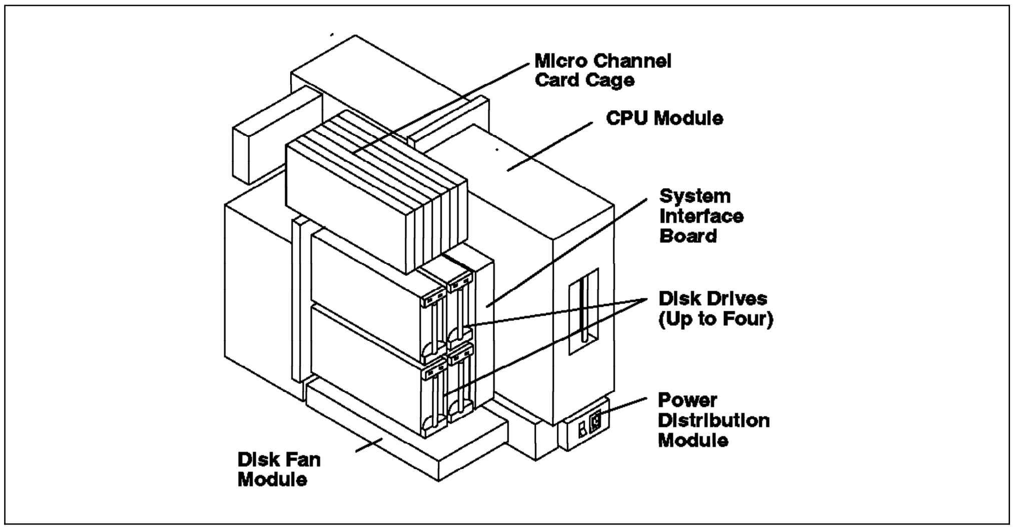


Figure 50. J30 Rear Internal View

4.4.4 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. You will find under this menu a new option which is the Removable Disk Management. 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
  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 filesystems 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 content 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 content 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?
    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.

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 filesystems 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 filesystems 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.

4.4.5 J30 SCSI Device Addresses

As in the G30, SCSI devices addresses in the J30 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 a 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 (FC #2416) and an SCSI internal cable (FC #2441) are required.

Following is an example of a disk address:

```
hdisk0  Available  X0-07-00-3,0  1.1 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.

Figure 51 on page 86 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).

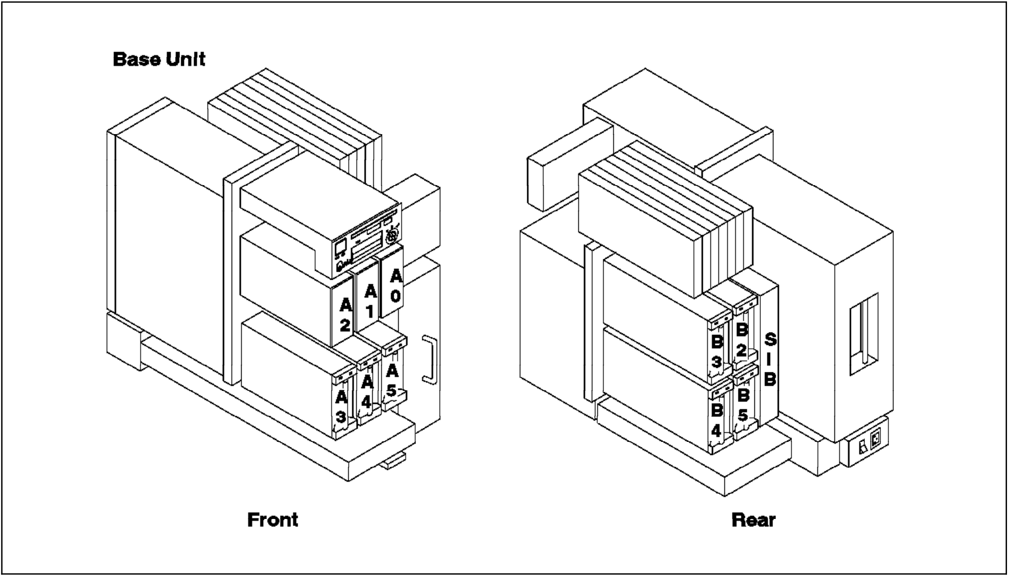


Figure 51. Front and Rear SCSI Devices Locations on the J30 Server

4.5 Model J01 Expansion Cabinet

The J01 is an expansion unit for the J30 that expands the number of microchannel slots. The J01 provides eight additional microchannel slots which allows the J30 to reach a total of fifteen (15) microchannel slots. Thus, more microchannel adapters and more devices can be supported on the J30.

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.

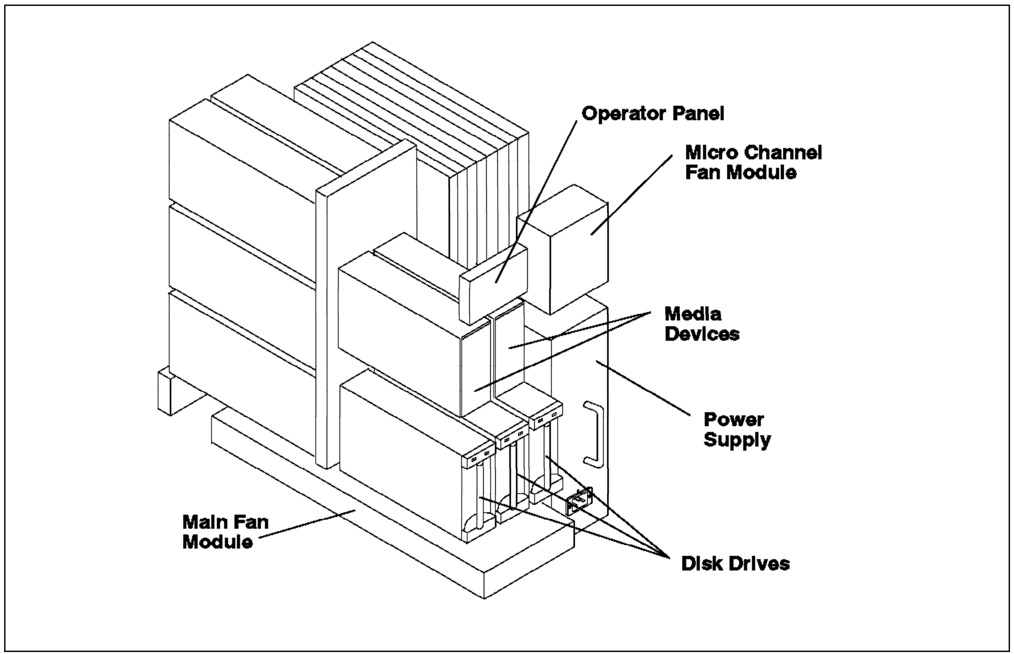


Figure 52. Front Internal View of the J01 Expansion Cabinet

Figure 53 on page 87 shows that the J01 has nine disk bays at the back of the unit.

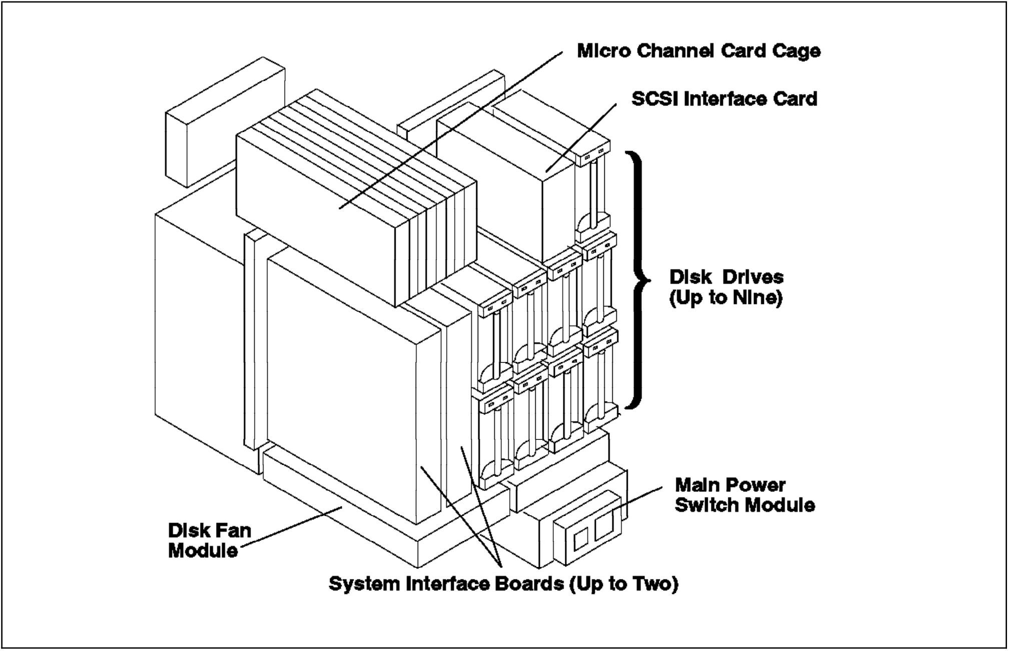


Figure 53. Rear Internal View of the J01 Expansion Cabinet

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 DE F/W adapter (FC #2416)
- Two SCSI cable for internal devices (FC #2441)
- Two media-to-disk bay conversion kits (FC #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 then 10 devices are installed in the expansion unit, a second SIB is required.

As in the J30 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 4.4.4, "Hot-Pluggable Disk Configuration Considerations" on page 83 in this chapter.

As in the J30, 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 to establish a connection between the SCSI-2 Differential F/W controller located in one microchannel slot and the backplane where are located the media or disk drives connectors. There is one BHS per expansion cabinet.

The J01 has the same dimension as the J30, that is:

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

4.5.1 J30 and J01 Interconnection

Figure 54 shows a J30 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.

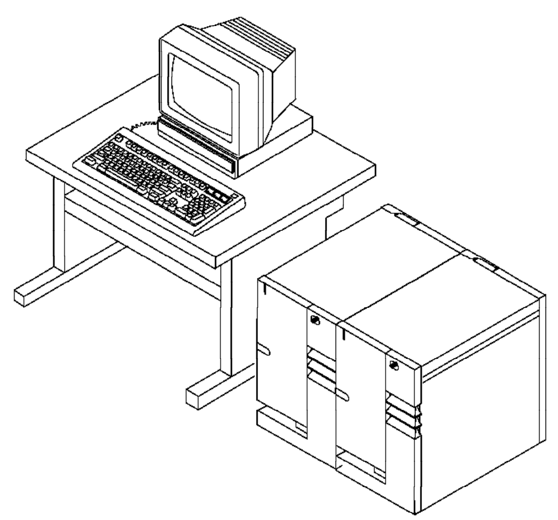


Figure 54. J30 and J01 Interconnection

To connect the J30 with the J01, follow this procedure:

- Connect the RS-485 cable between the J30 and the J01 (OUT port of J30 to IN port of J01).
- Connect the terminators to the IN connector on the J30 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 (FC #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 FC #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 the expansion unit cannot be used for connecting external devices.

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.

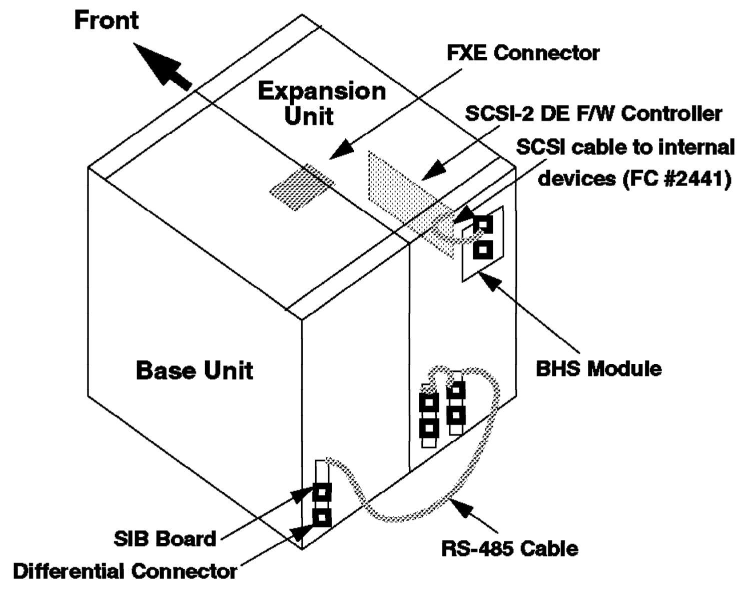


Figure 55. J30 and J01 Interconnection

4.5.2 Model J30/J01 Specifics

You will find below some specifics of the J30 and J01 units.

First, to be able to boot a J30 system, the top cover must be closed. The J30 cannot boot if the top cover is not in place. In fact, the top cover, when installed, pushes a 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 J30). 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.

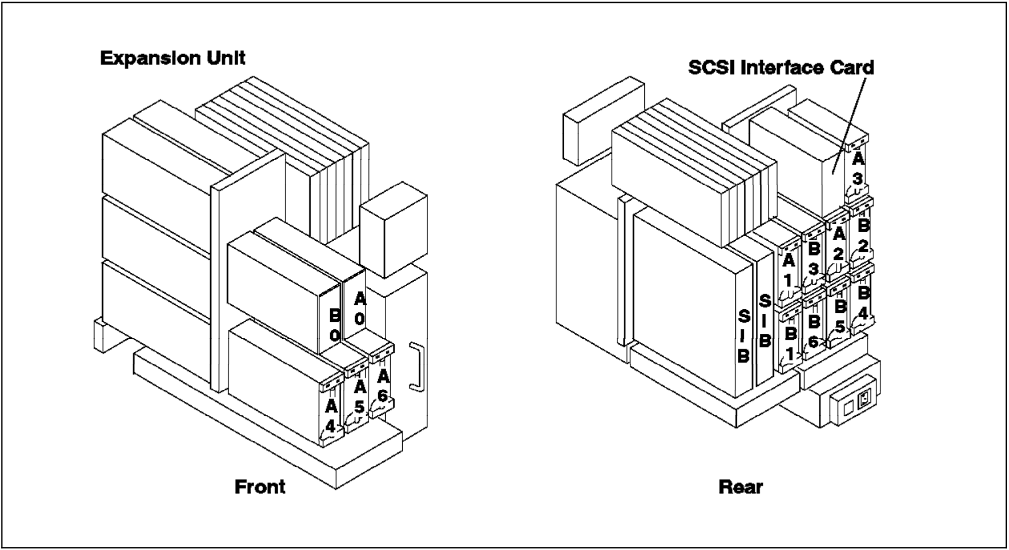


Figure 56. J01 SCSI Device Location and Addresses

4.6 Model R30 Rack Server

The IBM RISC System/6000 Model R30 is a drawer unit that can be mounted in an industry standard rack.

Like the J30, the R30 comes with four slots for processor cards and four slots for memory cards.

However, the R30 comes standard with a total of sixteen microchannel slots, fifteen 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 R30 is similar in design to the J30. The strong focus of the R30 is on high availability. The R30 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 Uninterruptible Power Supply (UPS).

The R30 drawer uses the same rack, R00, as the R10, the R20 or the R24. This allows four R30 drawers to fit into the same rack, which is very suitable for an HACMP cluster.

Existing IBM RISC System/6000 models, such as the 99X, R10, R20, and R24, can be upgraded to the SMP server R30.

Figure 57 is a picture of the R30 drawer installed in a rack.

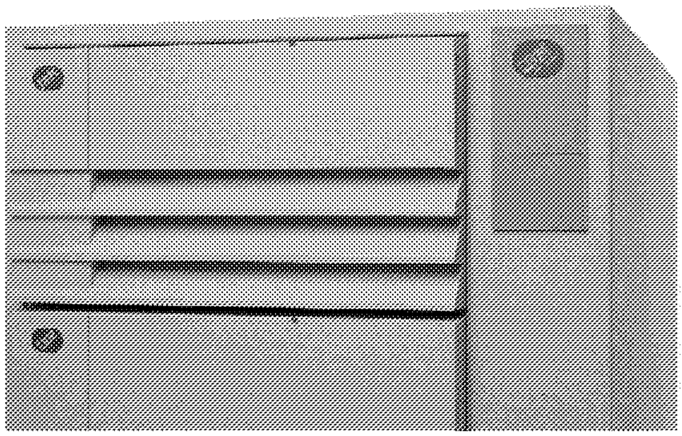


Figure 57. IBM RISC System/6000 Model R30 SMP Rack Server

4.6.1 Standard Configuration

The standard configuration of the R30 is the following:

- Two 75 MHz PowerPC 601 processors
- 1 MB L2 cache per processor
- 64 MB of memory, but a 128 MB card can be selected instead at no additional charge
- Fifteen microchannel slots available
- 1.1 GB SCSI-2 internal disk, but a 2.2 GB disk can be selected instead at no charge
- CD-ROM drive
- SCSI-2 Fast/Wide controller
- 1.44 MB, 3.5 inch diskette drive

4.6.2 Hardware Features

Following are the hardware features and expansion capabilities of the R30 drawer:

- Four CPU card slots
- Two, four, six, or eight 75 MHz PowerPC 601 processors
- Caches:
  - Level 1 (instructions/data): 32 KB per processor
  - Level 2 (instructions/data): 1 MB per processor
- Memory:
  - Four slots (four cards)
  - Bus width: 256-bit
  - 72-bit JEDEC SIMMs
  - 64 MB, 128 MB, 256 MB and 512 MB cards available
  - Up to 2 GB of memory
- Standard SCSI-2 Fast/Wide SE controller
- Internal storage capacity: 1.1 GB to 2.2 GB
- Disk drives:
  - 1.1 GB SCSI-2
  - 2.2 GB SCSI-2
- Tape drives:
  - 5 GB, 8 mm
  - 4 GB, 4 mm
  - 1.2 GB, 1/4-inch cartridge
- CD-ROM drive
- External storage
  - Rewritable optical drives
  - CD-ROM drives
  - Disk drives:
    - 1 GB SCSI-2
    - 1.1 GB SCSI-2
    - 2 GB SCSI-2
    - 2.2 GB SCSI-2
  - Tape drives
    - 5 GB, 8 mm
    - 4 GB, 4 mm
    - 1.2 GB, 1/4-inch cartridge
    - 1/2-inch cartridge and tape reel
- Storage/media Bays:

  - Two media bays
  - One disk bay
- Fifteen available microchannel slots (the sixteenth slot is taken by the SCSI-2 F/W SE controller)
- Standard I/O Ports provided by the System Interface Board (SIB):
  - Three serial ports
  - Two RS-485 IN and OUT ports
  - One parallel port
  - One battery backup-unit connector

4.6.3 Additional Information on the R30 Server

This section provides some additional information on the R30 that might be useful when configuring, installing or using it:

- Redundant fans:

  There are a number of fans in the R30: three media fans, three CPU fans and four I/O module fans. If any one of these fans fail, the system will continue to run. No error message will be displayed on the console as a result of this fan failure. The exception is the rightmost fan in the I/O module. If this fan fails, a message will be displayed. Under BUMP, a fan failure can identify the group of fans (not right down to the fan that failed). Note that rebooting the system will also identify the failed fan. If a second fan fails, the system will perform a shutdown.

- Dual Power Supply Option:

  This provides an option for a second AC power supply, or for a DC power supply, for power redundancy.

- Uninterruptible Power Supply (UPS):

  The UPS is intended to replace the battery backup on most systems. The battery backup unit is still available, but should not be used for racks having:

  - One or more SMP drawers
  - Two or more UP drawers

  The UPS is no longer an RPQ (Request for Price Quotation). It has now the TM (Type Model) 9910-U33.

  This UPS is called Exide Powerware Prestige 3000 Rackmount UPS for IBM RISC System/6000. This UPS will protect hardware and software by detecting electrical anomalies and providing power for a limited period of time. The UPS full rated battery hold-up is 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 is running 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.

- Cluster Power Controller:

  A CPC option (FC 6175) is available on the R00 rack. Basically, the CPC allows the user to connect several CPUs or disk drawers and to have a single ASCII terminal as a console for all the CPUs. From this console, you can power-off or power-on each of the CPU 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 6, “Cluster Power Controller” on page 143 in this redbook.

- Layout:

  The layout of the R30 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 the J30.

  The R30 CPU enclosure is a rack-mounted CPU containing a CPU module, I/O module, media module, and a power supply with a possible extra, optional power supply.

  The rear of the R30 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 J30.

  The power supply on the right can be replaced with a cooling unit or a DC supply input.

  There are two microchannel boards, providing 16 slots.

Figure 58 shows an internal view of the front of the R30 drawer.

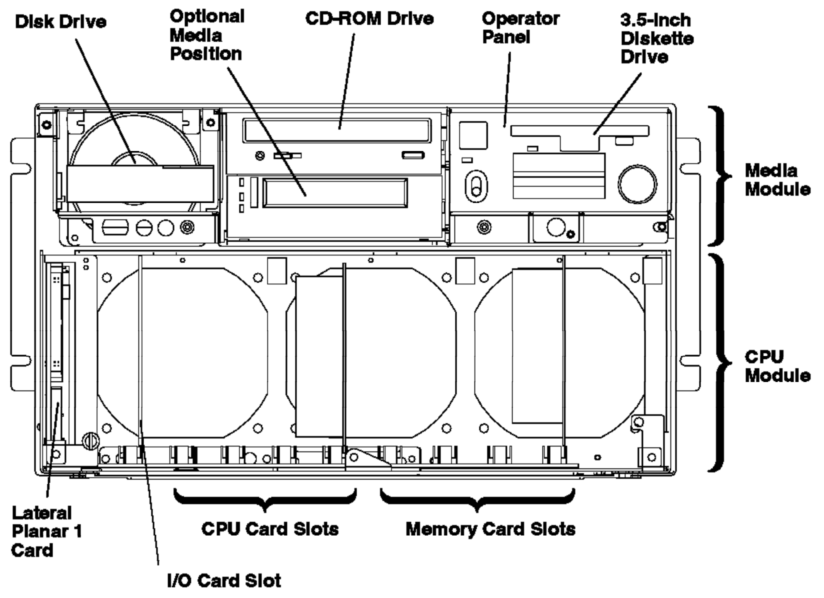


Figure 58. Front Internal View of the R30 Server

Figure 59 on page 95 shows the rear of the R30 drawer with its SIB (on the right).

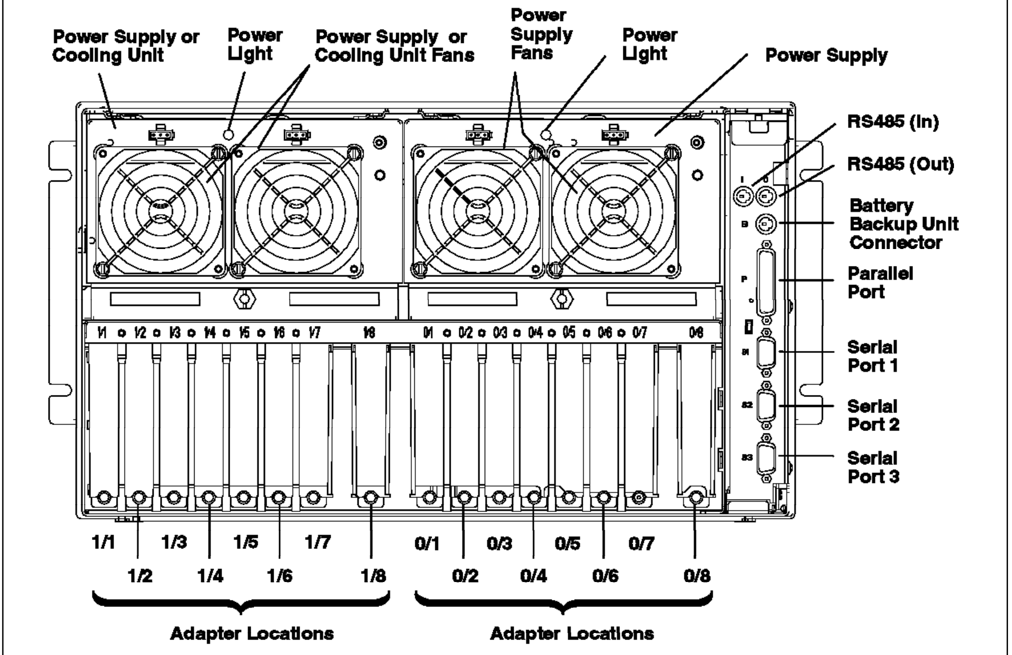


Figure 59. Rear View of the R30 Server

4.7 Model Conversion from UP to SMP

Model conversion from various existing uniprocessor systems to the three SMP servers is 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.

4.7.1 Model Conversion to G30

These are the available upgrade paths to the SMP G30:

- Models 340, 34H, 350, 360, and 370 are converted to a two-way G30
- Models 380 and 390 are converted to a four-way G30

  Note: For EMEA countries, only the 390 will be upgraded to a four-way G30.

New memory cards must be used for the G30. You should be aware of the following:

- Only one memory slot is available on the G30.
- Memory exchange options are available. Since old memory cards are not compatible with the G30, IBM exchanges currently installed memory cards for an equal or greater capacity of memory card compatible for the G30.
- 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 FC #4061 and FC #4062, respectively.

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

4.7.2 Model Conversion to J30

Model conversions from deskside models 5xx to the J30 are also offered. For all 5xx, except 58H, 590 and 59H models, the upgrade will come with one dual 601 75 MHz processor card, while for the 58H, 590 and 59H, the upgrade to J30 requires a second dual 601 processor.

The uniprocessor memory is not compatible with the existing J30 model, and a minimum of 64 MB memory is required for the SMP model J30.

These are the available upgrade paths to the SMP J30:

- Models 520, 52H, 530, 53E, 53H, 540, 550, 55E, 55L, 53S, 560, 56F, and 580 are upgraded to a two-way J30.
- Models 58H, 58F, 590, and 59H are upgraded to a four-way J30.

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 serial number is retained.

4.7.3 Model Conversion to R30

These are the available upgrade paths to the R30:

- Models 930, 950, 950E, 970, 970B, 970E, 970F, 980, 980B, 980E, and 980F are converted to a two-way R3U.
- Models R10 are converted to a two-way R30.
- Models 990, 990E, 990F, 99J, and 99K are converted to a four-way R3U.
- Models R20 and R24 are converted to a four-way R30.

New memory cards must be used in the R30. SIMMs from old 128 MB and 256 MB cards can be reused.

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

4.8 System Interface Board Functions

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

- A power microcontroller helps the BUMP 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 J30/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.

Finally, in the G30, the SIB is part of the system planar, while in the J30 and R30, it is in the form of pluggable modules.

Chapter 5. SystemGuard

This chapter introduces the SystemGuard service processor which is included in the IBM RISC System/6000 SMP servers models G30, J30 and R30.

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 also very important.

IBM's family of 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 was a memory error that could not be corrected, the system would detect this, reboot itself automatically and run without the bad memory component.

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

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

The main features of the SystemGuard are:

- Initialization process flow management
- Local as well as remote control of the system (power-on/off, diagnostics, reconfiguration, maintenance)
- Console mirroring to make remote actions visible and controllable by the customer
- Dial-out to a support center in case of system boot failure
- Run-Time surveillance

5.2 SystemGuard Power

SystemGuard has its own power boundary. This means that even if the system power is off (power button of the system pushed out), SystemGuard is still powered on. This allows control of the system even though the system is down. The only way to power-off SystemGuard is to turn off a specific switch on the J30 and the R30, or pull out the power cord on the G30. In order to power-on the entire system, you must first power-on SystemGuard by turning on this switch or by plugging the power cord in on the G30.

5.3 SystemGuard Components

SystemGuard introduces new hardware and firmware components that you need to understand before installing or booting an SMP system.

New hardware components are:

- A microprocessor called BUMP (Bring-Up MicroProcessor) with its 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 I2C (Inter-Integrated Circuit) bus to existing components, such as the System EEPROM, 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 NVRAM, the Flash EPROM, the Backup EPROM, and the S1 and S2 serial ports.

Physically, the BUMP is located on the I/O board.

Figure 60 on page 101 shows the SystemGuard hardware components. The MPB board stands for Multiprocessor Board. The COP is the Common On-chip Processor which is used for testing the PowerPC chip (601). The IONANs are components that handle the MCA buses.

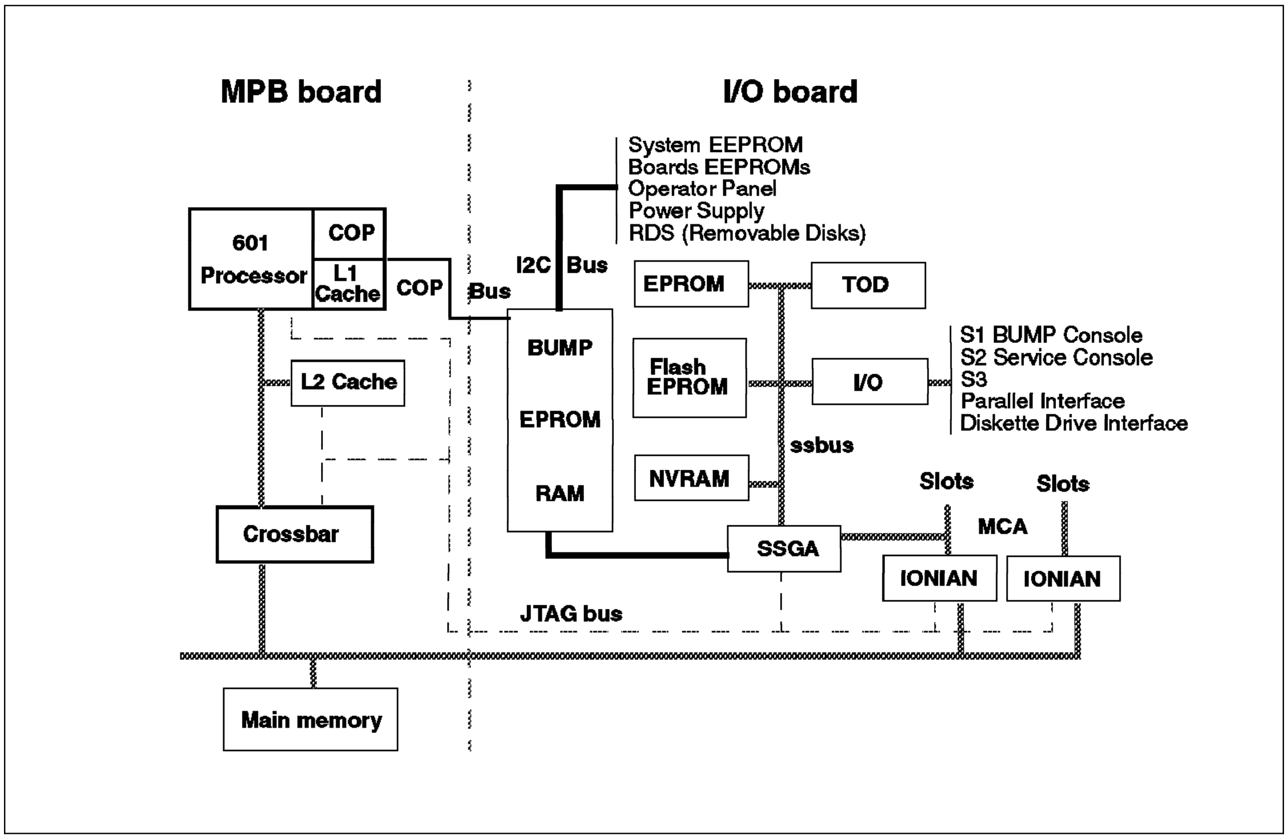


Figure 60. SystemGuard Hardware Components

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 LCD (Liquid Crystal Display) display.

The Operator Panel card contains the NVRAM, the NVRAM battery and the TOD (Time-Of-Day). It must be noted that 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 pushed in all the time if you want to be able to power-on or off the system remotely.
- Reset button: It resets SystemGuard to the Init phase and, depending on the key position, reboots the system to Maintenance or to AIX Multi-User.
- LCD display: It is made of two rows of sixteen characters. It displays the word `Stand-By` in the Stand-By phase, or it displays the usual three-digit bootup codes.
- 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.

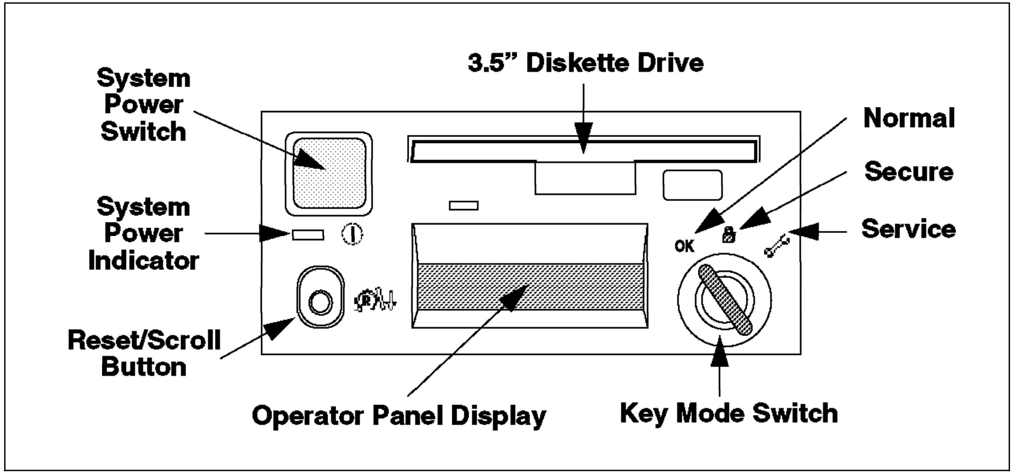


Figure 61. Operator Panel

5.5 SystemGuard Consoles

Apart from the Operator Panel, which can be considered as a small console and as a minimum interface to SystemGuard, SystemGuard works with two types of consoles:

- The *BUMP Console* is an ASCII terminal attached to the S1 serial port. This console provides the normal input to the BUMP. It can be local or remote. The line speed for the BUMP console must be set to 9600 bauds for either type of connection.
- The *Service Console* is an ASCII terminal attached to the S2 serial port. This console is usually remote and located in a Customer Support Center or an IBM Service Support Center. Basically, this console allows the support center to work with SystemGuard and/or AIX. Of course, the support center needs a specific authorization from the customer to access SystemGuard.

5.6 SystemGuard Functions

SystemGuard controls the system when:

- The system power is off. In this state, SystemGuard allows the system administrator or the service personnel to run specific tests, set SystemGuard parameters, reconfigure the system (CPU), or 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.

Access to SystemGuard can be done locally or remotely from the BUMP or from the Service Console through specific SystemGuard menus before the AIX is up as well as from AIX. These specific SystemGuard menus are the STAND-BY MENU and the MAINTENANCE MENU.

5.7 Physical and Electronic Key

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

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 the other IBM RISC System/6000 because of the control of SystemGuard. During bootup, SMP servers go through three different phases: Stand-By, Init and Run-Time.

5.8.1 Stand-By Phase

The Stand-By phase is present anytime the system power is off, and the BUMP is still powered.

If the system is not yet connected, the Stand-By phase is entered by plugging the unit into an electrical outlet that has power and by turning on a switch on the back of the J30 and the R30. The G30 does not have this switch; plugging the power cord in is sufficient.

At this 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 display.

The BUMP is active, and it can receive commands from the BUMP Console or Service Console (either local or remote). 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 and the `power` command entered.

5.8.2 Init Phase

Init phase is entered when the power-on button on the Operator Panel is pressed on or when the `power` command is entered on the BUMP Console or Service Console.

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

If the System Key is in Service mode, and if several conditions are met, the system loads the SystemGuard MAINTENANCE MENU. These conditions are: the Autoservice IPL flag disabled, the BUMP Console enabled and a Valid Service Contract.

If the System Key is in Secure mode, 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. The Stall state is exited, and control of the system is passed to AIX.

5.8.3 Run-Time Phase

This is the phase where the AIX operating system is in control of the system. The Run-Time phase is entered once the AIX operating system gets loaded. The control of consoles is handed over to AIX.

When AIX is stopped again, for example due to a shutdown, the system goes back to the Stand-By phase.

Figure 62 on page 105 shows the SystemGuard phases.

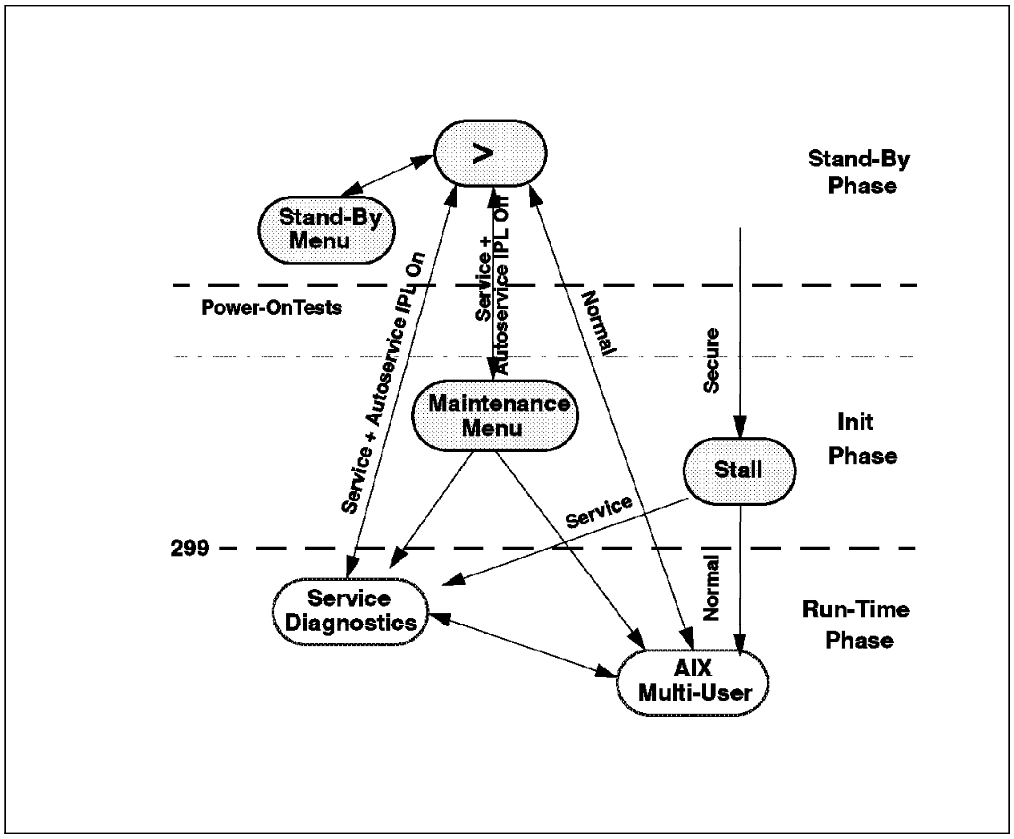


Figure 62. SystemGuard Phases

5.8.4 Phase Change (Stand-By to Init)

The phase change from Stand-By to Init is achieved by pushing the power button on the Operator Panel and by typing the keyword `power` at the Stand-By prompt (>). Note that if you type `power` while the power button is not pushed in, nothing happens until you press the power button. In this case, your `power` command has been taken into account by SystemGuard, and you don't have to reenter it.

`power` is the default power-on command sequence which can be changed by the system administrator from the MAINTENANCE MENU or from the Diagnostics' AIX SERVICE AIDS MENU.

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     - February 16, 1995
      ID 07.01 - POWER_ON in EPROM
#
      FLOPPY NOT READY!

      BUMP FIRMWARE     - May 19, 1995
      ID 07.04 - POWER_ON in FLASH PROM
```

Figure 63. Phase Change from Stand-By to Init

The message `FLOPPY NOT READY!` means that there is no specific downloadable diskette in the diskette drive.

The special diskette could be:

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

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:

- Tests on processors, caches, memory, and related hardware which cannot be turned off.
- Tests on other system resources which can be turned off by setting the Fast IPL flag on from the STAND-BY MENU.

Following is an example of PON tests output when the Fast IPL flag is on.

```
        BUMP FIRMWARE   - February 16, 1995
        ID 07.01 - POWER_ON in EPROM


        BUMP FIRMWARE   - May 19, 1995
        ID 07.04 - 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    - August 4, 1994
        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 will be displayed.

Note: The system will IPL on the first available physical processor. If for any reason processor 0 is not available, the system will IPL on processor 1. 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 possible, and go into the Stall state afterwards. No IPL will proceed. 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 resident 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 to 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.
- CPU Test Group: These tests, performed by all the processors, check the status of the CPU cards: processor, 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 these PON tests 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 Tests 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 Runtime)

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

Since SystemGuard is also giving up control of the two serial lines, nothing can be displayed on the consoles. The usual three-digit boot indicators are still displayed on the Operator Panel. Note that the code `570 virtual SCSI devices being` `configured` can take some time. It will say `walking the SCSI` and do several passes for each of the SCSI cards in the microchannel. This may take up to five minutes for each 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, this is no longer the BUMP Console; it is the AIX console.

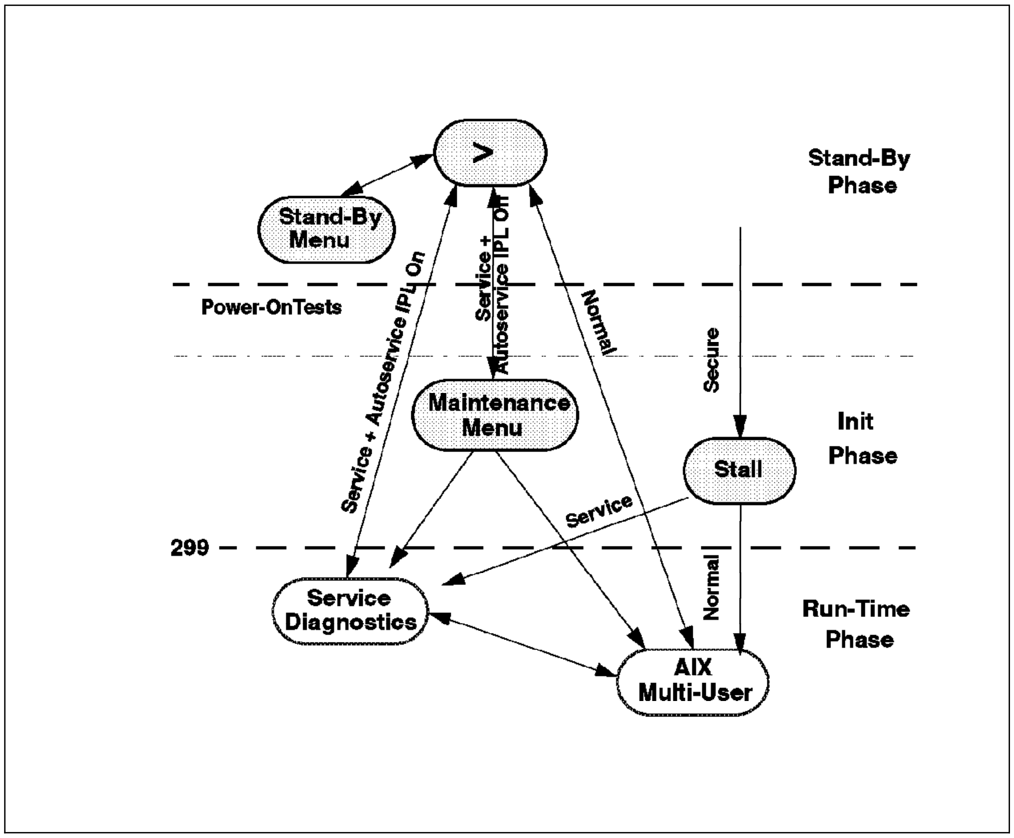


Figure 66. SystemGuard Phases

Please refer to the Service and Operator guides for the three-digit codes and test groups.

5.9 SystemGuard Parameters and Flags

A certain number of SystemGuard parameters and flags can be changed through different SystemGuard menus, from the Diagnostics interface and from AIX. Basically, there are four different groups of flags:

- Service support flags: These flags enable Service Console usage, maintenance usage and determine if dial-out messages will be sent to IBM or to a Customer Service Center. These flags are stored in the SID (System Identification) field of the System EEPROM.
- Diagnostics flags: These flags are used to control the service, diagnostics and maintenance from a customer point of view. For example, the customer can modify one of these flags to authorize setting the Electronic Key from the Service Console or to authorize the dial-out.

- Modem and Site Configuration flags: These flags allow the customer to customize modem configuration for the Service Console.
- Phone numbers flags: These are 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 locations. They can be changed from the SystemGuard STAND-BY MENU, the SystemGuard MAINTENANCE MENU, the Diagnostics interface, and also from AIX.

The STAND-BY MENU is stored in the BUMP EPROM. The MAINTENANCE MENU is stored in the Flash EPROM.

Getting into these various menus or interfaces depends on the way some flags are set before booting up the system.

It is important to understand the flowchart in Figure 67 on page 111.

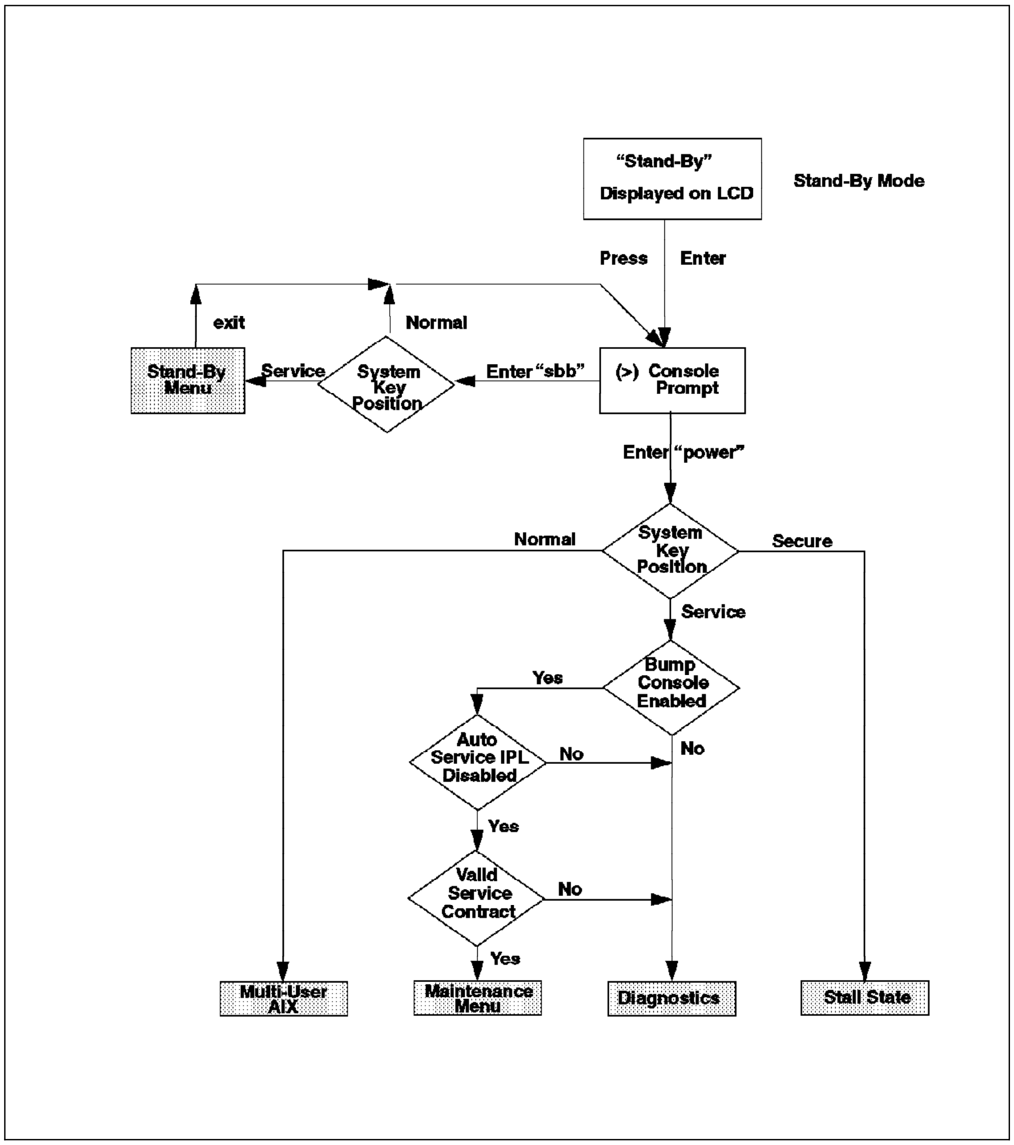


Figure 67. SystemGuard Flowchart

Basically, if the system is in Stand-By mode and the System Key (Physical or Electronic Key) in Service mode, you will be able to access the STAND-BY MENU and work with SystemGuard.

If you power-on the system from Stand-By mode with the System Key in Normal position, the system will boot to AIX Multi-User.

If you power-on the system with the System Key in Service position, you can go to the MAINTENANCE MENU or to Diagnostics, depending on the state of three flags: BUMP Console Present, Autoservice IPL, and Service Contract Validity.

If you power-on the system with the System Key in Secure, the system will stall.

Here is some information on the meaning of the different flags:

- BUMP Console Present: When the BUMP Console is enabled, the LED codes and BUMP messages are displayed on the console during the Init phase. If the BUMP Console is not enabled, it is like a regular IBM RISC System/6000; no codes and no messages are displayed on the console during the Init phase. Only AIX messages appear when the system starts loading AIX.

Note that if you are running the level 5 of the SystemGuard firmware, the BUMP Console is disabled by default, and if you enable it, it will be disabled after every shutdown. If your system is in Service mode, you might go to Diagnostics instead of Maintenance due the BUMP Console being disabled by default.

If you are running level 7 of the SystemGuard firmware, the BUMP Console is enabled by default and stays enabled after a shutdown.

- The Autoservice IPL flag, if enabled, means that you want to go to Diagnostics when booting with the System Key in Service mode.
- The Service Contract flag should be normally set when there is a contract between the customer and IBM. When this flag is valid, the IBM Service Center can access the system and do some maintenance. If there is no Valid Service Contract, it is not possible to enter the MAINTENANCE MENU or enable console mirroring. The Service Contract is set by default to an unlimited number of days (exactly 32767 days). The Service Contract should always be valid. On a pre-GA system, you might encounter a -1 value, which means that the service contract is not valid, or a 0 value, which means that the contract is valid for only one day (24 hours).

5.11 SystemGuard Menus

Because little memory is available to store the SystemGuard firmware, SystemGuard menus are low-level menus; they make extensive use of abbreviations or acronyms. You should refer to *7013 J series Operator Guide* page x-1 for a full list of these acronyms and abbreviations.

SystemGuard is menu-driven, and menu choices are usually numbered. Letters are sometimes used and can be entered in either lowercase or uppercase (SystemGuard is case insensitive). The letter *x* is often used to exit the current menu and return to the main menu (or leave SystemGuard, if given from the main menu). Commands are only treated after you press the Enter key. Until you press Enter, you can use the Backspace key to edit a command. If you enter a command that does not match the available options, 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 mode (the word Stand-By must be displayed on the LCD display). Make sure that the system is plugged into an electrical outlet, and the main power switch in the rear of the system is set to on.

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 into the STAND-BY MENU from here, you need to set the System Key into Service mode, either by turning the Physical Key to Service or the Electronic Key to Service.

To turn the Electronic Key to Service, follow the following steps:

1. Get the prompt by pressing Enter on the BUMP Console.
2. Press Enter again to put the cursor on top of the prompt.
3. At this point, press the Escape key once, and then press the s key. This puts the Electronic Key into the Service position even though the Physical Key is in Normal position.
4. Press Enter again.
5. Enter the keyword `sbb` to display the STAND-BY MENU.

The STAND-BY MENU appears with several options, as shown below:

```
  STAND-BY MENU :  rev 16.00

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.

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 two for the R30 machine.
- *7013 J Series Operator Guide,* SA23-2724, chapter three for the J30 machine.
- *7012 G Series Operator Guide,* SA23-2740, Appendix B for the G30 machine.

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 the Fast IPL flag to skip the second phase of PON tests and the BUMP Console Present flag to enable or disable the BUMP Console during bootup. This menu allows also the Customer Engineer to test the interconnection between the BUMP and the different components through the I2C bus or the SSbus. For instance, it is possible to send a specific string of characters to the LCD and read the result on the Operator Panel display. Or, it is possible to turn on the LED on the Operator Panel or to turn the power supplies and fans on without letting the system IPL.

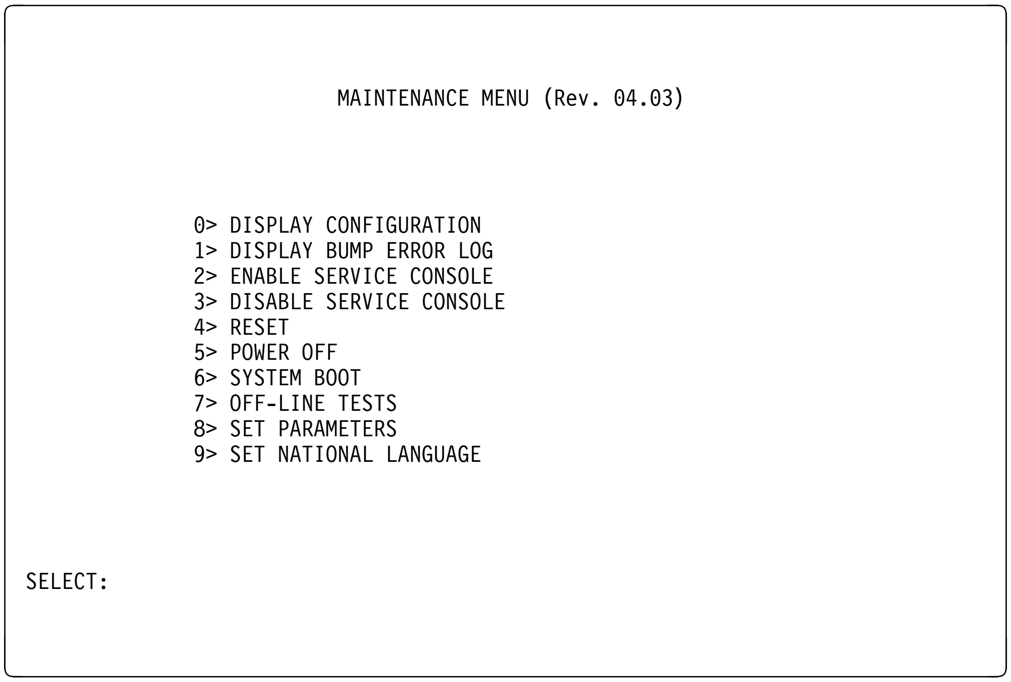
5.11.2 Maintenance Menu

The MAINTENANCE MENU also enables you to display the configuration of the system in a non-cryptic, easily understandable way, to perform various tests, to continue IPL either from network, a specific SCSI device or from the boot list, and to set flags concerning various system operation.

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 case for current systems shipped).
4. Turning the System Key to the Service position.
5. Powering-on the system.

The MAINTENANCE MENU shown below should appear just after the 292 code is displayed on both the console and the LCD.



```
                        MAINTENANCE MENU (Rev. 04.03)



               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 4.1.2 for visualizing and changing SystemGuard parameters. These commands allow the system administrator to carry out various tasks, thereby eliminating the need for system reboot or operator intervention. This is ideal for remote support. It also allows you to perform tasks, such as enabling and disabling processors.

Let us look at these commands and examples of their output:

- `cpu_state {-l | -d Number | -e Number}`

  This command allows you to list the processors status, 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` is the processor number. Note that disabling and enabling processors only takes effect after 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 -w} <index> <value>` for changing flags

  `mpcfg { -r | -s }` for restoring or saving flags in the 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 some 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.

  `-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.

  `-w` Indicates that the change will be applied to a password.

- `keycfg -d` for displaying the status of the System, Physical and Electronic Keys, respectively. These keys are also called the Mode Switch, Key Mode Switch and the Electronic Mode Switch.

  `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: There are two things you should be aware of:

- The Physical Key overrides the Electronic Key; so the Physical Key must be in Normal if you want to be able to set the Electronic Key.
- In AIX V4.1.2, the output of the command only works if LANG is set to C. So, before running the command, do an `export LANG=C`.

- `mirrord`

  This daemon is used to implement the mirroring function between the BUMP Console on S1 and the Service Console on S2. This daemon is automatically started at the boot time if the Service Contract is valid. It is just sleeping until the mirroring is activated.

- `survd`

  This daemon implements a heartbeat protocol between SystemGuard and AIX. If SystemGuard has not received a message from AIX in a specified delay time, SystemGuard assumes that AIX is hung and reboots the system. The survd daemon is not started automatically; user root has the ability to start or stop it.

  ```
  survd -d <number of seconds>
  ```

  This sets the heartbeat delay time. The default delay time value 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 running processes, `survd` might not even get to the run queue; therefore, SystemGuard activates a reboot.

  ```
  survd -h
  ```

  The flag `-h` issues a hardware reboot instead of a software reboot.

  `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 does not get any messages from the daemon and assumes that AIX hung.

5.13 Processor and Memory Failure

As mentioned before, SystemGuard monitors the hardware and software, and in case of failure, it 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.

If a memory failure is unrecoverable, then a checkstop also occurs. AIX error handler will log and report the failure:

- A dump is taken on 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 a transient failure, the system will come back up. AIX copies logout data from the NVRAM to a file and logs the event in the error log file.
2. If the PON tests fail because of a solid hardware failure, SystemGuard will deconfigure the failed processor or the memory block and restart or reboot the system.
3. If the boot fails, even in the reduced hardware configuration, the offline Maintenance mode is entered, and the problem is reported via the code on the Operator Panel display. In this case, if the dial-out function is set, the system will dial-out to a service center.

5.14 Some Common SystemGuard Tasks

The following tasks will be done through the STAND-BY and MAINTENANCE MENUs that are part of SystemGuard. Note that these tasks can also be carried out from AIX diagnostics.

5.14.1 How to Set the Electronic Key

The key can be set electronically, making it easier to provide remote support without physically touching the machine. You can do this from the Stand-By mode or from AIX.

5.14.1.1 Setting the Electronic Key from Stand-By Mode

1. Go into Stand-By mode.
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. At this point, press the Escape 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; this is a way to check that the System Key is in Service.
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 n key. This puts the Electronic Key to Normal position.

5.14.1.2 Setting the Electronic Key from AIX

1. While AIX is running, log in as user *root*.
2. Type the following command to look at the current status of the Electronic Key:

   ```
   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
   ```

Note that, each time, you can run the `keycfg -d` command to verify the status of the key.

5.14.2 How to Display the System Configuration

The system configuration can be displayed through the STAND-BY MENU or through the MAINTENANCE MENU.

5.14.2.1 Displaying Configuration through 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 not configured. This is done by pressing the reset button with the mode switch in the Service position.

To display the configuration of the system, enter the STAND-BY MENU, and from the Main menu, 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         CCAACCAAAAAAAA
FLASH_FW 0704   MM size  0080               OP_KEY  NRM     E_KEY   SRV
OPP   D78610    19H0494
SP    D78605    19H0471                     IOC   E38030    96G4400
CPU0  D78605    19H6472                     CPU1  D78605    19H6472
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 seen from this panel.

First, you can see in the `SID TM` field the type and model of the machine, 7013-J30, in our example.

The SID Y2 field contains a number which is used for building the uname of the machine. For machines built in Austin, Texas, this number is the serial number of the machine without the two first digits. The first two digits correspond to the plant number (26 for Austin plant).

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 extension 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. You can see in our example the system has four processors that are configured.

The MM conf field shows the memory configuration. You can see that this field has sixteen digits. Each digit gives the status of each memory bank. Refer to Chapter 3, “SMP Servers Architecture” on page 49 in this book to get more information on what constitutes a memory bank. There are sixteen digits corresponding to the maximum number of banks that you can have in an SMP system. As for 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 on the system, and then deduct 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 in the system, 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 the SystemGuard firmware stored in the Flash EPROM.

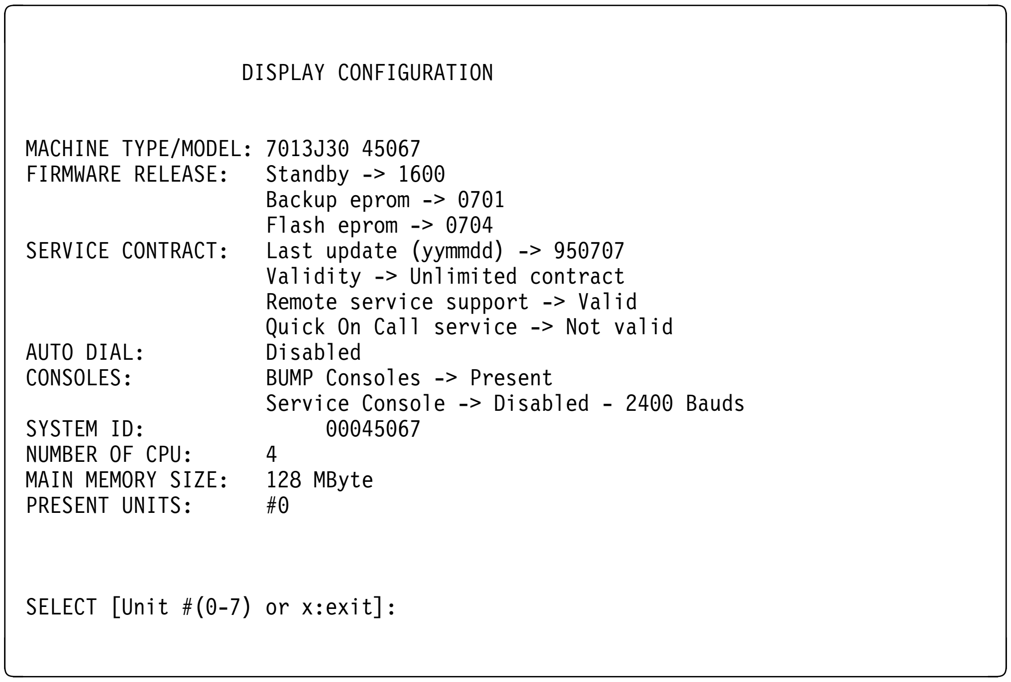
5.14.2.2 Displaying Configuration through the Maintenance Menu

The system configuration can also be displayed through the MAINTENANCE MENU. You will find the same kind of information displayed previously but in a different and more readable way.

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 -> 1600
                    Backup eprom -> 0701
                    Flash eprom -> 0704
SERVICE CONTRACT:   Last update (yymmdd) -> 950707
                    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. The interesting feature here is that you can see the level of the firmware stored in the BUMP EPROM, the level of the firmware stored in the Flash EPROM and the level stored in the Backup EPROM.

5.14.3 How to Set Fast IPL

If the Fast IPL flag is enabled, SystemGuard will skip the POST. By default, the Fast IPL flag is disabled; enabling it will only last one reboot.

There are three ways to enable it: through the STAND-BY MENU in Stand-By mode, through the MAINTENANCE MENU or by using AIX commands.

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. Enter 6 to set the Fast IPL flag. You will then get the current status of the flag and be prompted to change it. If it is disabled, enter y (yes) to enable it.

The Fast IPL flag is now enabled; the POST will not be run when the system boots. This will save several minutes.

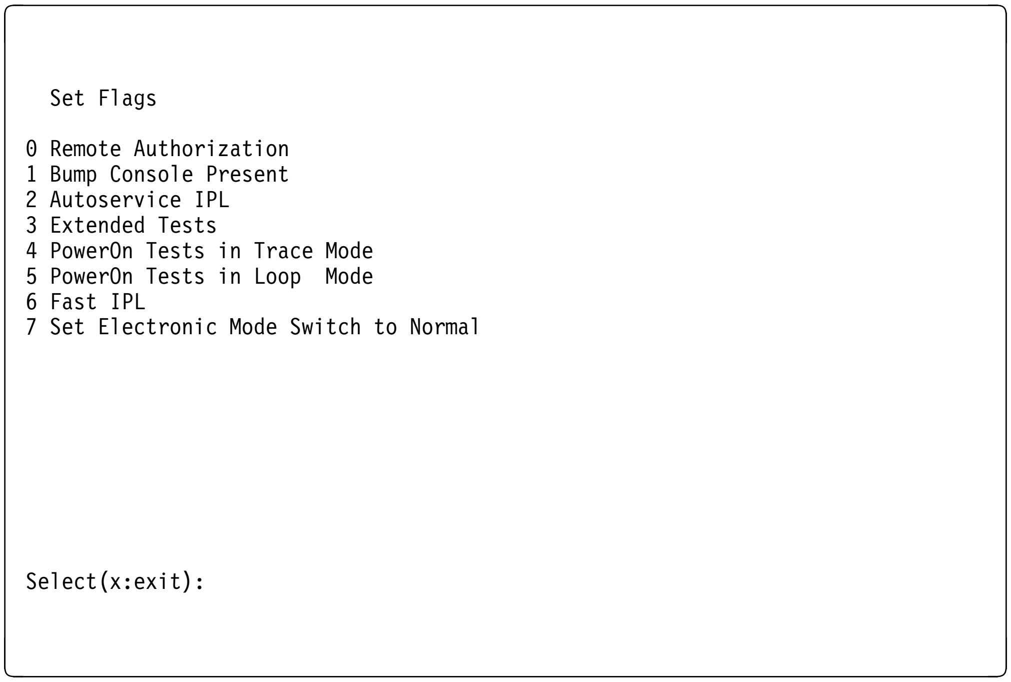

```
  Set Flags

0 Remote Authorization
1 Bump Console Present
2 Autoservice IPL
3 Extended Tests
4 PowerOn Tests in Trace Mode
5 PowerOn Tests in Loop  Mode
6 Fast IPL
7 Set Electronic Mode Switch to Normal








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, generally, the current 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 :

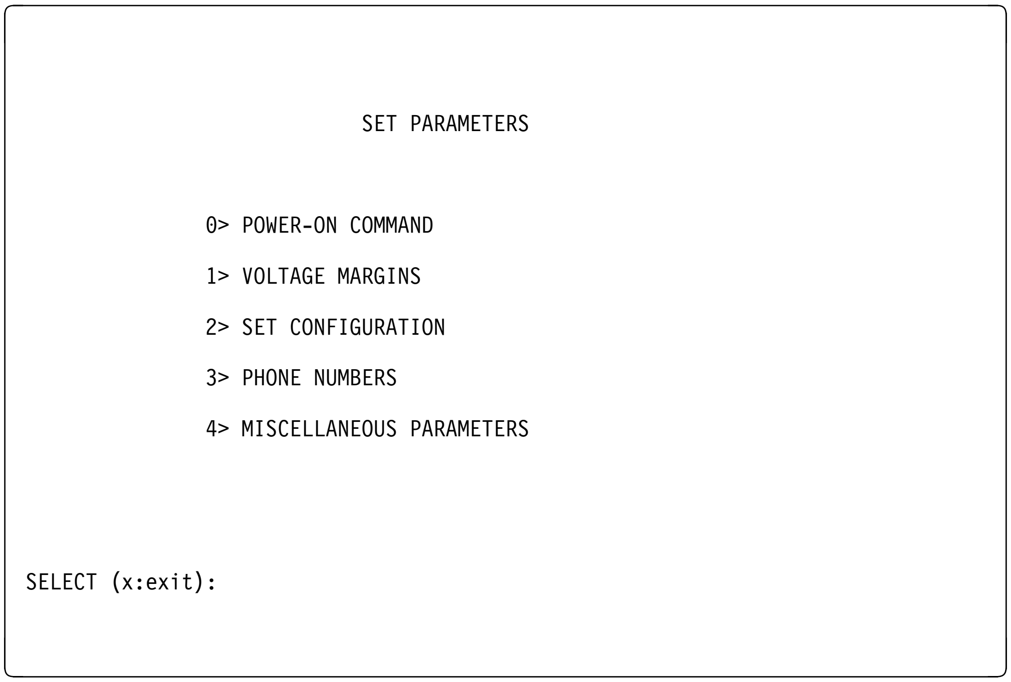
   ```
   mpcfg -df
   ```

5.14.4 How to Set the Service Line Speed

By default, the service line speed is 1200 bauds or 2400 bauds, depending on the level of SystemGuard. This speed can be changed 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 tty1 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 up and running, log in as user *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 log in to 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. Enter 0 from the SET FLAGS menu to change the Remote Authorization flag.
4. If it is set to disabled, select **y** (yes) to change it to enabled.
5. 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 user `root`.
2. Type the following command to view the current setting:

   ```
   mpcfg -df
   ```

3. Type the following command to change the flag:

   ```
   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

Let us first introduce console mirroring.

5.14.6.1 Console Mirroring Concepts

Console mirroring is a way to provide the customer a view of what the person working remotely from the Service Console 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 the mirroring starts).

Mirroring only works on the two 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 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 can also be supported.

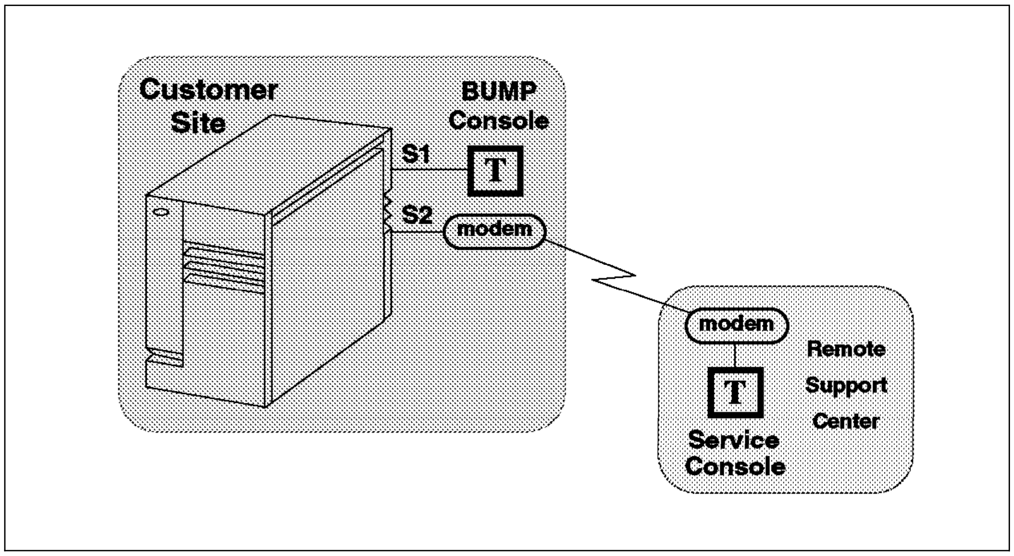


Figure 75. Console Connection

When mirroring is active, the customer on the BUMP Console and the support personnel on the Service Console both see the same output on their screens, and either one may enter characters. One person can even start typing and the other finish it. For example, the support personnel may log in root on the Service Console, and the customer may enter the root password on the BUMP Console. Therefore, the remote support personnel will not need to know the root password.

Console mirroring is possible during the following SystemGuard phases: Stand-By, Init and Run-Time (AIX running).

The prerequisites for console mirroring are:

- Remote Service Support 1
- Service Contract Validity 0 - 32767
- /usr/share/modems/mir_modem file present (for mirroring when AIX is up)

5.14.6.2 Setting Up Console Mirroring

In order to set up console mirroring, you need first to authorize the Service Console, and set up the right line speed. Refer to previous chapters 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 user `root`.
2. Type the following command to make sure that 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 the mirror daemon (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. Now, the support personnel should be able to work on the remote console, and the customer should be able to watch on the BUMP Console what the service personnel enter on the Service 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 set 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 file that specifies the type and characteristics of the modem. This modem file is required even if the Service Console is connected locally without modem. Thus, a file with no modem must be provided. The default name of the modem file is mir_modem. Please refer to Appendix A, “SystemGuard Remote Operation Configuration” on page 237 for supported modem files.

5.14.7 How to Enable Surveillance

Surveillance is implemented by 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 the surveillance. The `number of seconds` determines the heartbeat delay time, where ten seconds is the minimum, and the default is sixty 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 hung since it no longer receives a heartbeat. Thus, use the command in step three to turn off SystemGuard.

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 is the automatic sending of certain errors to a 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 POST error or boot device not found, a problem report is sent to the remote service center.

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 message that gets sent to the Service Console 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 Setting Up Dial-Out from AIX

Problem reporting can also be implemented at the operating-system level. Activating mirroring in case of a specific problem on the system is a way to report the problem to the service support.

If errnotify is set up to set the Electronic Key to Service whenever there is an error matching the selected criteria in errnotify (object class in the Object Data Manager), the key in Service would start the mirrord daemon and console mirroring. Once console mirroring is running, the script called by the errnotify object can send a message to the S2 t send mail to the remote service personnel or run a customized script.

Here is an example of a file, /tmp/sample.add, which is going to be used for reporting tape drive errors to the Service Console.

```
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 would 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
```

where, console mirroring is set up; the message is echoed to the BUMP Console which is displayed on the Service Console (due to console mirroring); the error log entry for the tape drive is added 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 for the remote personnel connected via the Service Console to reboot AIX from the remote site.

5.14.9.1 Prerequisites

The following procedure must be carried out from the BUMP Console in order to allow AIX to boot remotely from the Service Console.

1. Get into the MAINTENANCE MENU (reboot the system with the key in Service mode).
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 user *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 S2 port or the remote console can activate a reboot. The system can be rebooted either in AIX Multi-User or in Diagnostics to run diagnostics on the hardware.

5.14.9.2 Rebooting to AIX Multi-User

1. While AIX is running, log in as user *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 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.
9. You should get the AIX login prompt in about ten minutes.

5.14.9.3 Rebooting to Single-User and then to Multi-User

This allows the remote support personnel connected to the system via a modem to the S2 port to shut down and reboot the system in Diagnostics mode for hardware diagnostics purposes. After running diagnostics, the remote personnel can reboot the system in AIX Multi-User without having a need to physically touch the machine.

1. While AIX is running, type `shutdown` to shut down the system.
2. Once the system is shutdown, you should see the Stand-By mode 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 S2 that we had set earlier (`power` in our example).
4. The system will reboot in Diagnostics. From here, diagnostic tasks can be run.
5. Once completed, activate 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 reboot in AIX Multi-User:

   ```
   init 2
   ```

8. After about ten minutes, the system should have rebooted and loaded AIX.
9. If S2 was configured as a tty from AIX as well, an AIX login screen should appear on this 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 booted with the system key in Service position, it either boots in the MAINTENANCE MENU, in Diagnostics or from an SCSI device, depending upon the various 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; refer to the MAINTENANCE MENU section in this redbook for details.
2. From the MAINTENANCE MENU, enter 8 to select the **SET PARAMETERS** menu.
3. From this menu, enter 4. 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 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 off the SCSI device, off a tape drive for example.
6. Insert the bootable tape in the tape drive.
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 to boot from, 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, but we recommend that you go through all the previous options.

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 132, 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 an 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**).
- A supported SCSI device, such as a tape drive.
- A bootable SCSI media, such as a bootable tape (install or mksysb).

To boot off 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 media 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 reinstalled 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

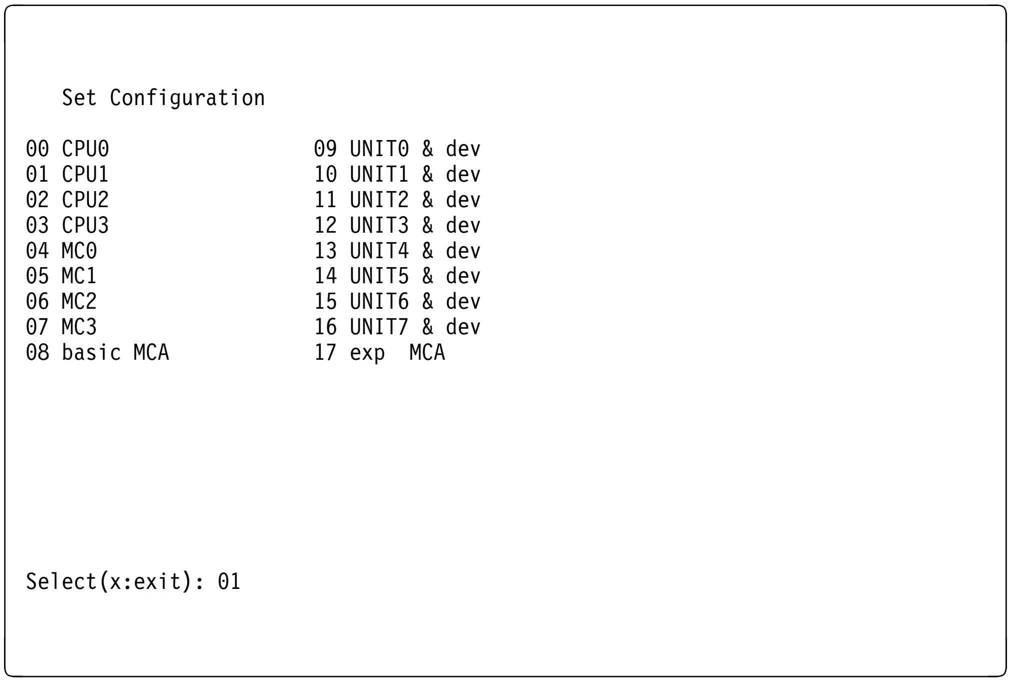
To find out more information on NIM (Network Install Manager), refer to *AIX Version 4.1 Network Installation Management Guide and Reference,* SC23-2627.

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 redbook.
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 processor 1.
7. To enable CPU1, follow the same procedure as above, except choose a status **C**,(`03` in above example).
8. Exit the STAND-BY MENU, and continue booting the machine.

5.14.12.2 Enabling/Disabling Processors through the Maintenance Menu

1. Enter the MAINTENANCE MENU. Refer to the MAINTENANCE MENU section of this redbook.
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, CPU1 for example, type the following command: `cpu_state -d 1`

  Look at the result with `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
```

Chapter 6. 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. It has a feature code FC 6175.

It allows a single console to be used for several CPUs. The console can be an ASCII terminal or a PC running DOS. With the CPC control program, a DOS-based terminal emulator is provided so that you can 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 RISC System/6000 models and peripherals:

- 7012-G30
- 7013-J30
- 7015-R10, R20, R21, R24, R30, 950, 970, 990
- 7013-570, 57F, 580, 58H, 590, 59H, 591 (Note)
- Disk units: 9333, 9334, 7134, 7135

Note: The power-on feature is not available on the 5XX deskside models. If the 5XX systems has been shutdown, someone must be physically present to turn the power back on.

6.1 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

6.1.1 CPC Connectors

The following figure shows the connectors and features that are available on the front panel of the Cluster Power Controller.

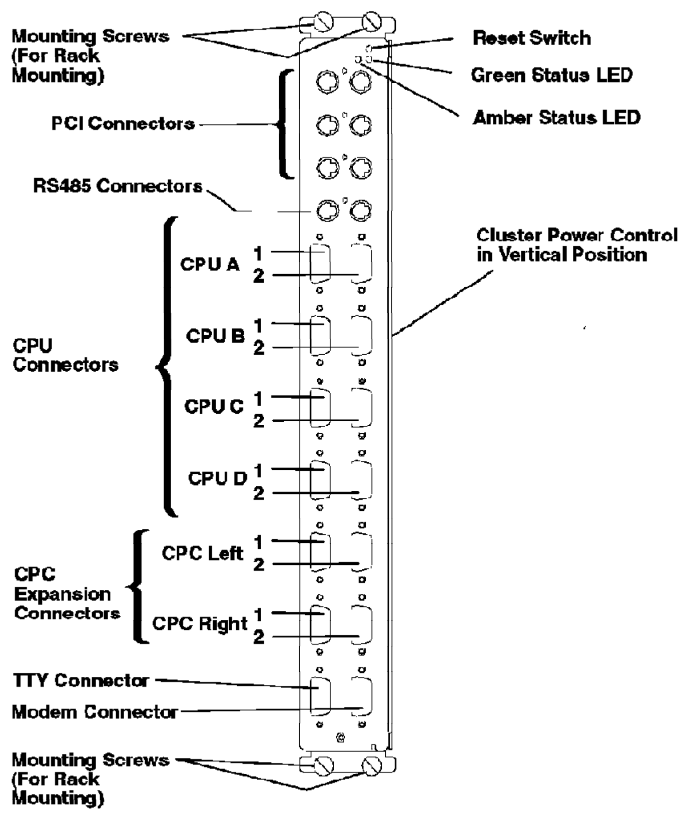


Figure 91. CPC Front Panel

- *Reset Button*:

  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 the attached system unit and disk drawers is not affected when recycling 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.

- *Disk Drive Connectors (RS485)*:

  These connectors are reserved for future use. The ports are marked 6-1 and 6-2. The J01 expansion cabinets will be connected to and controlled by the J30. However, the J30 can be powered on from the CPC.

- *CPU Connectors*:

  These connectors are used to connect the S1 port of the system units to the TTY device or console and to connect the S2 port of the system units to the device attached to the modem port.

- *CPC Expansion Connectors*:

  These connectors are used for connecting multiple CPCs in a daisy chain fashion.

- *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 for the Service Console.

Please refer to the 7015 Model R00 Rack Installation and Service Guide, page 1-8 or the Cluster Power Control Operator and Service Guide, page 1-2 for more information.

6.1.2 CPC Port Connections

Following is a conceptual diagram of a typical CPC-to-CPU connection. Here, we have two G30 servers and one J30 server connected to one BUMP and yp one remote console through the CPC.

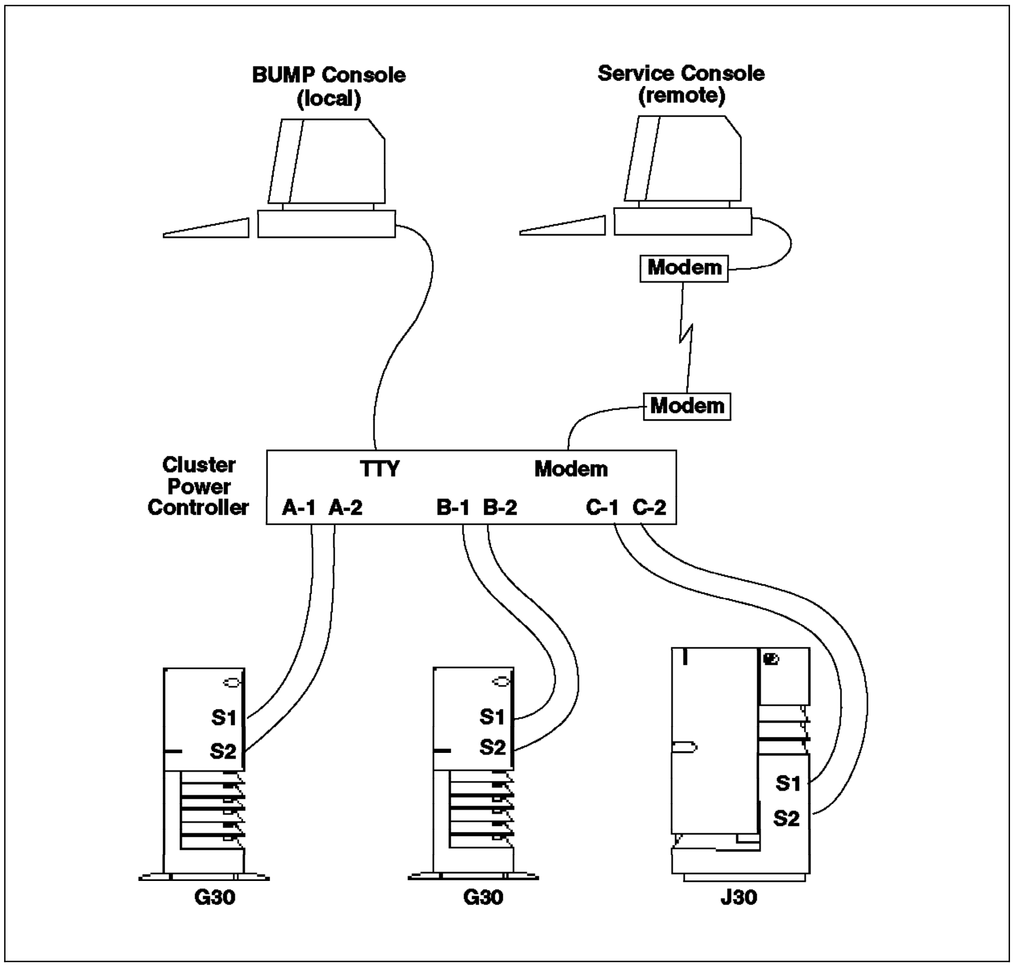


Figure 92. Typical CPC-to-CPU Connection

S1 and S2 of the first G30 are connected to CPU-A Port 1 and Port 2, respectively. S1 and S2 of the second G30 are connected to CPU-B Port 1 and Port 2, respectively. S1 and S2 of the J30 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.

Table 6. CPC Port Connections

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

Note:

A	Available or open connection
B	Power control for UP (excluding 5XX) or Disk Subsystems
K	Serial port cables
SMP	R30, J30 or G30 (SMP) system units
UP	570, 580, 58H, 590, 59H, 591, R10, R20, R21, R24 (UP) system units
933x	9333, 9334 disk subsystems

The maximum number of devices that can be attached to a single CPC using a combination of device types are:

- 4 x SMP + 6 x 933x
- 2 x UP + 2 x SMP + 4 x 933x
- 4 x UP + 2 x 933x

Other combinations are possible, but you get an idea of the mixture.

6.1.3 CPC Cables

The following diagram illustrates the cables required to connect a CPC to a single CPU (G30, J30 or 5XX) and a single disk unit.

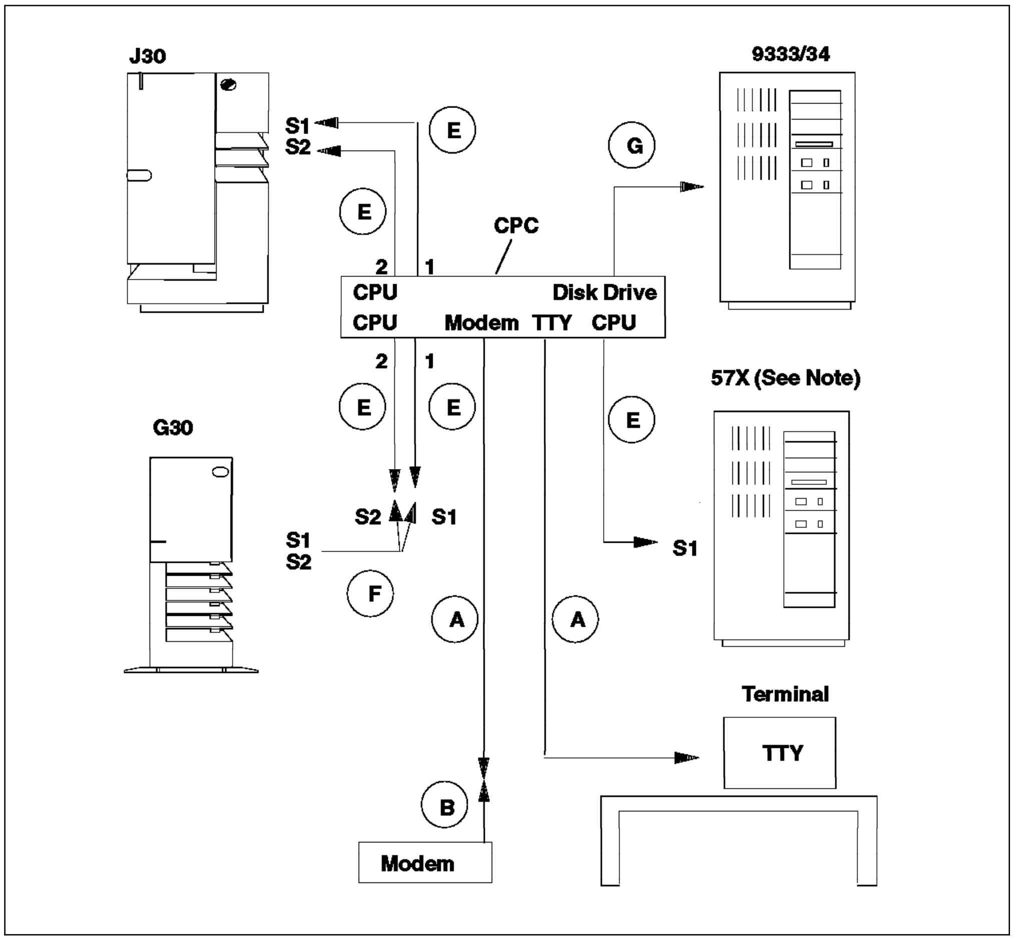


Figure 93. Single CPU Connected to CPC

The following table gives a description of the cables that can be used to connect the CPC to the various devices.

Table 7. CPC Connection Cables

Cable	IBM P/N	Description
A	6298963, FC 6176	CPC(9F) - TTY(25M) Null modem cable, 10ft
B	58F2861	Null modem adaptor or terminal/printer interposer (25F) - (25M)
C (Note 1)	6450242(kit)	R30(9F) - CPC(25M) AT serial adaptor connector cable, 10in
D	11H7336, FC 6177	CPC(9F) - CPC(9F) Null modem CPC-CPC interconnect cable, 25ft
E	11H7337, FC 6178	CPC(9F) - System(25F) Null modem serial port cable, 10ft
F (Note 2)	11H3834	S1/S2 G30(25F) - S1(25M) and S2(25M) 1 to 2 Y-cable, 1 ft
G (Note 3)	42F6839	CPC(4P) - 933x(4P) Power control interface cable, 10ft
K	12H1605, FC 6180	CPC(4P) - R10, R20, R24 (5P) Power control interface cable, 10ft

Note:

1. Supplied with R30 to convert 9F SIB ports to 25M
2. Supplied with G30 to convert S1/S2 25M SIB port to 2 x 25M
3. Supplied with peripherals

Legend

9F	9-pin female connector
25F	25-pin female connector
25M	25-pin male connector
4P	4-pin PCI connector
5P	5-pin PCI connector

6.1.4 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. (Optional) Connect the serial port 2 (S2) of the UP system unit to the corresponding CPU (A-D) Port 2. This allows 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 systems units, group the shared disks as a separate system so that the power can be controlled separate to the system units.
- The system unit key switch should be in the Normal position for remote power-on.

6.2 CPC Installation

CPC can be connected to the three servers G30, J30 and the R30 in different ways. For example, one CPC to one CPU, one CPC to multiple CPUs or multiple CPCs to multiple CPUs. CPC also allows connection of disk drawers or deskside units. In the desktop (G30) and the deskside (J30) 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 rested on top of the G30 and J30.

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.

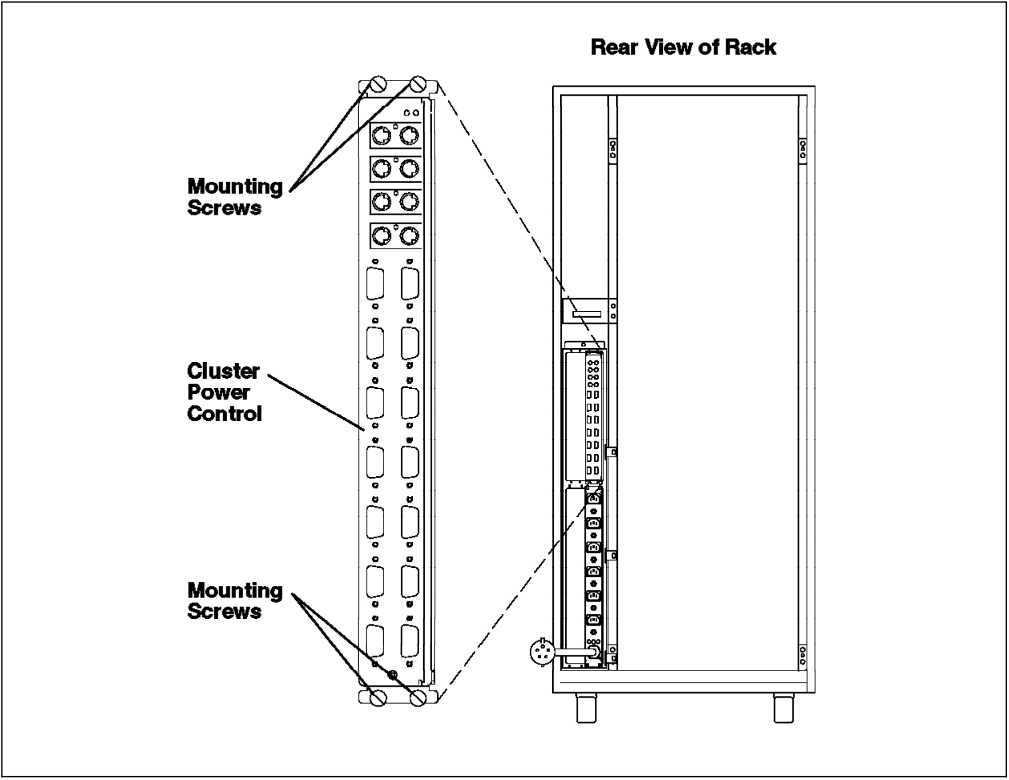


Figure 94. Rack-Mounted CPC

6.2.1 Prerequisites

Following are the prerequisites for CPC installation:

1. If installing the CPC (FC 6175) in a 7015/R00 rack, verify at least one of the following features is installed in the rack: (FC 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. Must have the microcode installation diskette and instructions. Available in FBM 11H7663.
4. AIX V3.2.5 or later of operating system (AIX V4.1.2 for SMP).

6.2.2 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. (Optional) Attach a modem to the modem port.
5. Power-on the CPC by plugging in the power cord.
6. Customize the CPC; that is, configure it and save the configuration.

6.2.3 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). Following is the description of the light status progress:

1. First ten seconds, the POST runs and the lights stay Off.
2. After twenty 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 twenty seconds, both the lights start flashing, it means:
   - 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 6.4.7, "Daisy Chaining CPCs" on page 166 for more information.
4. If, after twenty seconds, the tests fail, only the Amber LED will stay On. The Green LED will be Off. This indicates the CPC is defective; replace it.

Use the following table as a guide:

Table 8. Cluster Power Control Light Status Indicator

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

6.3 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 95. CPC Main Menu

The following figure shows the various CPC Menu options and the paths required to access these options:

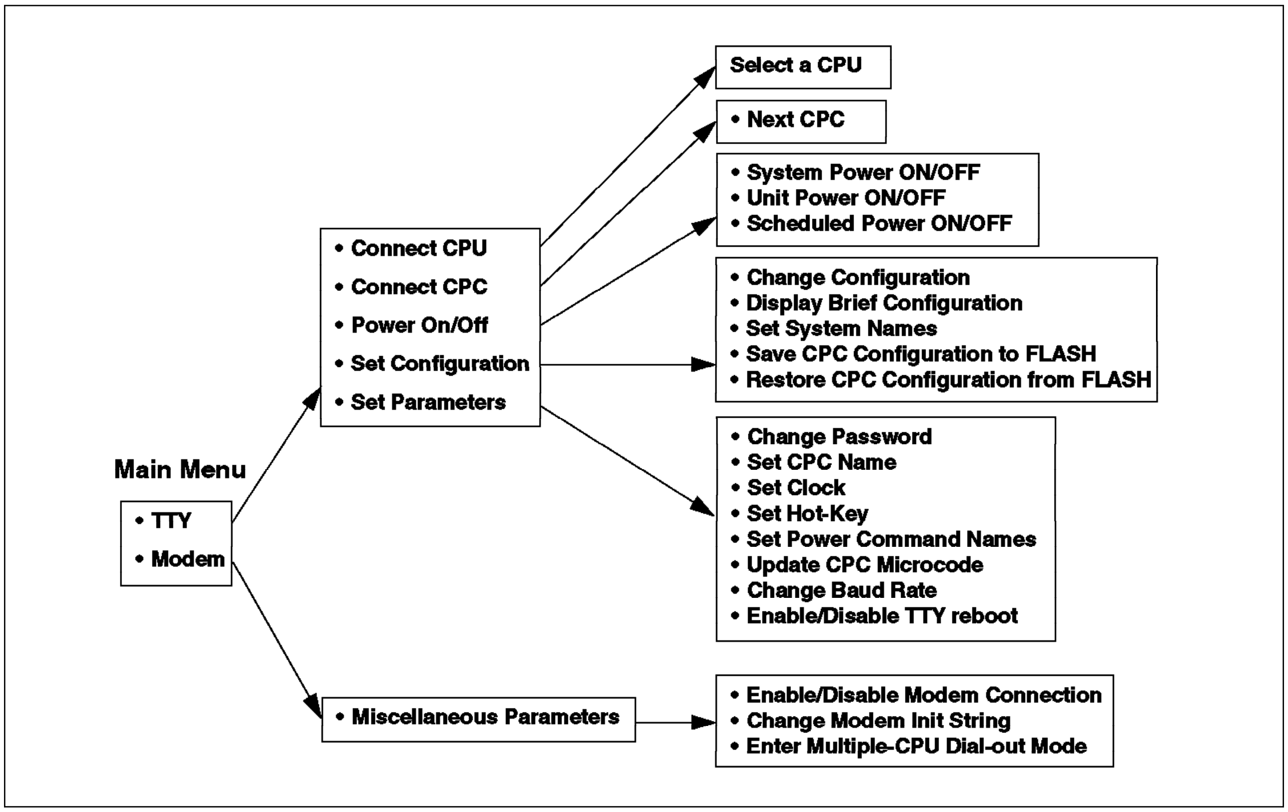


Figure 96. CPC Menu Options

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

6.3.1 Configure the CPC

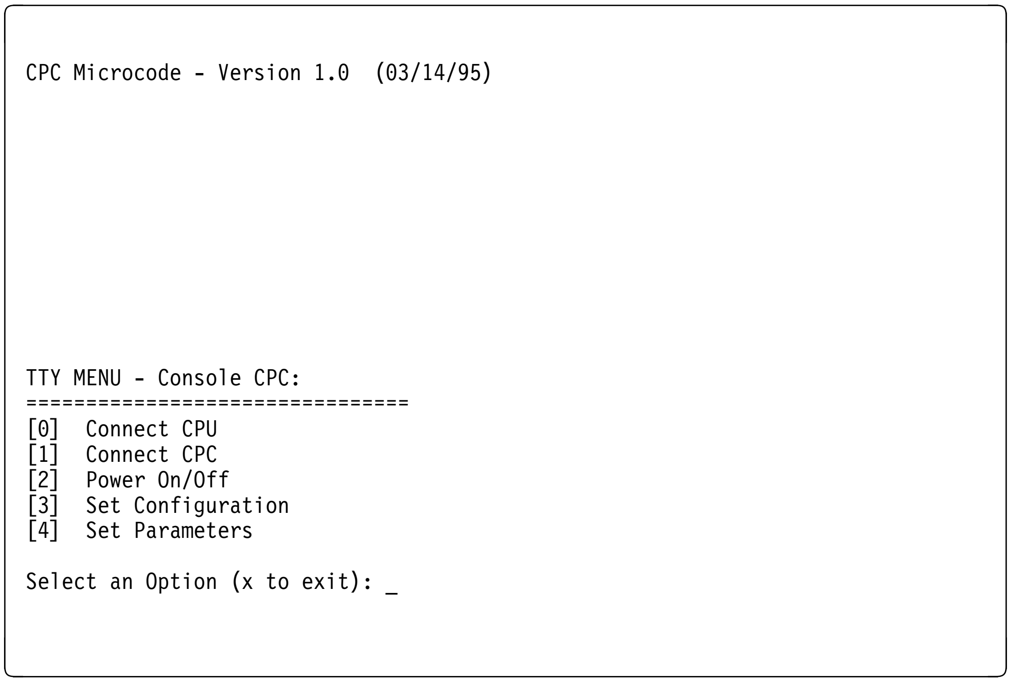
The following operations are performed to configure the CPC:

- 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:

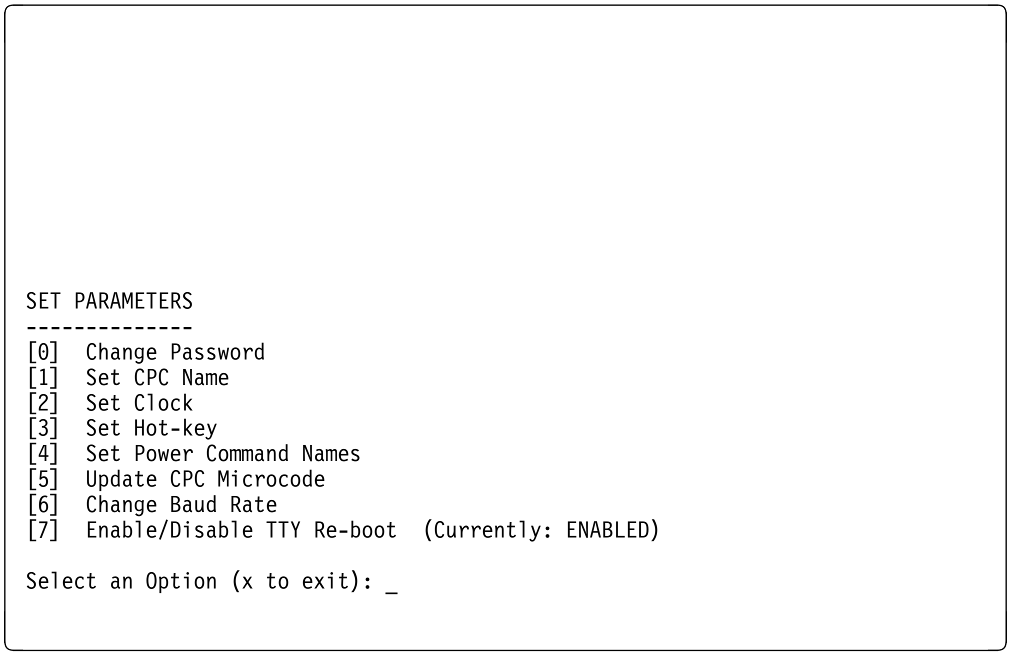


```
CPC Microcode - Version 1.0  (03/14/95)










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 97. 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 98. SET PARAMETERS Menu

3. Enter 1 to select **Set CPC Name**. The following line appears:

```
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.
```

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. There is no password set on the CPC when it is delivered to a customer.

   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; now we are ready to configure the system units (CPU).

6.3.2 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): _
```

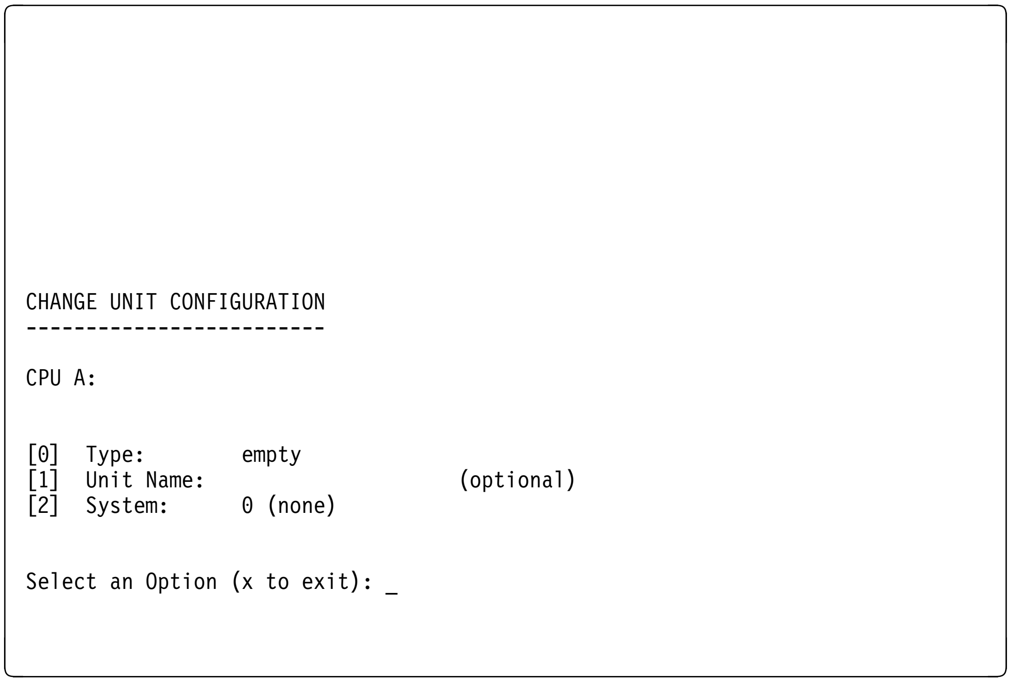
Figure 99. SET CONFIGURATION Menu

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 100. CHANGE CONFIGURATION - SELECT UNIT Menu

4. Enter `0` to select **CPU A**. The following screen will appear:



```
CHANGE UNIT CONFIGURATION
-------------------------

CPU A:


[0]  Type:        empty
[1]  Unit Name:                     (optional)
[2]  System:      0 (none)


Select an Option (x to exit): _
```

Figure 101. 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.

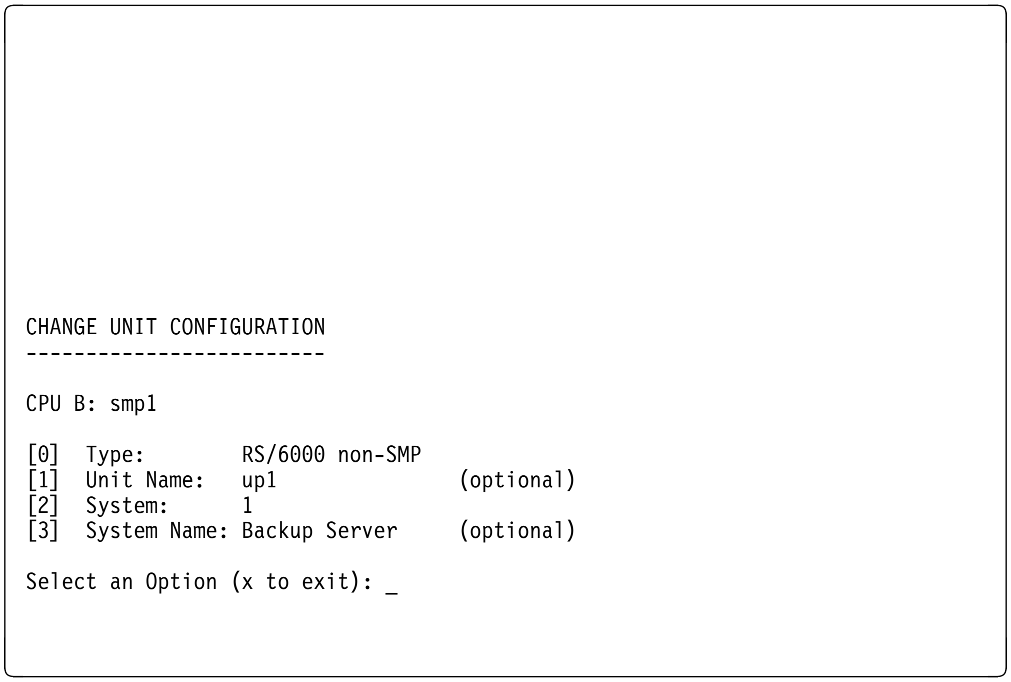
   We configured a UP system on the CPU A ports so the CPU Type was set to `R`.

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. (Optional) Enter `2`, enter the System Number and then 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` for the System Number.

8. (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 for another system unit, the existing System Name will be displayed. Enter `3`; 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. For this example, we used `Backup Server` for the System Name.

   Here is an example of the screen output after the configuration:



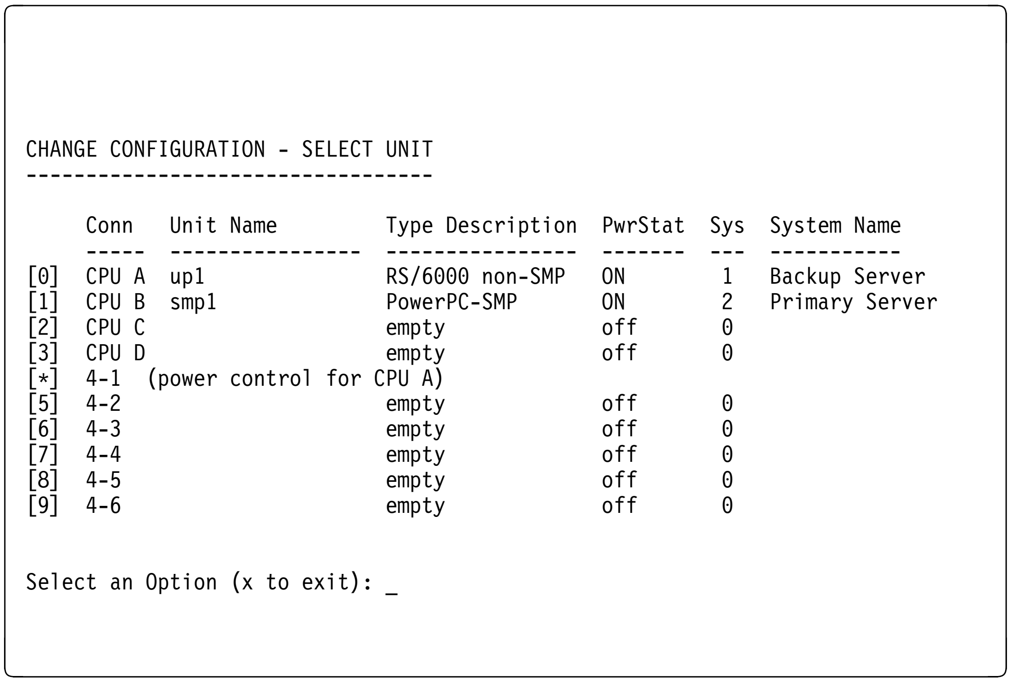
```
CHANGE UNIT CONFIGURATION
-------------------------

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 102. 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 103. CHANGE CONFIGURATION - SELECT UNIT Menu

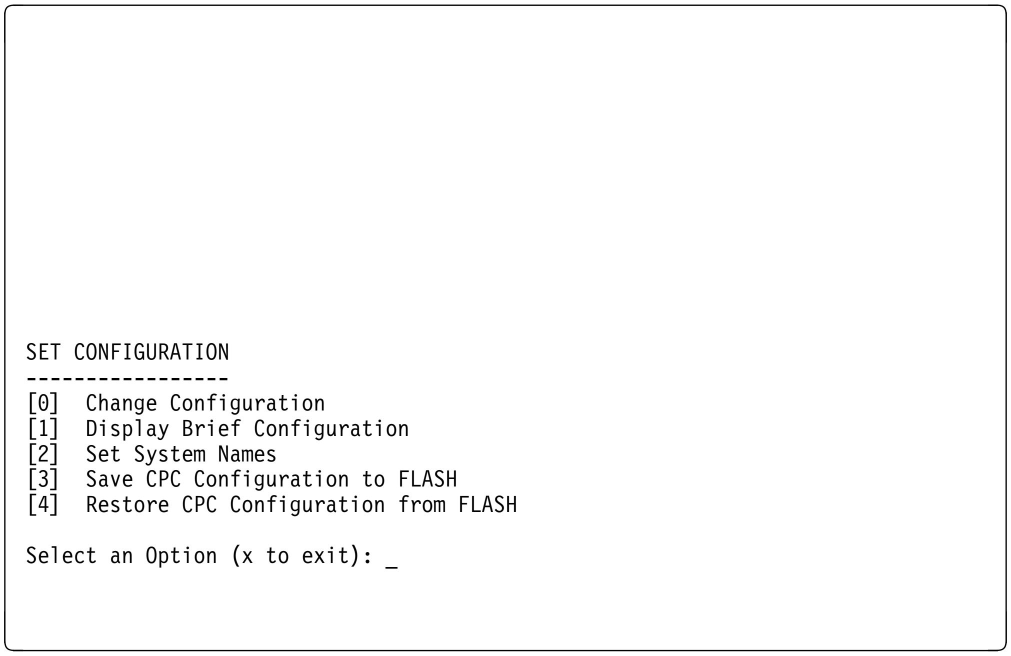
10. Enter x to exit back to the TTY menu.

6.3.3 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 104. 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 105. 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 106. CHANGE UNIT CONFIGURATION Menu

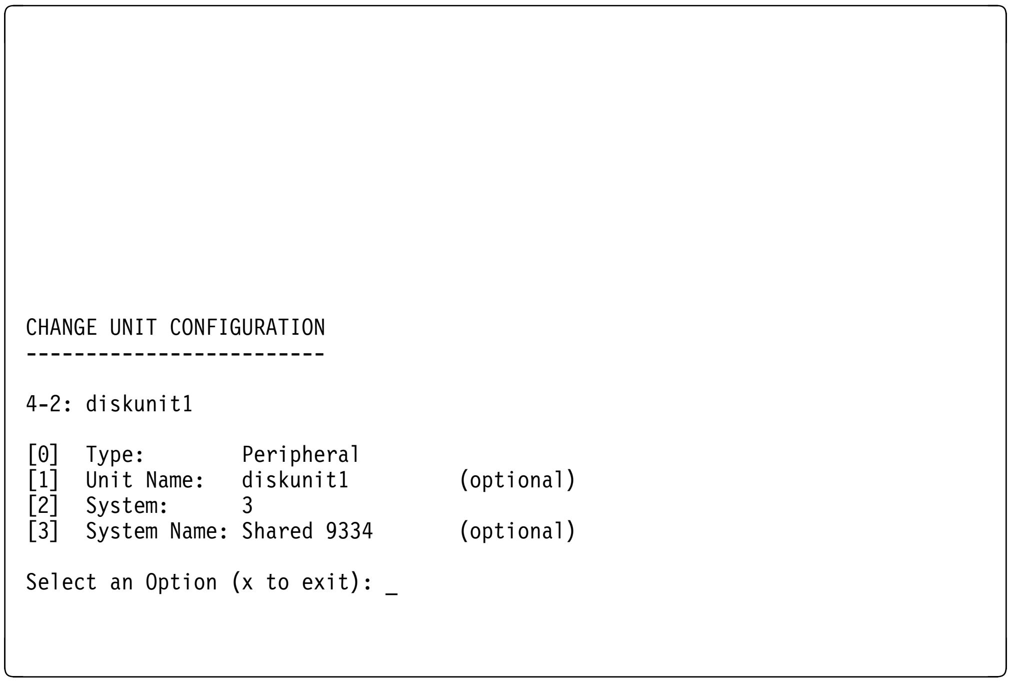
5. Enter 0 to select the **Type** option. The following line appears:

```
Set Peripheral Type (d = peripheral, . = empty):
```

Enter `d` for the peripheral type.

6. (Optional) Enter `1` to select the **Unit Name** option. The Unit Name should be something that the customer can use to uniquely identify that partcular disk subsystem. For this example, we used `diskunit1` for the Unit Name.
7. (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. 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.

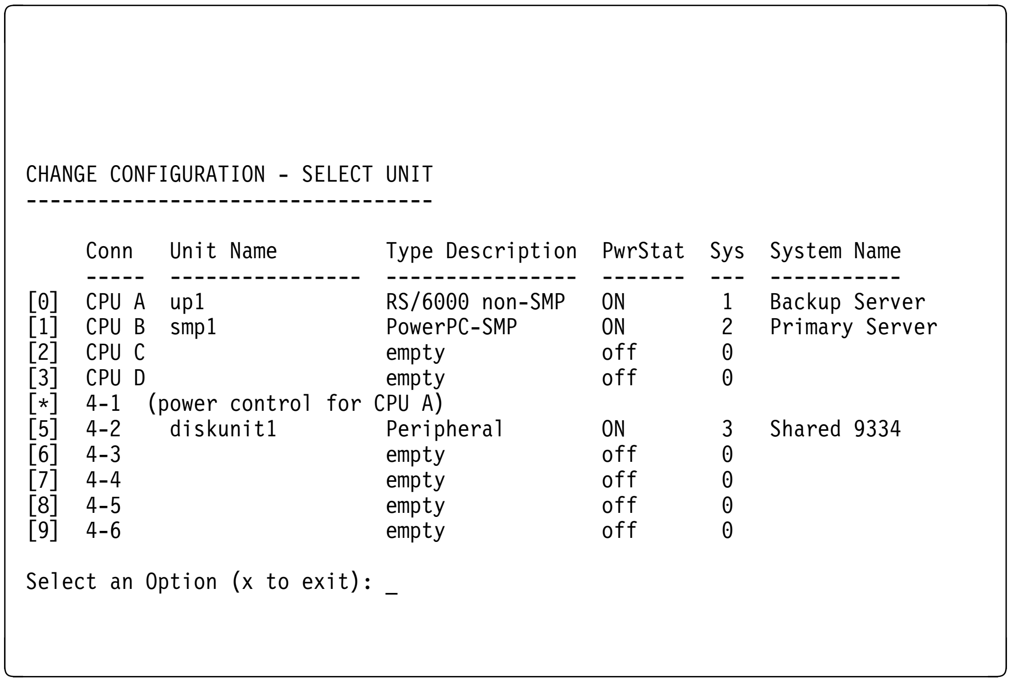
   Once the system number is entered, an additional field for the System Name appears.
8. Enter `3` to select the **System Name** option. Enter a unique system name. Once again, you need to enter something that is easily identified by the customer. For this example, we used `Shared 9334` for the System 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 107. 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 108. 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 system reboots, it will be reconfigured using the existing parameters in the flash memory.

6.3.4 Installation of Poweroff User

For an orderly shutdown of the RISC System/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 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 RISC System/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
   ```

4. Check that the poweroff user ID has been appended to the end of the /etc/passwd file.

   ```
   grep poweroff /etc/passwd
   ```

6.4 CPC Operations

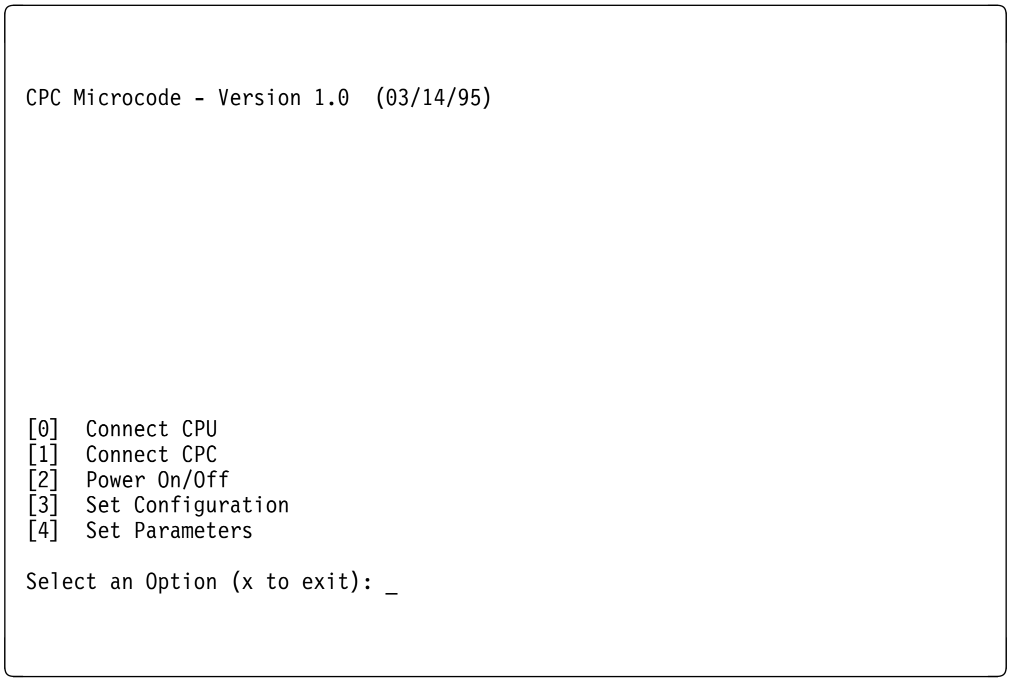
After the CPC has been installed and customized, a number of different operations are now possible.

6.4.1 How to Connect and Log Into the CPUs

Before the terminal can be connected to the CPU, make sure the cables are connected, CPC has powered up with no errors, 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 109. 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): _
```

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.
```

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.

6.4.2 How to Power-Off/On Systems From the CPC

The power off/on feature allows control of power of the systems and peripheral drawer. This feature can be used through the CPC program menus.

To power-off or power-on 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 complete, exit to previous menus by entering `x`.

6.4.3 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.

6.4.4 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.

6.4.5 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. Some terminals do not have a power-saving function, or the customer 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.

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`.

6.4.6 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 RISC System/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 RISC System/6000 and then downloads the microcode to the CPC.

1. Insert the CPC Microcode diskette into the diskette drive on the RISC System/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, using the terminal attached to the TTY port on the CPC, enter `0` to select 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 RISC System/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 hot-key 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 Downloaded Microcode Image. Do not power-off the CPC until it has completed the update.
11. Enter `4` to 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.

6.4.7 Daisy Chaining CPCs

The following diagram illustrates three CPCs in a daisy chained configuration.

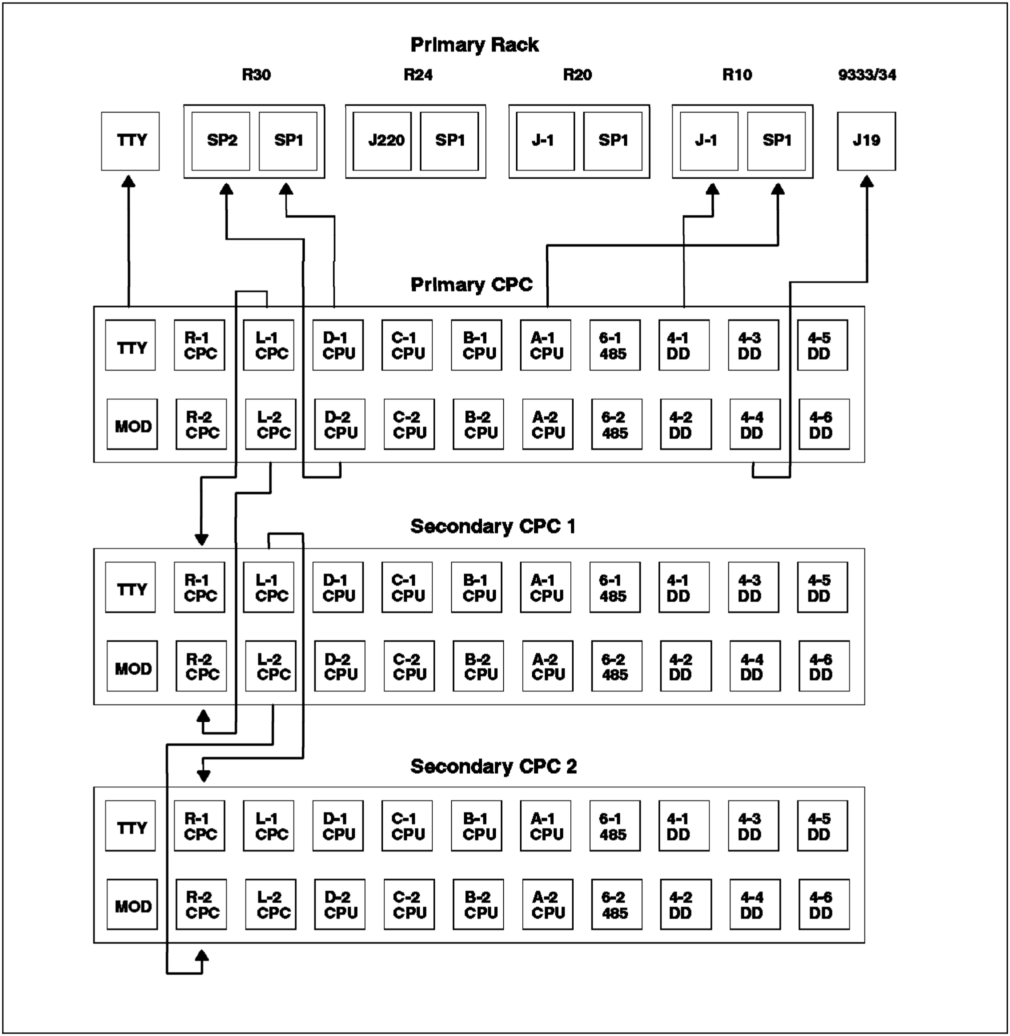

Primary Rack
R30
R24
R20
R10
9333/34
TTY
SP2
SP1
J220
SP1
J-1
SP1
J-1
SP1
J19
Primary CPC
TTY R-1 CPC L-1 CPC D-1 CPU C-1 CPU B-1 CPU A-1 CPU 6-1 485 4-1 DD 4-3 DD 4-5 DD
MOD R-2 CPC L-2 CPC D-2 CPU C-2 CPU B-2 CPU A-2 CPU 6-2 485 4-2 DD 4-4 DD 4-6 DD
Secondary CPC 1
TTY R-1 CPC L-1 CPC D-1 CPU C-1 CPU B-1 CPU A-1 CPU 6-1 485 4-1 DD 4-3 DD 4-5 DD
MOD R-2 CPC L-2 CPC D-2 CPU C-2 CPU B-2 CPU A-2 CPU 6-2 485 4-2 DD 4-4 DD 4-6 DD
Secondary CPC 2
TTY R-1 CPC L-1 CPC D-1 CPU C-1 CPU B-1 CPU A-1 CPU 6-1 485 4-1 DD 4-3 DD 4-5 DD
MOD R-2 CPC L-2 CPC D-2 CPU C-2 CPU B-2 CPU A-2 CPU 6-2 485 4-2 DD 4-4 DD 4-6 DD

Figure 110. Daisy Chaining CPCs

In this diagram, there are three CPCs. The connection between the CPCs uses the CPC left and CPC right connectors.

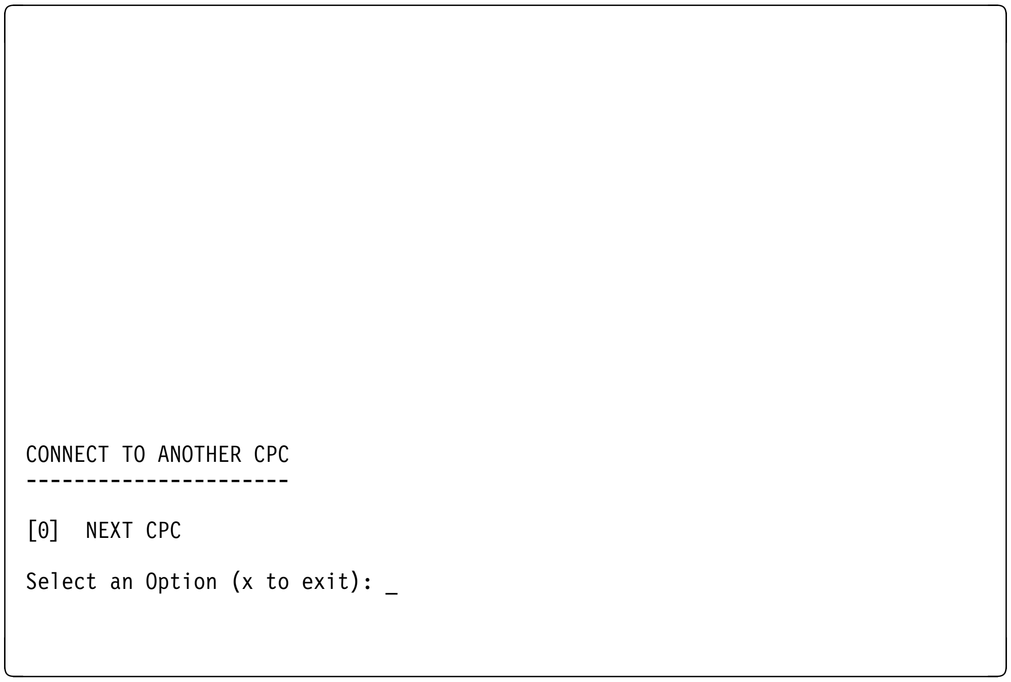
1. The CPC L-1 port of the first CPC (primary CPC) gets connected to the CPC R-1 port of the second CPC (secondary 1 CPC), and the CPC L-2 port of the first CPC (primary CPC) gets connected to the CPC R-2 port of the second CPC (secondary 1 CPC)
2. The secondary 1 CPC gets connected to the secondary 2 CPC by connecting CPC L-1 port of the second CPC to CPC R-1 port of the third CPC (secondary 2 CPC) and connecting the CPC L-2 port of the second CPC to the CPC R-2 port of the third CPC.

6.4.8 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 done by:

1. From the main menu, select the **TTY** option or modem option if connecting via the modem port.

2. Console CPC menu appears. From this menu, enter `1` to select the **Connect CPC** option. The following screen appears:



```
CONNECT TO ANOTHER CPC
----------------------

[0]  NEXT CPC

Select an Option (x to exit): _
```

Figure 111. 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 7. Installing an SMP System with AIX V4.1

The objective of this chapter is to describe how to install an SMP system with AIX V4.1. Installing an SMP system 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.

The intention of this chapter is not to cover all of the AIX V4.1 features and enhancements but to at least make you more comfortable with the items that relate to installing and managing your SMP system.

The areas that will be covered are:

- AIX V4.1 Packaging
- AIX V4.1 Installation Methods
- SMP Specifics
- UP to SMP Migration
- SMP Network Installation Example
- AIX V4.1 Software Maintenance

AIX Version 4.1 represents the most significant enhancements to AIX since its initial introduction. One of the main differences between AIX V3.2 and AIX V4.1 is the way AIX V4.1 is packaged.

7.1 AIX V4.1 Packaging

AIX V4.1 is packaged in two ways:

- AIX Version 4.1 for Clients (1-2 users license)
- AIX Version 4.1 for Servers (multi-user system).

The actual contents of the two packages are slightly different, the client package being a subset of the server. Although AIX V4.1 for Clients can be ordered on any systems, most of the SMP systems will be used as multi-user servers; so AIX Version 4.1 for Servers should be ordered for these systems.

One of the major goals for AIX Version 4.1 was to reduce the amount of disk space and memory required. It was broken up to several smaller parts; so you only need to install the parts that are required.

7.1.1 Packaging Terminology

It is useful to understand the terminology as it relates to the packaging of AIX Version 4.1 and associated products.

Fileset is the POSIX term for the smallest installable unit within a product. A fileset is part of the package. For example, bos.net.tcp.client. is a fileset for the bos.net package. This new packaging allows installation of only what is necessary (one or several specific filesets) generally requiring less disk space than AIX Version 3.

A **package** is a collection of filesets that are built together to form one installable image as a bff (backup file format) file. For example, bos.net is a package.

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 (bff) image.

Fileset Update is an update that corrects or enhances function in a previously installed fileset. Since each fileset can be serviced separately, fixes for AIX V4.1 (fileset updates) are smaller and more localized. These are equivalent to a subsystem update on AIX V3.2.

Bundle is a file that contains a number of filesets for installation. They can be thought of as an installation profile. A number of bundles are supplied with AIX Version 4.1 for various environments, for example, client, server and application development. A system administrator can create his own bundles if the default bundles are not suitable.

Update Bundle is a collection of fixes and enhancements that updates the installed software to the latest level available on the media.

Maintenance Bundle is a predetermined level of fixes and enhancements for the Base Operating System (BOS). It is equivalent to the PMP (Preventive Maintenance Package) levels on AIX V3.2.

Product Offering is a selected set of packages (or LPPs) which are shipped together on the same physical media.

7.1.2 Packaging Impacts

There are some significant changes to the packaging of AIX Version 4.1 over AIX Version 3. Items that are now included in AIX Version 4.1 are:

- AIXwindows 2D + CDE
- iFOR/LS Runtime
- Xstation Manager
- All Device Support (Note)

Note: The client media has most of the device drivers except a few that were determined to be for servers.

However, there are some options in AIX Version 3 that are now separate LPPs for AIX Version 4.1, and these need to be ordered separately:

- C Compiler
- Display Postscript
- InfoExplorer Databases
- Performance Tools
- X.25 Support

7.1.3 Bundles

The AIX V4.1 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           Pers-Prod.bnd
App-Dev.bnd          GOS.autoi            Server.bnd
App-Dev.def          Graphics-Startup.bnd Server.def
BOS.autoi            Hdwr-Diag.bnd
Client.bnd           Hdwr-Diag.def
```

Additional bundles can be created by the system administrator using the Install and Update Software Manager (Virtual Storage Manager), and 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.

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. The extended RAS support and hardware diagnostics are included in this bundle.

Personal Productivity

This bundle, Pers-Prod.bnd, installs packages and options that provide an enhanced *Personal Productivity* environment for AIX V4.1 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 Developer

This bundle, App-Dev.bnd, is essentially the same as the client with the addition of the filesets for application development and debugging.

Hardware Diagnostics

This bundle, Hdwr-Diag.bnd, includes the bos.diag and devices.base packages plus the diagnostics filesets for the devices that are discovered during BOS installation. Once this bundle is installed, the diagnostics filesets for new devices will also be installed when using `cfgmgr -i`.

Other

Update, Maintenance Level and All can be used as *bundles* for installation.

All these bundles are available on the both types of packages - AIX Version 4.1 for Clients and AIX Version 4.1 for Servers. There will be a difference to what is installed as some of the filesets are not available on the Client media.

7.1.4 Fileset Names

With the repackaging of AIX V4.1, the operating system was divided into a lot of smaller components or filesets. All the of the filesets were renamed for AIX V4.1.

The following conventions were used when naming a software package and its filesets for AIX V4.1:

- All the fileset names must be of the form Package_Name.Option where Option is unique for the software product package, and Package_Name is the name of the software product 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 software 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.

7.1.5 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 portion of a product
.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

7.1.6 Compatibility Filesets

Compatibility packages follow the same naming conventions as other packages, except that the word *compat* appears somewhere in the name. This warns the customer that this function will be removed in a future release of the product. For example, bos.compat.links, AIX V3.2 to AIX V4.1 Compatibility Links.

The Base Compatibility Function package contains those commands, links and tools that have been marked for removal from the Base Operating System. These commands are generally not standard-compliant commands and should only be applied as needed since the information and commands they contain will not be available in a future release of AIX.

bos.compat.cmds This fileset contains commands that are being removed from the AIX system. These commands are generally not standard-compliant commands. For example, the `copy` and `li` commands are being removed.

bos.compat.imk This fileset provides input method keymaps and links which may be necessary for those applications that depend on the input method conventions used in AIX Version 3.1.

bos.compat.links This fileset provides backwards compatibility for those links that were added in the AIX Version 3.2 release for Diskless Support, but have been removed in the AIX Version 4.1 release.

bos.compat.lan Provides a COMIO interface for token-ring, Ethernet and FDDI devices that are compatible with the COMIO interface present in AIX Version 3.2.5. It supports usermode access only.

bos.compat.msg This fileset provides symbolic links for backwards compatibility for message catalog packages that use the AIX Version 3.1 naming conventions for language-territory.

bos.compat.net This fileset provides backwards compatibility for those links that were added in the AIX Version 3.2 bosnet.obj product.

bos.compat.NetInstl This fileset provides the capability to serve Version 3.2 clients with a Version 4.1 server.

bos.compat.termcap This fileset provides an older method of defining terminal definitions. Most people use terminfo. The library provides terminfo interface routines for curses applications. The same routines are provided by libcurses.a, as well.

bos.compat.termcap.data This fileset provides the termcap tabset files for various terminals.

There are also a number of compatibility filesets for X11R3, X11R4, Motif 1.0, Motif 1.1.4, and X11 fonts. If an application is not working on AIX V4.1, check the list of installed compatibility filesets. All the compatibility filesets will be installed as part of the Migration Installation.

7.1.7 Device Driver Packaging

The AIX V4.1 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 config 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 RISC System/6000 Models 220 and 230. **buc** is the PowerPC 601 bus that is used to connect the graphic adapters in the RISC System/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.

Note: Install all the device drivers or a superset of the device support required for all your systems, if you are cloning AIX V4.1 systems with `mksysb`. Turn on your tape drive if you are installing from CDROM.

7.1.8 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 V4.1 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 V4.1 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.

7.1.9 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 V4.1. 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`.

7.2 AIX Version 4.1 Installation Methods

AIX V4.1 will now install a smaller, lighter system during the BOS installation than we are used to with AIX V3.2. This is done to economize on disk space, to avoid the redundancy of installed programs, and to reduce the BOS installation time. The exception is *Migration Installation* which will attempt to install the equivalent function of the original AIX V3.2 system.

Just as in AIX Version 3, there are a number of choices for installing AIX V4.1, and these are as follows:

- New and Complete Overwrite installation from an AIX V4.1 product tape or CD-ROM
- Preservation Install
- Migration Install from AIX V3.2
- mksysb Install
- Network Install

7.2.1 Installation Flow

In order to boot a system to get to the Installation and Maintenance menu, the procedure is the same as for AIX V3. That is, turn the key to Service position, and switch on with installation media in the appropriate device, which will be either tape or CD-ROM.

Note: Remember that a diskette can no longer be used as boot media for AIX V4.1. 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, we had 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.

You can choose the option **Start Install Now with Default Settings**, if you are confident it will give you what you want. This varies between a Preservation Install if you are coming from another Level of AIX V4.1 or AIX V3.1 or a Migration Install if you are installing onto an existing AIX V3.2 system. We generally found it better to choose the option **Change/Show Installation Settings** rather than the Install Now with Default Setting because the default settings can vary as described above, 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 V4.1.

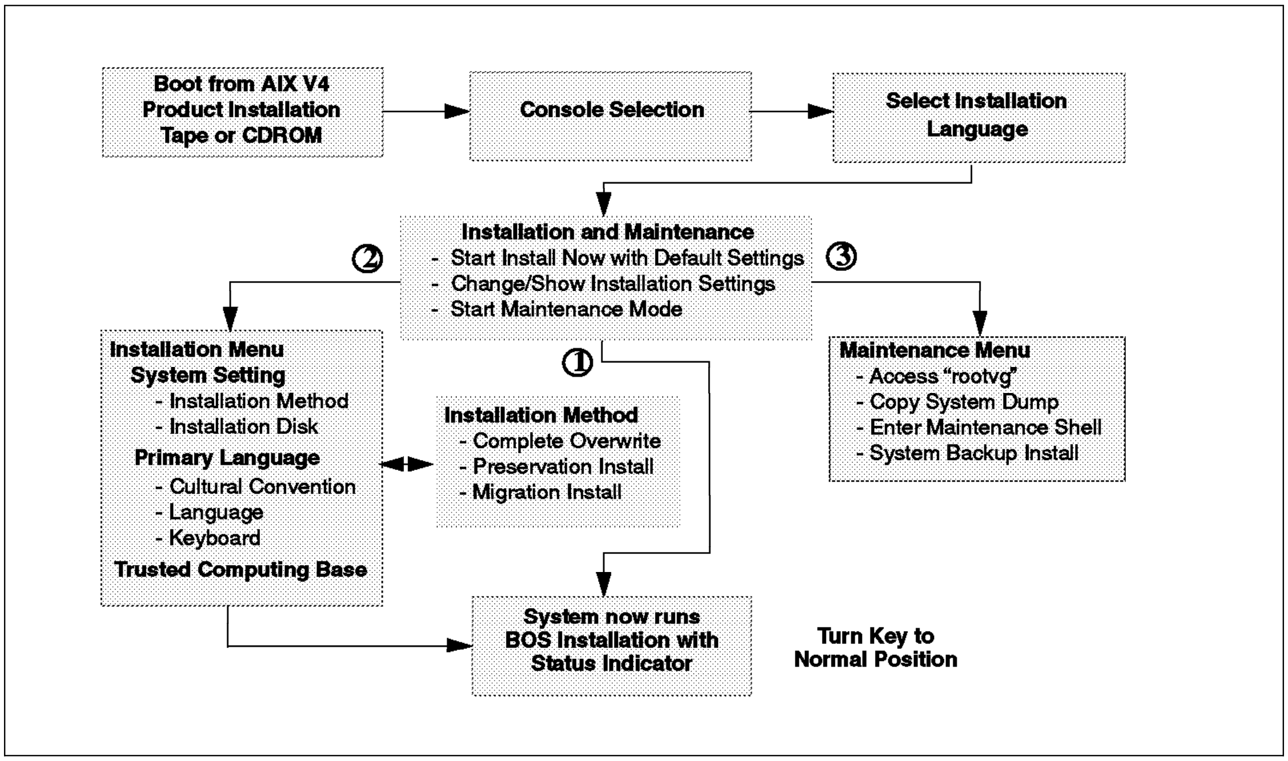


Figure 112. AIX V4.1 Installation Flow

7.2.2 Default Installation

AIX Runtime is the term used to describe the minimum disk utilization installation available with AIX Version 4.1. It is the default environment if no additional software is installed.

The packages and options included in this installation are automatically installed by the BOS product installation, and they are listed in the BOS.autoi auto installation file. This installation includes the bos.rte package, the appropriate bos.rte.up (Uniprocessor) or bos.rte.mp (Multiprocessor) option, the required device driver support (devices.xxx.xxxx), and the base packages/options. The BOS.autoi file will install additional required support, such as a minimum set of terminal information files, ASCII SMIT and ASCII install assistant.

If the system has a graphics adapter (which can be the case with the G30 model), an additional set of packages listed in the GOS.autoi file will be automatically installed. This will install the minimum AIXwindows 2D environment along with graphical SMIT, graphical install assistant and the visual system management support. Otherwise, the filesets listed in the ASCII.autoi file will be installed. If a non-C primary language and locale were specified, the message and locale packages/options (for the primary language/locale) matching the installed filesets will also be installed.

7.2.3 New and Complete Overwrite Installation

The New and Complete Overwrite Installation is done either on a new machine which has nothing installed on it or on an existing machine where the data and programs are no longer required or easily recreated.

Choose **Change/Show Installation Settings** if you are going to do a Complete Overwrite Installation, and then you can change the System Settings to set the target disk(s) for the installation. The Primary Language option allows you to select a language which will take effect after the first boot. The language selection will be based on the following locale data, and generally they will be the same, although you have an option to choose your own combination.

Cultural Convention - to set date, time, money

Message Catalogue - to set primary language

Keyboard

There is also an option to install the Trusted Computing Base (TCB) which will install the trusted shell and trusted path and will enable system integrity checking, should you require this. The TCB option is only supported for a Preservation or a Complete Overwrite installation.

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 new LED codes displayed during BOS installation and the associated status messages that are displayed on the screen are listed below:

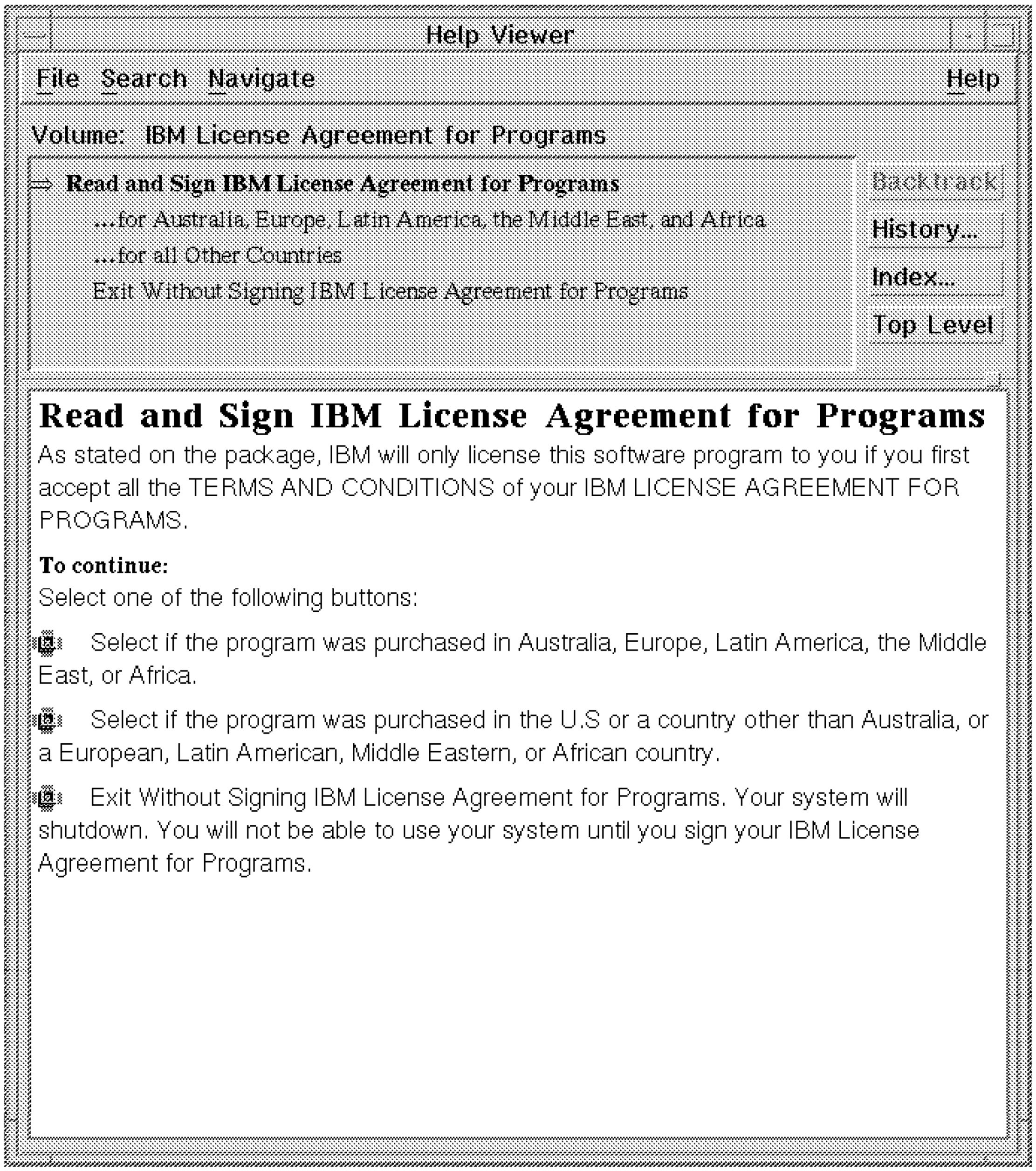
c48 (initial menu)
c50 Preparing target disks
c46 Making paging logical volumes
c46 Making boot logical volume

```
c46 Making logical volumes
c46 Forming the jfs log
c46 Making file systems
c46 Mounting file systems
c54 Restoring base operating system
c52 Initializing disk environment **
c54 Installing all packages **
    < installp summary information > **
    < copyright information for each package > **
    < installp completion summary information > **
c46 Creating boot image
    ( rebooting ... )
```

The messages with ** next to them in the above list will not be displayed for a mksysb installation.

7.2.3.1 Software License Agreement

After installing AIX V4.1, you are presented with the licensing agreement. At this point in time, some countries are not required to sign the license agreement, but a country option must be selected to be able to continue. If you use the exit option, the system will continue to prompt you.


Help Viewer

File Search Navigate Help

Volume: IBM License Agreement for Programs

⇒ **Read and Sign IBM License Agreement for Programs**
...for Australia, Europe, Latin America, the Middle East, and Africa
...for all Other Countries
Exit Without Signing IBM License Agreement for Programs

Backtrack
History...
Index...
Top Level

Read and Sign IBM License Agreement for Programs

As stated on the package, IBM will only license this software program to you if you first accept all the TERMS AND CONDITIONS of your IBM LICENSE AGREEMENT FOR PROGRAMS.

To continue:
Select one of the following buttons:

Select if the program was purchased in Australia, Europe, Latin America, the Middle East, or Africa.

Select if the program was purchased in the U.S or a country other than Australia, or a European, Latin American, Middle Eastern, or African country.

Exit Without Signing IBM License Agreement for Programs. Your system will shutdown. You will not be able to use your system until you sign your IBM License Agreement for Programs.

Figure 113. IBM License Agreement

7.2.3.2 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 V4.1 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
```

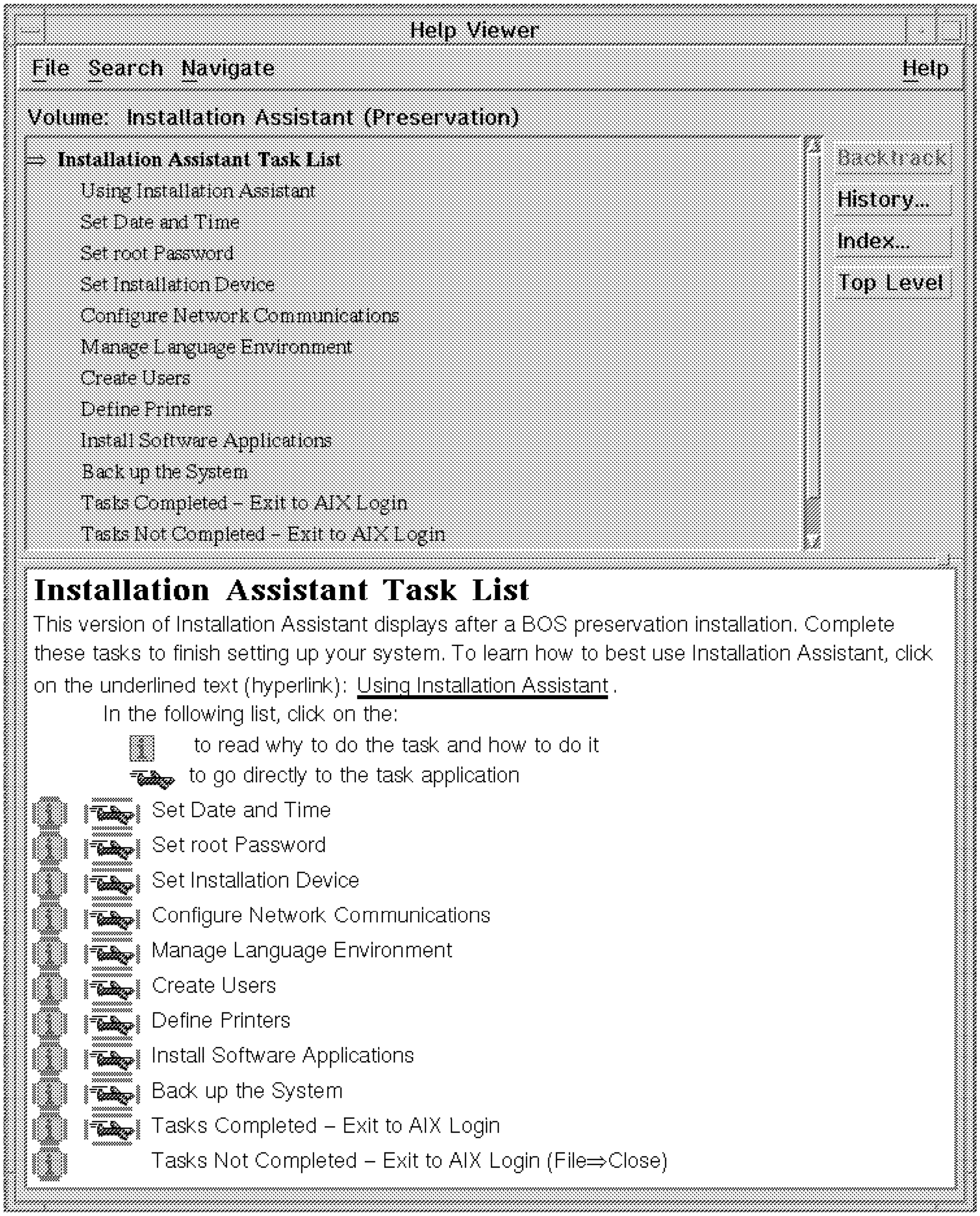
for the ASCII version or

```
# install_assist &
```

for the GUI version, as shown in Figure 115 on page 181.

```
                         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 114. ASCII Installation Assistant



Volume: Installation Assistant (Preservation)

⇒ **Installation Assistant Task List**
Using Installation Assistant
Set Date and Time
Set root Password
Set Installation Device
Configure Network Communications
Manage Language Environment
Create Users
Define Printers
Install Software Applications
Back up the System
Tasks Completed - Exit to AIX Login
Tasks Not Completed - Exit to AIX Login

Installation Assistant Task List

This version of Installation Assistant displays after a BOS preservation installation. Complete these tasks to finish setting up your system. To learn how to best use Installation Assistant, click on the underlined text (hyperlink): Using Installation Assistant .

In the following list, click on the:

to read why to do the task and how to do it

to go directly to the task application

Set Date and Time
Set root Password
Set Installation Device
Configure Network Communications
Manage Language Environment
Create Users
Define Printers
Install Software Applications
Back up the System
Tasks Completed – Exit to AIX Login
Tasks Not Completed – Exit to AIX Login (File⇒Close)

Figure 115. 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.

7.2.3.3 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, the dump image is copied from the paging space to the /var filesystem on reboot. If the system has a lot of memory, there may not be enough space in this filesystem 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, multi-user server is to use a dedicated dump device so that the /var filesystem 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.

```
# 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.

7.2.3.4 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
```

7.2.3.5 bosinst.data and image.data Files

You have the option of customizing subsequent installations once AIX is installed and avoid having to answer prompts on your console in order to install the system. In order to avoid prompts on your console, you need to follow the steps detailed below after you have completed your first AIX V4.1 system installation.

1. Copy the file **/var/adm/ras/bosinst.data** to **/bosinst.data**
2. Edit the variables in that file as per the comments in the file itself. Another reference is the *AIX Version 4.1 Installation Guide*.
3. Create a file called */signature* that has a one-line entry with the word “data” in it.
4. Back these files up to a diskette:

   ```
   # ls ./signature ./bosinst.data | backup -iqv
   ```

5. 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 V4.1 onto two disks in a system, the bosinst.data file was used:

```
control_flow:
    CONSOLE = /dev/lft0
    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 = 00-08-00-0,0
    SIZE_MB =
    HDISKNAME =

target_disk_data:
    LOCATION = 00-08-00-1,0
    SIZE_MB =
    HDISKNAME =

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 4.1 Installation Guide* in the chapter “Customizing the BOS Install Program.”

You can also modify the image.data file which 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* for a description of the image.data file.

7.2.4 Preservation Installation

The AIX V4.1 Preservation Install will perform the same functionality a s it did in in 3.2. This installation preserves the /etc/filesystems file and /home directory as well as user-created volume groups and logical volumes. A Preservation Install will also preserve the previous dump and page devices.

7.2.5 Migration Installation

A Migration Installation is only offered as an option when booting from an AIX V3.2 system, and this option will not appear on the Installation Menu if the previous version was AIX V3.1. When migrating from AIX V3.2 to AIX V4.1, AIX V3.2.5 is the preferred version for production environments to be migrated from.

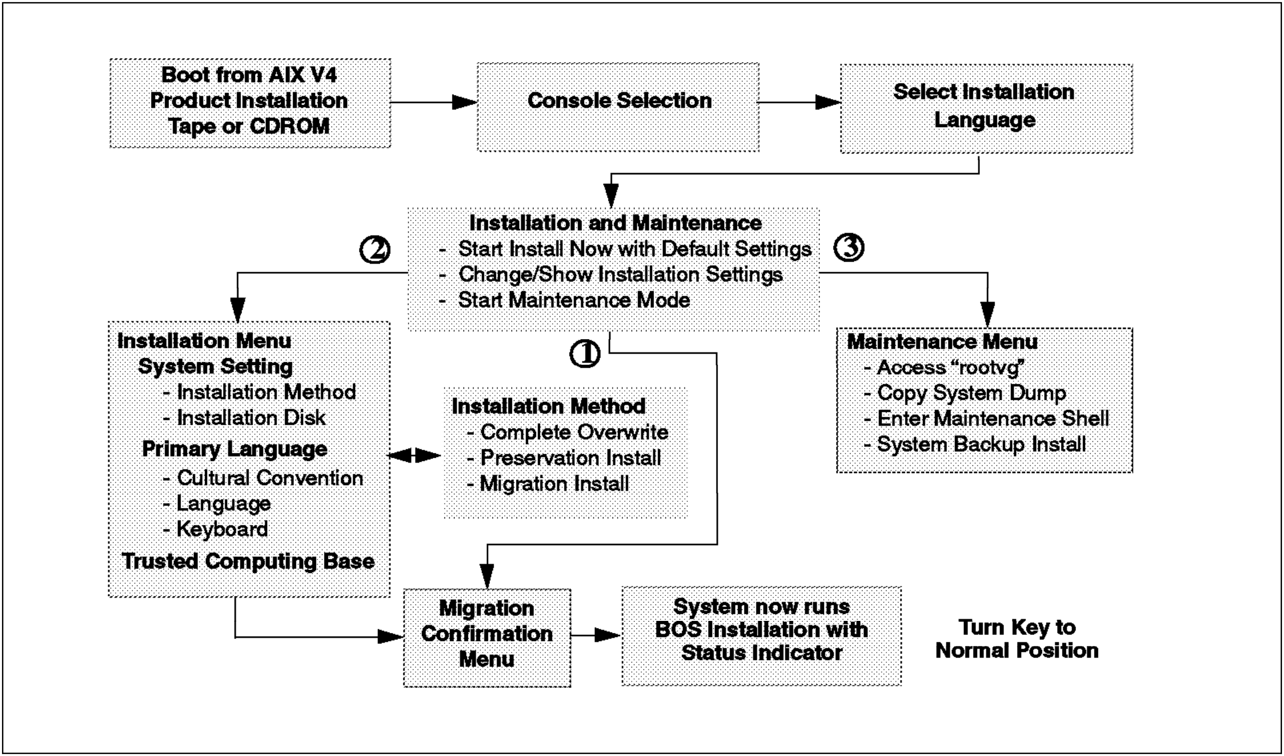


Figure 116. Installation Flow for Migration

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 filesystem.

The Migration Installation will then remove the bos.obj files based on the 3.2 software VPD (Vital Program Data). It will then install bos.rte, and it will automatically install the equivalent, non-chargeable 4.1 filesets (anything included in the AIX V4.1 client/server package) in place of the AIX V3.2 filesets. The only exception to fileset mapping is that only the Data Link Control devices that are defined and configured on the system will be reinstalled. All the compatibility filesets are installed. The device driver support is completely reinstalled with the new AIX V4.1 device drivers. The following products will be removed from the system:

- AIXwindows Interface Composer
- BOS ADT xde
- Display Postscript
- XL C Compiler
- OpenGL, PEX, PHIGs and X11 3D
- PC Simulator

Any AIX V3.2 LPPs or vendor products that were installed on the system will be left untouched. Finally, the migration routine will invoke a routine which will merge the configuration files back to their previous state.

We would suggest that when you migrate to AIX V4.1, do not install the CDE runtime environment straight away. Test your existing X11 environment on AIX V4.1 first, and then when you have sufficient time and knowledge, move to CDE.

7.2.5.1 Obsolete Entries

If a system has been migrated from AIX V3.2, you will see a number of obsolete entries by using `lslpp -l`. These filesets (options) contain files that were not completely replaced by the AIX V4.1 equivalent filesets. In some cases, the files or programs (such as X.desktop) will still work, but are not shipped with AIX V4.1. Do NOT try and deinstall these filesets because some of the files will also be part of the AIX V4.1 equivalent filesets.

```
X11fnt.coreX.fnt     1.2.3.0  OBSOLETE  AIXwindows Core X11 Fonts
X11fnt.oldX.fnt      1.2.3.0  OBSOLETE  AIXwindows Miscellaneous X
X11rte.ext.obj       1.2.3.0  OBSOLETE  AIXwindows Run Time
X11rte.motif1.2.obj  1.2.3.0  OBSOLETE  AIXwindows Motif 1.2 Run Time
X11rte.obj           1.2.3.0  OBSOLETE  AIXwindows Run Time
bos.obj              3.2.0.0  OBSOLETE  The Base Operating System
bosadt.lib.obj       3.2.0.0  OBSOLETE  Base Development Libraries &
bosext1.extcmds.obj  3.2.0.0  OBSOLETE  Extended Commands
bsl.en_US.aix.loc    3.2.0.0  OBSOLETE  AIX Locale - English ( United
bsl.en_US.pc.loc     3.2.0.0  OBSOLETE  PC Locale - English ( United
bsl.lat-1.fnt.loc    3.2.0.0  OBSOLETE  HFT Latin-1 Font Library
bos.obj              3.2.0.0  OBSOLETE  The Base Operating System
bosext1.extcmds.data 3.2.0.0  OBSOLETE  Extended Commands
```

7.2.6 mksysb Installation

A system can be installed by restoring a backup of a previously installed system. Figure 117 on page 186 shows the installation flow from such a backup.

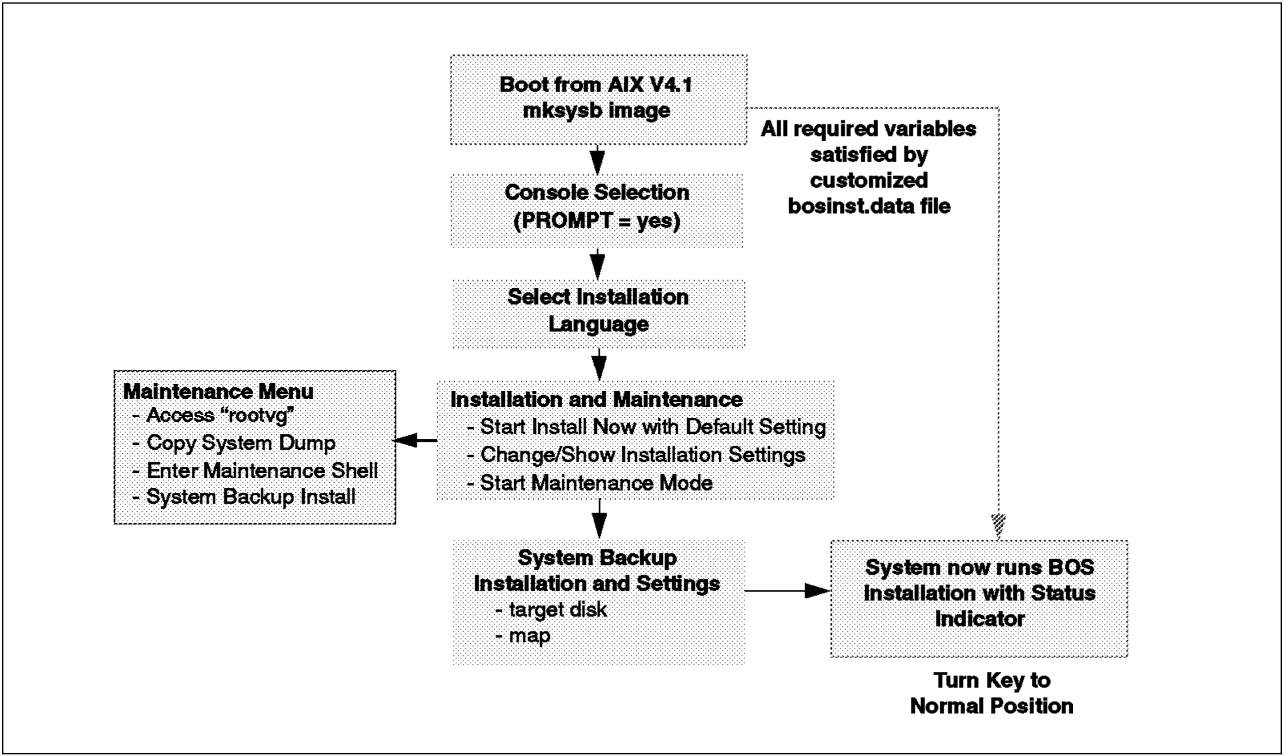

Boot from AIX V4.1 mksysb image
All required variables satisfied by customized bosinst.data file
Console Selection (PROMPT = yes)
Select Installation Language
Maintenance Menu
- Access "rootvg"
- Copy System Dump
- Enter Maintenance Shell
- System Backup Install
Installation and Maintenance
- Start Install Now with Default Setting
- Change/Show Installation Settings
- Start Maintenance Mode
System Backup Installation and Settings
- target disk
- map
System now runs BOS Installation with Status Indicator
Turn Key to Normal Position

Figure 117. Installation Flow for mksysb

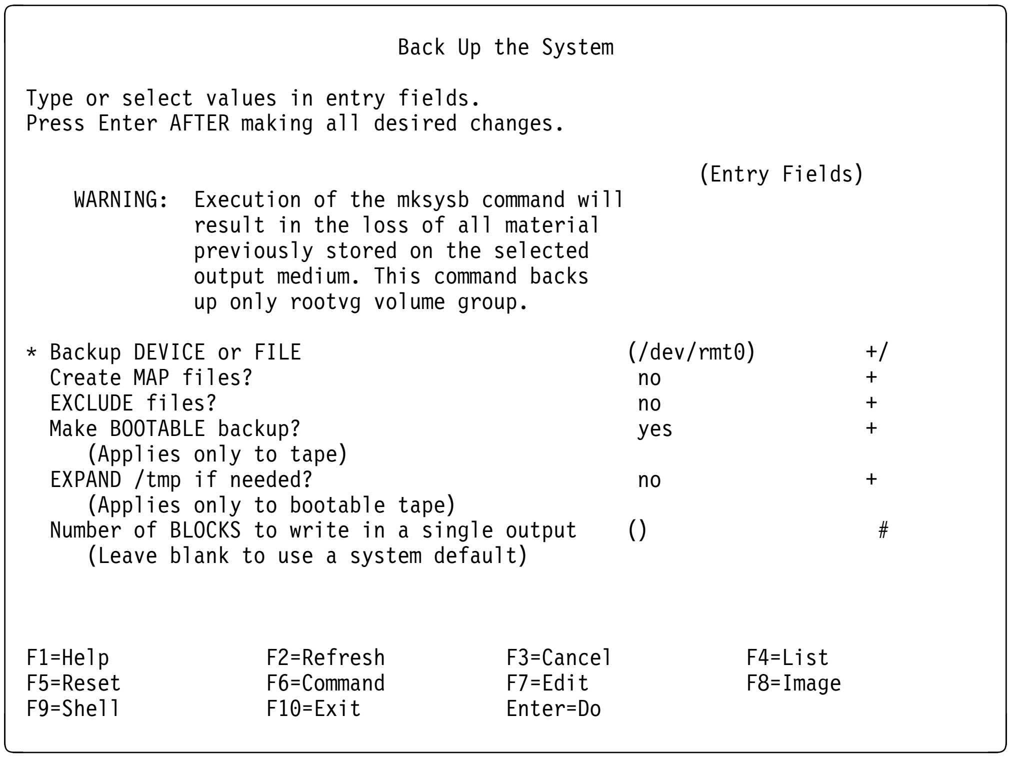
In order to create a mksysb, you need to type in

```
# smitty mksysb
```

or take the option

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

from the initial SMIT screen, where you will see the option `System Backup Manager`. The AIX V3.2 fastpath `smitty startup` and the option to `Backup the System .` on the smitty fs menu are now excluded. The screen will look like this:



```
                               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 118. 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 1

Since we are now dealing with minimum install environments in AIX V4.1, it is possible that a mksysb from one system will not install on another machine because of differing requirements in device support.

If you intend to install multiple machines with a mksysb image, you must install the required device support onto the original machine before the mksysb image is created.

Attention 2

If you are trying to install a mksysb that has been generated on a UP system after using the upgrade option to migrate AIX V3.2.5 to AIX V4.1.x, you have to boot off your installation media and then choose option **3** (Start Maintenance Mode) from your Installation Menu. This is because the `smit mksysb` from the UP system will not create an MP-bootable tape; it will create a UP-bootable tape only.

7.2.7 Network Installation

The Diskless Workstation Manager (DWM) utilities in AIX V3.2 have been replaced by the Network Installation Manager (NIM) in AIX V4.1. NIM can manage diskless and dataless systems running AIX V4.1 as well as network install stand-alone systems. Using NIM, you can set up an installation once for machines with identical requirements or customize an installation for the specific requirements of a given machine. More than one machine can be installed at the same time although the number of machines installed simultaneously will depend on the throughput of the network.

The NIM environment is comprised of server and client machines. A server provides the resources (files and programs), and a machine that is dependent on a server is the client.

Before using NIM, NFS and TCP/IP must be installed and configured, and the networks that will be part of the installation must be configured. Also, gateways must be initialized and name resolution must be set up so that all machines that will be part of the network have a resolvable host name.

SMP systems can be installed via the network using NIM. There are some differences when compared to the uniprocessor systems.

- The only supported configuration is as a stand-alone system. All the filesystems and resources are located on the SMP system's local disks. Diskless or dataless configurations are not supported on the SMP.
- Ethernet and token-ring are the only supported network interfaces for network install. This is a limitation of the support in the Read-Only Storage (ROS) on the SMP systems. The IPL ROM emulation diskette that is used on the RISC System/6000 UP systems to provide the FDDI support does not work at all on the SMP systems.
- The boot process at the hardware level is different for the SMP machines. The SystemGuard Maintenance Menu that was discussed in Chapter 5, “SystemGuard” on page 99 is used to select the network interface and boot from a NIM server.

7.3 SMP Specifics

This section introduces some AIX V4.1 specifics to the SMP systems.

7.3.1 AIX V4.1 Levels for SMPs

A number of levels of AIX V4.1 are already available, and additional support will be added over time. The following list shows what has been delivered, so far, and the AIX V4.1 levels that are supported on the SMP systems.

AIX V4.1.0	US English only
AIX V4.1.1	Internationalization
AIX V4.1.2	SMP support (Up to four processors)
AIX V4.1.3	SMP support (Up to eight processors)

This section is a collection of SMP-specific items that relate to AIX V4.1.

7.3.2 MP Kernel

AIX V4.1 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.

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)

The BOS Runtime fileset, bos.rte. becomes a number of smaller filesets after installation. These filesets are named 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 link /unix to the correct kernel.

This link is only used by the `bosboot` command when rebuilding the boot image.

During bootup, the platform type is checked to see if it is a UP or an MP system, and the appropriate kernel is loaded using the full pathname, /usr/lib/boot/unix_up or /usr/lib/boot/unix_mp.

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 V4.1.

Note that **the MP kernel will work on a UP system**. However, performance might not be as good as it would be if the UP system was running a UP kernel. The performance penalty is typically about 10 percent.

However, **the 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 system 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.

The bos.rte.mp fileset contains the following files:

```
# lslpp -f bos.rte.mp
         Fileset               File
  ----------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.rte.mp 4.1.2.0    /usr/sbin/open_door
                        /usr/sbin/cpu_state
                        /usr/sbin/bindprocessor
                        /usr/lib/boot/unix_mp

Path: /etc/objrepos
  bos.rte.mp 4.1.2.0    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 8, “SMP Performance Tools” on page 209.

7.3.3 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 IBM RISC System/6000 that have a Micro-Channel bus.
rs6ksmp	applies to symmetric multiprocessor models.
rspc	applies to the PReP-compliant systems that have an ISA bus, such as the RISC System/6000 40P.

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 Version 4.1, 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.disk.proto
/usr/lib/boot/rspc.cd.proto
/usr/lib/boot/network/rs6k.ent.fddi

/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.*

7.3.4 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 systems.

`bootinfo -z` will return a numeric:

`0`	The machine is not MP-capable
`1`	The machine is MP-capable

`bootinfo -T` will return one of the following:

`rs6k`	which means RISC System/6000 UP
`rs6ksmp`	which means RISC System/6000 SMP
`rspc`	which 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

7.3.5 Creating an MP Boot Image

The `bosboot` command is used to create a boot image, and this is the same as in AIX V3.2. It uses the prototype files listed in the Platform Types section.

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 would use to create an MP boot image is:

```
# bosboot -a -d hdisk0 -k /usr/lib/boot/unix_mp -L
```

where

`-a`	Creates a complete boot image and device.
`-d`	Specifies the boot device. This flag is optional for the hard disk.
`-L`	Enables lock instrumentation for MP systems 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 system.
`-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).

Two other flags could be of interest :

`-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. But if you want to create an MP boot image on a UP system, you need to specify the type, `rs6ksmp`.
`-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.

7.3.6 SMP CPU-ID

On traditional IBM RISC System/6000 UP systems, the output of the `uname -m` command is tied to the serial number of the CPU card.

Since an SMP system has several processors, the CPU-ID used for product licenses cannot be tied to a processor and must be unique.

Therefore, for SMP systems, 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 (A0 for a 7013-J30, A6 for a 7012-G30, and A3 for a 7015-R30)

Note: For SMP systems built in Austin, the CPU-ID is the serial number of the machine. This is not the case for machines built in EMEA.

7.3.7 New SMP Devices

If you issue the following command,

```
# lsdev -C
```

you will see a number of new devices that were not present in the 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 J30 and R30 will also show two Micro-Channel busses (bus0 and bus1).

7.4 UP to SMP Migration

Upgrading a UP to an SMP system is different from upgrading a UP to another faster UP because the SMP technology is different from the UP technology. Therefore, migrating from a UP to an SMP system must be done carefully and with some knowledge of the differences between a UP system and an SMP system.

The intent of this section is not to describe in detail how to migrate a UP system to an SMP system but to outline the main steps and the main issues when performing such an upgrade. A detailed procedure and utilities should come with the upgrade.

Most of the time the UP system to be upgraded will be running AIX V3.2. Since the SMP requires at least AIX V4.1.2 to run, and AIX migration from AIX V3.2 to AIX V4.1.2 or later will first be necessary on the UP system before migrating the reusable hardware to the SMP system.

7.4.1 Migration Checklist

When migrating a UP system to an SMP system, there are a number of items that need to be checked before the upgrade can take place.

These key items are:

- AIX V4.1 device support:

  If you are running AIX V3.2.5 on your UP system, not all the existing adapters are supported by AIX V4.1. This may mean that you wait for the device support to be available, or replace the old hardware with supported hardware (for example - replacing a 64-port card with a 128-port adapter).

- AIX V4.1 LPPs support:

  You need to check that the installed LPPs are supported on AIX V4.1. If new versions are required for AIX V4.1, they need to be ordered and available before migrating the system to AIX V4.1.

  You might have to wait for the availability of some LPPs before starting the migration to AIX V4.1 on the UP system.

- Software that are no longer included in AIX V4.1:

  You need to check that there is no software which is required, but not included, in AIX V4.1. For example, the C compiler and InfoExplorer are required to be ordered separately, but they were part of AIX V3.2.

- Custom-made application availability:

  You need to identify all custom-made applications running on your system. Then port them on AIX V4.1. Porting these applications on a UP system running AIX V4.1 is not sufficient. The applications must be MP safe before migrating them to the SMP system.

- Third-party applications availability:

  If your environment is dependent on key applications from a software vendor, you must first contact the software vendor to ensure that their application is supported on AIX V4.1 and MP safe. In some cases, a new version might be required so you must order an upgrade.

- Third-party hardware availability:

  If you use third-party hardware (such as adapters, terminals, printers, modems) in your environment, it is advisable to verify that this hardware is supported and works with AIX V4.1.

- Reusable hardware:

  Before starting the upgrade, you need to check which hardware you can reuse through the upgrade. Some adapters, disks, memory SIMMs are reusable. Some hardware are not supported like the 64-port adapter.

7.4.2 Migration Procedure

When upgrading a UP system to an SMP system, there are two cases. The UP is at AIX V3.2.5, or the UP is already at AIX V4.1.3. Since the second case is a subset of the first case, we are only going to cover the first case.

The main upgrade steps are the following:

- Back up the existing environment

- Document your system (which hardware is installed at which location, which LPPs are installed and so on)
- Migrate the UP system to AIX V4.1.x (4.1.2 or 4.1.3 or later)
- Test your environment at AIX V4.1.x
- Back up the system again
- Install all the required device drivers and filesets such as the MP kernel
- Create the MP kernel links
- Create an MP boot image
- Create a bootable tape (using `mksysb`)
- Migrate the reusable hardware
- Boot on the SMP with the bootable tape
- Reload the system backup

Attention

The following sections are not a step-by-step procedure for upgrading a UP system to an SMP. They just give an idea of the main steps in the upgrade process. A detailed procedure provided with the upgrade kit should be followed carefully.

7.4.3 Migrating the UP System to AIX V4.1

The third step is to migrate the environment to AIX V4.1 on the existing UP system. This is where the above considerations regarding device support, LPPs, third party applications, database, and hardware need to be taken into account to ensure that there are no problems because this is the highest risk area.

Note: If you intend to migrate your system to AIX V4.1.2 instead of AIX V4.1.3 or later, before you start to install AIX V4.1, you need to set your dump device to the paging area because there are problems with AIX V4.1.2. If not, you will not be able to create new logical volumes or change the size of /tmp after the migration to AIX V4.1.2. This problem is fixed in AIX V4.1.3.

Use the following command to set the dump device to the paging space:

```
# sysdumpdev -p /dev/hd6 -P
```

Also, you need to check that there is about 8 MB free in /tmp before starting the migration. If you are short of disk space, you can remove the dump logical volume. Use the following command to free-up the allocated PPs (2 PPs default):

```
# rmlv -f hd7
```

Boot your system from your installation media, and take the option for Migration Installation from the screen when it is presented. Be aware that the Migration Install is slightly different from Complete Overwrite or Preservation Install as the user will be presented with a migration confirmation menu as follows.

```
migration menu preparation in progress



   Please wait ...





      Approximate            Elapsed time
   % tasks complete          (in minutes)


          12                       2          < ... Status Message ... >
```

The point is not to walk away from the machine thinking that it will go ahead and install; it won't. It will display a menu. It takes two to three minutes, waiting for input, to display the following menu:

```
Either type 0 and press Enter to continue the installation, or type the
number of your choice and press Enter.

     1  List the saved Base System configuration files which will not be
     merged into the system.  These files are saved in /tmp/bos.

     2  List the filesets which will be removed and not replaced.

     3  List directories which will have all current contents removed.

     4   Reboot without migrating.

>>>  0  Continue with the migration.

88  Help

+-----------------------------------------------------------------------
WARNING: Selected files, directories, and filesets (installable
options) from the Base System will be removed. Choose 2 or 3 for more
information.
```

Figure 119. Migration Confirmation Screen

After the migration has completed, a software license agreement that you need to complete will come up, and a smaller version of the Installation Assistant (as per the following figure) will appear. You can complete any of the tasks here that you might want.

```
                   Installation Assistant

Move Cursor to desired item, and press Enter

   Set Installation Device
   Manage Language Environment
   Install Software Applications
   Backup the System
   Task Completed - Exit to AIX login



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

Some of the above options will also start up the Visual Systems Manager should you have a graphics screen as a console on your UP system.

There are a number of files in the /tmp/bos directory that could be of interest. A list of the removed filesets is stored in /tmp/bos/filesets.gone, and a list of directories removed is stored in /tmp/bos/directories.gone.

7.4.4 Dump Device and Paging Space

If, before a migration to AIX V4.1.2, you forgot to point your dump device to the paging area with the `sysdumpdev` command, you can perform the following workaround. It fixes a discrepancy between the device information in the boot image and the LVM data for /tmp.

```
# su
# putlvodm -L  getlvodm -l hd3
# lvrelminor hd3
# synclvodm rootvg hd3
```

Check your system dump and paging. Since you are dealing with a server, it would be best to have a dedicated dump device; so create an `hd7` logical volume by issuing:

```
# mklv -y'hd7' -a'e' rootvg <number of PPs>
```

where `<number of PPs>` is the number of Physical Partitions that you require, which you can get by using the command:

```
# sysdumpdev -e
```

You would then point your dump device to your new logical volume. For example,

```
# sysdumpdev -P -p /dev/hd7
```

It would also be beneficial to check your paging requirements and adjust as necessary.

7.4.5 Installing Required Device Drivers

Once your system is at AIX V4.1, it is **essential** that you add the MP-specific device support (and device support for any new devices that are on the SMP system) onto the UP system. Otherwise, these devices might not be enabled, or you might not even be able to install the backup tape on the SMP. In order to do this, enter the following SMIT fastpath:

```
# smit install_selectable_all
```

Use PF4 to list what is on your installation media; use / to find the item and PF7 to select the required items:

- bos.diag.rte - Hardware diagnostics for SystemGuard commands `mpcfg` and `keycfg`
- bos.sysmgt.serv_aid - Software error logging and service aids for SystemGuard daemons, `survd` and `mirrord`
- bos.rte.mp - MP Kernel
- devices.mca.8efc - 16-bit SCSI I/O controller
- devices.mca.fed9 - Standard I/O adapter
- devices.rs6ksmp.base - SMP Base System Device Support

This should be enough device support to install onto your SMP system. Additional device support can always be added after the system has been installed by using SMIT or by issuing:

```
# cfgmgr -i /dev/rmt0
```

If you do not want to follow this route, you can add all the device support on your installation media onto your UP system before you migrate, but this option uses extra disk space and requires more resource for updating.

7.4.6 Creating MP Kernel Links

The next step is to create the link from /unix to /usr/lib/boot/unix_mp on the UP system by issuing the command:

```
# ln -fs /usr/lib/boot/unix_mp /unix
```

so that you can create the correct boot image on the UP system before performing the mksysb.

7.4.7 Creating an MP Boot Image

You must create an MP boot image with the following command:

```
# bosboot -a -d /dev/hdisk0 -k /usr/lib/boot/unix_mp -T rs6ksmp
```

Once the MP boot image has been created, you must install an upgrade utility which is provided on a diskette with the upgrade kit. You must run that upgrade utility (shell script).

At this point you need to back up the system and create a bootable tape. Then you can migrate the reusable hardware to the SMP.

7.4.8 Restoring the Backup

Before booting the SMP system from your backup tape, you will need to reset the NVRAM from the SystemGuard Stand-By menu. A specific procedure is provided with the upgrade kit.

Then boot the SMP system from your backup tape. At your initial Installation 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): _
```

Choose option **3** to `Start Maintenance Mode for System Recovery`.

```
                    Maintenance

    Type Number

>>> 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): _
```

Choose the tape drive you want to install from on the next screen, and after one or two minutes, you will be presented will a screen that will allow you to choose the language that you will use for the install. This will then present you with the following screen:

```
        System Backup and Installation settings

Either type 0 and Press Enter to Install with the Current settings, or
type the number of the setting you want to change, and press Enter.

      Setting:                            Current Choice(s):

    1 Disks where you want to install.........hdisk0
              Use maps........................No
    2 Shrink Filesystems......................No


     >>> 0 Install with the settings listed above.


    88  Help ?              +--------------------------------------------
    99 Previous Menu        | WARNING-Base Operating System Installation
                            | will destroy or impair ALL data on the
                            | destination disk hdisk0.
>>>  Choice (0): _
```

Choose option **1** to check that you are putting it on the correct disks, and you can then start the install of your mksysb. This presents you with the following screen:

```
                  Installing Base Operating System

Turn the key to the NORMAL position any time before the
installation ends.

         Please wait...




      Approximate           Elapsed time
   % tasks complete         (in minutes)

          57                      10         < ... Status Message ... >
```

7.5 Example of an SMP Installation Using NIM

In this example, we use an IBM RISC System/6000 Model 530 as a NIM Master and Server to install a J30 SMP system over a token-ring network.

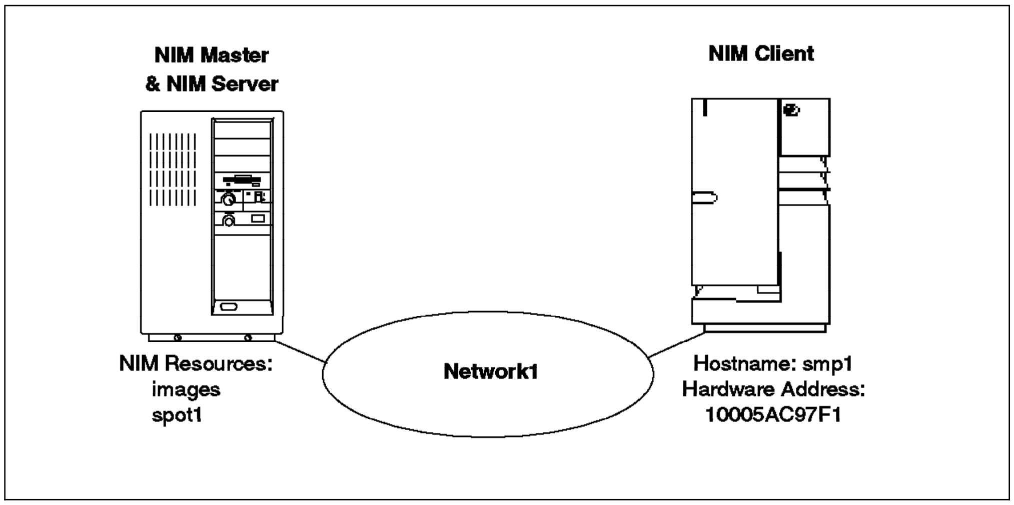


Figure 120. NIM Setup

1. Install the NIM software on the NIM Master. Use the following SMIT fastpath to load the NIM software, which is part of the bos.sysmgt package:

   ```
   # smitty install_latest
   ```

   Install the following filesets:

bos.sysmgt.nim.client	NIM client tools
bos.sysmgt.nim.master	NIM master tools
bos.sysmgt.nim.spot	SPOT creation tools

   SPOT is the Shared Product Object Tree which provides the */usr* filesystem for diskless and dataless clients as well as the network boot resources for all client configurations.

2. Then create the /export filesystem as follows:

   ```
   # crfs -v jfs -g rootvg -a size=8192 -m /export -A yes
   # mount /export
   # mkdir /export/nim
   # mkdir /export/nim/scripts
   ```

3. Use the `nimconfig` command to specify the name of the network (Network1 in our example) and the default port (1058 in our example):

   ```
   # nimconfig -a netname=Network1 -a pif_name=tr0 \
   -a master_port=1058 -a ring_speed=16
   ```

   The SMIT fastpath is: `smitty nimconfig`

   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
   ```

   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
   ```

   This tells that you need a SPOT and an lpp_source resource.

4. To create the lpp_source resource, create a separate filesystem into which to load software images and mount it:

   ```
   # crfs -v jfs -g rootvg -a size=614400 -m /lpp_images -A yes
   # mount /lpp_images
   ```

   Then load the tape in the tape drive and define and create the NIM lpp_source resource (this can take up to an hour to complete):

   ```
   # nim -o define -t lpp_source -a location=/lpp_images \
   -a server=master -a source=/dev/rmt0.1 images
   ```

   The SMIT fastpath is: `smitty nim_mkres`

5. To create the SPOT, create a separate filesystem for the network boot images and mount it as `/tftpboot`:

   ```
   # crfs -v jfs -g rootvg -a size=57334 -m /tftpboot -A yes
   # mount /tftpboot
   ```

   Then define the /usr SPOT named spot1 in our example. Since we just want to install AIX onto the SMP system, convert the /usr filesystem on your master into a SPOT to save disk space (this can also take up to an hour to complete):

   ```
   # nim -o define -t spot -a location=/usr -a server=master \
   -a source=images spot1
   ```

   The SMIT fastpath is: `smitty nim_mkres`

   The NIM master is now set up. Now you need to begin the definition of the clients and the resources required to install them.

6. Define your client (SMP) system to NIM (in our example the SMP hostname is smp1 and the NIM object name is nim_client):

   ```
   # nim -o define -t standalone -a platform=rs6ksmp \
   -a if1='Network1 smp1 10005AC97CF1' -a ring_speed=16 nim_client
   ```

   The SMIT fastpath is: `smitty nim_mkmac`

7. Use the following command to allocate the lpp_source resource, images (which points to /lpp_source) and the SPOT resource, spot1, to the NIM machine object called nim_client.

   ```
   # nim -o allocate -a lpp_source=images -a spot=spot1 nim_client
   ```

   The SMIT fastpath is: `smitty nim_alloc`

8. Initiate the installation of the client.

   The NIM resources are exported to the client system. An attempt is made to contact the client system to perform a reboot, but this just times out because our SMP system is not running AIX V4.1 and is not yet configured as a NIM client.

   ```
   # nim -o bos_inst -a source=rte nim_client
   ```

   The SMIT fastpath is: `smitty nim_mac_op`

9. Manual intervention is required at the client to initiate the network boot. In order to do this, you need to bring up the SystemGuard Maintenance Menu. For information on how to get into the SystemGuard Maintenance Menu, refer to Chapter 5, “SystemGuard” on page 99.

   From the SystemGuard Maintenance Menu, take the following options:

   a. Select **System boot**.

b. Select **Boot From Network**.

c. Select the **Select BOOT (Startup) Device** option.

d. Select the network (in our case token-ring at 16 Mb).

e. Enter IP addresses for client and bootp server and then option 99 to return to the Main Menu.

f. Test the connection by using the **Send Test Transmission (PING)** option.

g. Start the Network Install.

h. Answer the questions as per the Complete Overwrite Install to choose languages. Refer to 7.2.3, "New and Complete Overwrite Installation" on page 177 for more information.

The above example was used to pull a new installation from a server. Once the system is installed with AIX V4.1 and a configured NIM client, it is possible to do a push install of operating-system updates or even a reinstallation of the operating system. The network installation can be totally automated using a modified bosinst.data file and a customization script. For further information, please refer to the *AIX Version 4.1 Network Installation Management Guide and Reference*.

7.6 AIX V4.1 Software Maintenance

In AIX V4.1, the software maintenance levels are now identified by Version Release Modification Fix (VRMF) levels. The VRMF levels will be displayed by using the `lslpp` command. You will no longer see any (PTF) Program Temporary Fix (U4xxxxx) numbers. For example, the AIX V4.1.2 bos.rte.up fileset has a VRMF of 4.1.2.0, indicating Version 4, Release 1, Modification level 2 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 V4.1 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 V4.1 will come 3-6 months apart.

The **Fix** level is incremented whenever a fix is added to the fileset. Fixes for AIX V4.1 are created on customer demand.

Each fileset in AIX V4.1 can be serviced separately. Fixes will be 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 V4.1.

7.6.1 Fileset VRMF Numbering

The following diagram is used to show that the various filesets will show different VRMF for a particular level of AIX V4.1. For AIX V4.1.2, only some of the filesets will be at level 4.1.2.0.

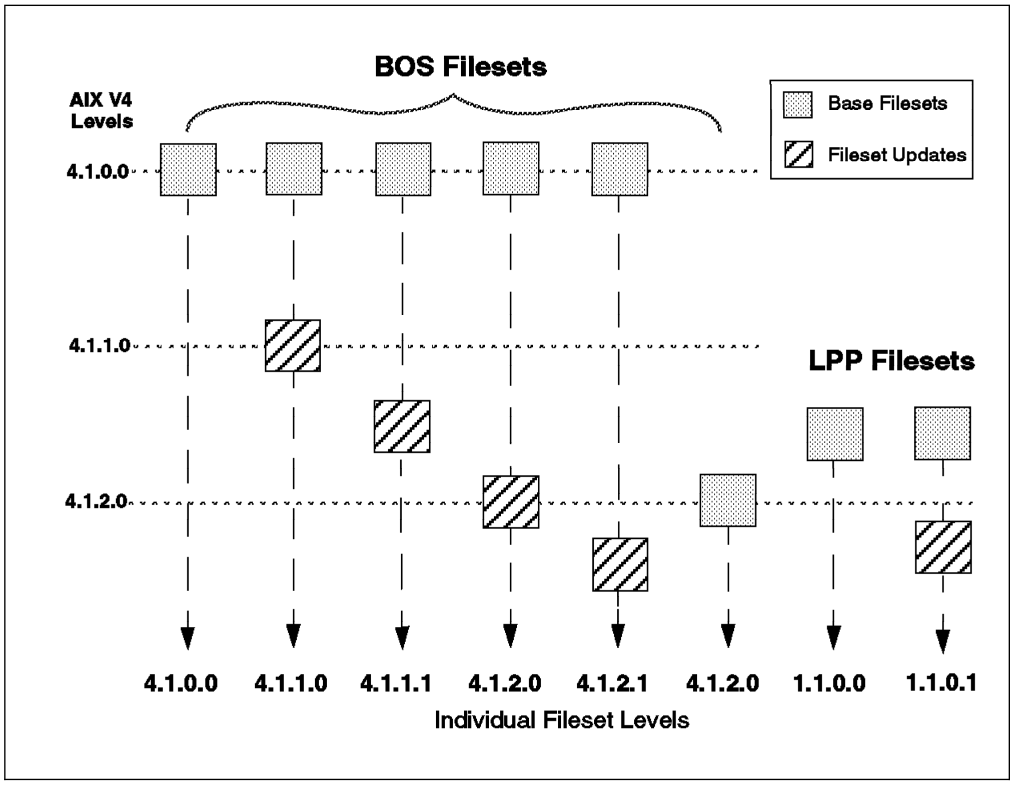


Figure 121. Fileset Numbering Examples

If the fileset for the base operating system was shipped in 4.1.0 and if there were absolutely no code changes to that fileset since 4.1.0, then, the VRMF is unchanged. For example, the level of bos.dlc.com for AIX V4.1.2 is 4.1.0.0.

If the fileset for the base operating system was shipped in 4.1.0 and if there were changes to the code for 4.1.1 or 4.1.2, the modification level for the VRMF will be adjusted to be inline with the operating-system level. For example, the level of bos.rte.up was 4.1.2.0 for AIX V4.1.2 and 4.1.1.0 for AIX V4.1.1. The content of the fileset was modified for both levels of AIX V4.1.

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 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.

7.6.2 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
4.1.2.0
```

To list the names of the known Maintenance Levels, enter:

```
# oslevel -q
Known Maintenance Levels
------------------------
4.1.2.0
4.1.1.0
#
```

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 = "4.1.1.0_AIX_ML"
        abstract = "AIX V4.1.1.0 Maintenance Level"
        type = "p"
        filesets = "bos.acct:4.1.1.0\n\
bos.adt.base:4.1.1.0\n\
bos.adt.debug:4.1.1.0\n\
bos.adt.graphics:4.1.1.0\n\
bos.adt.include:4.1.1.0\n\
 ...

fix:
        name = "4.1.2.0_AIX_ML"
        abstract = "AIX V4.1.2.0 Maintenance Level"
        type = "p"
        filesets = "bos.acct:4.1.2.0\n\
bos.adt.base:4.1.2.0\n\
bos.adt.debug:4.1.2.0\n\
bos.adt.graphics:4.1.1.0\n\
bos.adt.include:4.1.2.0\n\
 ...
```

7.6.3 instfix Command

There is a new `instfix` command with AIX V4.1, and there are a number of options to this command. It can be used 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"
```

7.6.4 Applying Updates

There are a number of SMIT fastpaths to enable you to install maintenance to your AIX V4.1 system.

`smitty update_all` will install all the fixes that are available on the installation media.

`smitty install_maintenance` allows the system administrator to install a later maintenance level.

`smitty install_fileset` allows the system administrator to install individual fixes. Use / to search for the fix and PF7 to select multiple filesets.

Chapter 8. SMP Performance Tools

Tuning is an integral requirement for any system. There are a variety of tools included with AIX V4.1 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.

8.1 AIX V4.1 Performance Tools Considerations

The AIX V4.1 performance tools can be categorized as following: tools that are totally new in AIX V4.1, tools that are SMP specific, standard tools or commands that were available in AIX V3.2 and have been modified to support the SMP environment, and tools that were included in AIX V3.2 but are now part of an LPP.

Following are tools that did not exist before in AIX V3.2 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 4.1. 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 tunnelling encapsulation microscope): allows insertion of instrumentation code at the entry and exit points of existing programs and library subroutines.

Following are commands that 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 unavailable. This command is part of the bos.rte.mp fileset.
- The `bindprocessor` command is used to distribute the workload on the system. It enables the binding of individual processes to a processor within the SMP system. This command is also part of the bos.rte.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 V3.2, 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.

  Note: The `vmstat`, `iostat` and `sar` commands are part of the bos.acct fileset in AIX V4.1.

The following AIX performance tools were available with AIX V3.2 and have not changed in that they still show either global system performance or process performance. However, these tools were previously included in AIX V3.2. 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 commands, such as `filemon`, `fileplace`, `lvedit`, `netpmon`, `rmss`, `stripnm`, `svmon`, `tprof`, and `lockstat`. This LPP is required to use Performance Toolbox which will be discussed in 8.9, “Monitoring your SMP with Performance Toolbox” on page 226.

- `filemon`: Monitors the performance of the filesystem 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`: The logical volume editor is 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.

8.2 Processes and Threads Status

The ps command is one of the most useful commands when 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 processes and threads 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 fields 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.

Following are some output examples 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 output that the `4everunbound` process is active. You can see it 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-thread process.

Note that, in this example, no threads are bound to a specific processor (- in the `BND` column).

Following 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 in the `BND` column instead of the `-`.

8.3 Binding a Process

AIX V4.1 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. The process to be bound must be 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 context 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.

But 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 V4.1 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 V4.1, 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 by using :

```
# sar -P ALL 1 <n>
```

where `<n>` is the number of seconds to run the `sar` command.

Bind the process to that processor 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
```

8.4 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 using the `bindprocessor()` call. The `bindprocessor()` call must be used in the source code of your program.

If the `bindprocessor()` call is used within a piece of code to bind threads to processors, the threads will stay with these processors and cannot be unbound individually.

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 bind 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.

8.5 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 will reference 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 245. 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 will help us to illustrate the use of the standard performance tools on the SMP system.

8.5.1 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.

8.5.2 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 processors one at a time.

8.5.3 Threads-Related Information

The `pstat` command is the non-interactive 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`: shows thread slot user structure of kernel threads
- `-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`: is the process ID to which the thread belongs.
- `CPUID`: is 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`: is the thread's scheduling policy.
- `PRI`: is 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 4.1, 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 V4.1, this policy is defined to be equivalent to RR, except that it applies to threads with non-fixed 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 V4.1.

8.5.4 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, character read, write, 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 line gives the average of all averages.

The `-P` flag, when specified, is equivalent to `-acmuw`. When the `-P` flag is not specified, this is equivalent to specifying `-abckmqruvwy`.

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.

8.5.5 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 has generally been used for an overall look at resource utilization while running a multiuser workload.

The vmstat output yields the kthr column which 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.

8.6 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 number of logical processors numbering will change after the reboot.

After 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
```

8.7 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 V4.1 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 V4.1 defines subsystems comprised of 256 lock classes in /usr/include/sys/lockname.h.

However, it is the developer's responsibility to define and implement an appropriate locking strategy to protect the program's own data.

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, one 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 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 to:
  - `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 V4.1.2, 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 V4.1 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 which 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 4.1, this possibility can be investigated with the `lockstat` command;

Look for the following pointers:

- Check `lockstat` output for `Ref/s` > 10 000
- 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.

8.8 Tuning Guidelines

This is not a detailed description on how to tune an SMP system. The intent here is to give some outlines that may be useful when working with an SMP system to improve the performance of the system or to at least identify the location of the bottleneck.

As far as tuning is concerned, the main difference between an SMP and a UP is the number of processors. Since processes and threads can run on any processor, you might look at the CPU usage on all the processors and discover that one or several processors are 100 percent busy, while others are not very loaded.

Since memory, disks and network adapters are shared between all the processors, the tuning methodology is very similar to a UP system when the system is not CPU bound. Load balancing between the processors is probably the main difference between an SMP and a UP.

A method that can be employed for tuning your SMP system in case of a performance problem is the following:

- Check the availability of the processors using the `cpu_state` command:

  ```
  # cpu_state -l
  ```

- Are all the processors available? One disabled processor can be the cause of your performance problem. This is, of course, an obvious reason.
- Check the balance of workload between processors using the `sar` command. See 8.5.4, “Measuring the Processors Load” on page 220 for more information on the use and options of the `sar` command.

  ```
  # sar -P ALL 1 <n>
  ```

- Determine if some processes or threads are bound. This can hurt the performance of your application, especially if your application is multithreaded and bound to one processor. Refer to 8.2, “Processes and Threads Status” on page 211 and to 8.5.3, “Threads-Related Information” on page 218 for more information on the use of the `ps` command options and the `pstat` command.

  ```
  # ps -m -o THREAD
  # pstat -A
  ```

- Unbind any bound processes that hurt your system performance with the `bindprocessor` command.

  ```
  # bindprocessor -u <pid>
  ```

- If your system is still CPU bound even with all processors available, there are several solutions:
  - Parallelize your application using mutiple threads or multiple processes. This requires some programming skills and time.
  - Add more processors. This will help if your application is already parallelized.
  - Change processor technology (use faster processors).
- If your system is not CPU bound and still has some performance problems, the procedure is very similar to a UP. The bottleneck can be the memory, the I/O subsystem or the network. In this case, use the regular performance tools to determine the bottleneck's location.
- If you did not find any bottleneck, you might want to check is you have any lock contention on your system.

8.9 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.

8.9.1 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 the presentation program that displays the required output.
- Performance Aide 2.1 for AIX V4.1 (program number 5696-899): This product must run on any system that is to be monitored.

The following terms are used to refer to PTX 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 PTX 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 PTX, you need a graphical display if the SMP is a G30. If the SMP is a J30 or an R30, you need an Xstation. You can also monitor any SMP system from a graphical workstation.

You need to install the following filesets:

- perfagent.server
- perfagent.tools
- perfmgr

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. Therefore, `xmperf` and `3dmon` are part of this fileset.

If you monitor your SMP from an Xstation, 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
```

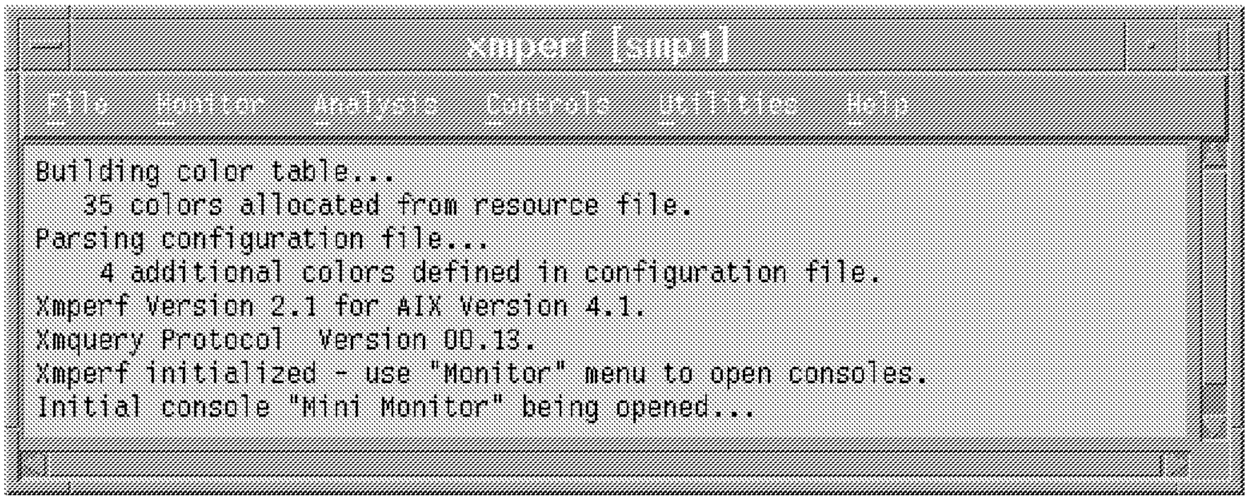
If you want to monitor your SMP from a remote host (a graphical workstation), enter the following command:

```
# xmperf -h <hostname>
```

8.9.2 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).

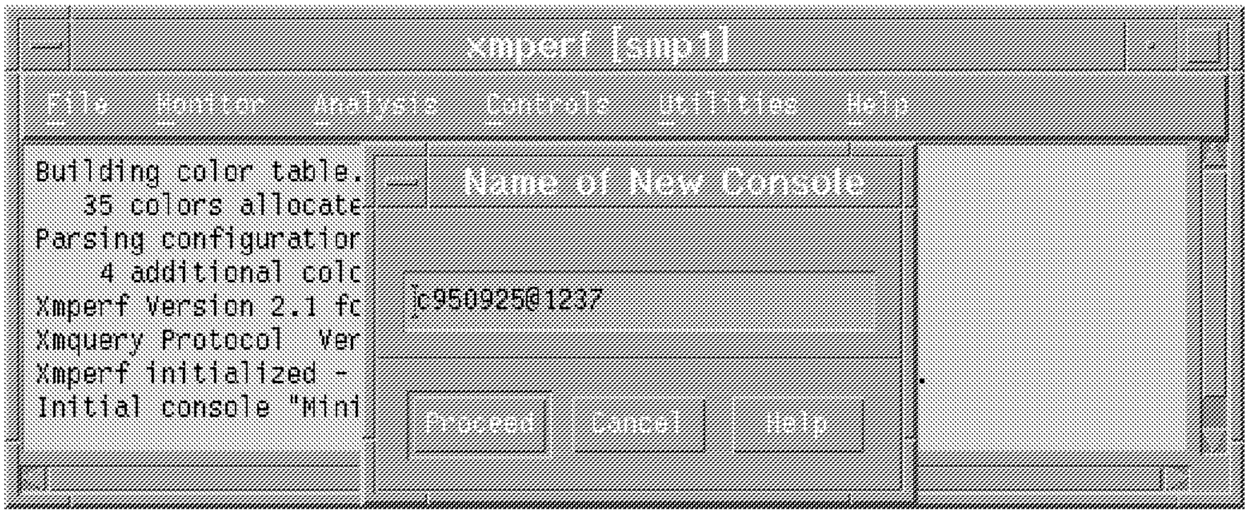
Start PTX by using `xmperf` from the command line. You will then be presented with an intial window that looks like Figure 122.



```
Building color table...
   35 colors allocated from resource file.
Parsing configuration file...
    4 additional colors defined in configuration file.
Xmperf Version 2.1 for AIX Version 4.1.
Xmquery Protocol  Version 00.13.
Xmperf initialized - use "Monitor" menu to open consoles.
Initial console "Mini Monitor" being opened...
```

Figure 122. xmperf Initial Screen

Use the Monitor pull-down menu to select **Add New Console**. You will then get a sub-window inviting you to enter a console name. Give a meaningful name to your console instead of using the unique, default name.



```
c950925@1237
```

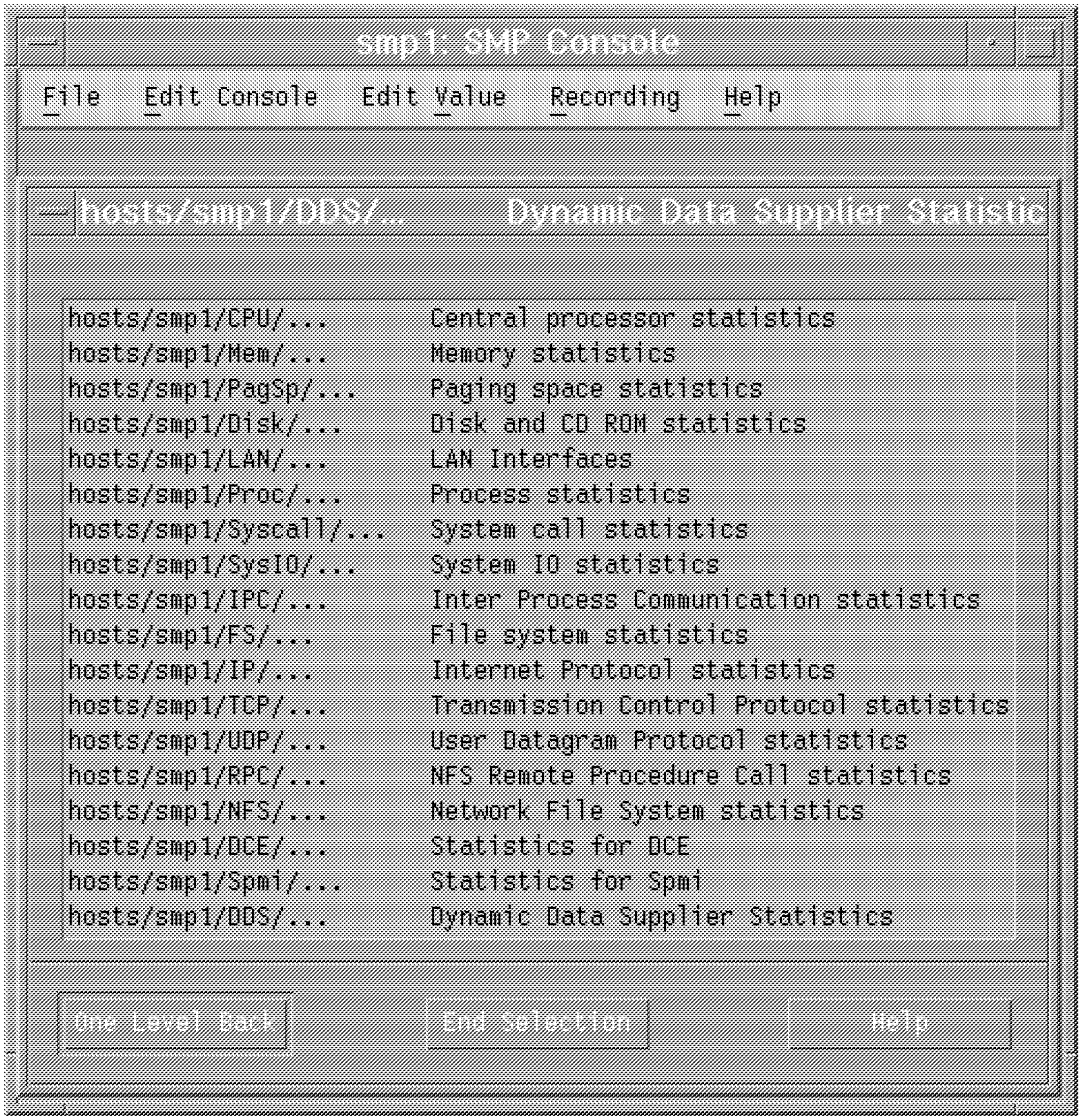
Figure 123. Creating a New Console

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 `xmservd` running. 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.

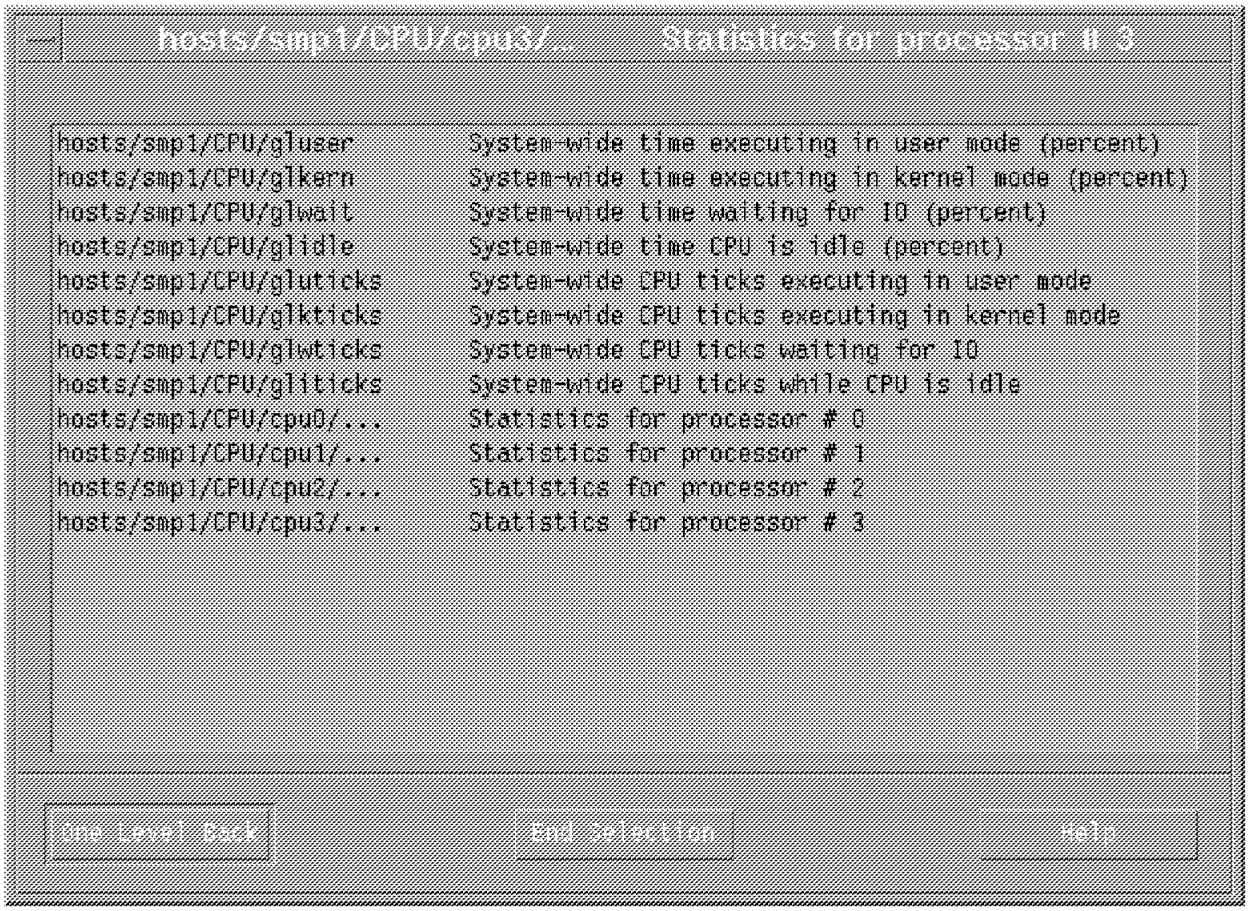


```
hosts/smp1/CPU/...       Central processor statistics
hosts/smp1/Mem/...       Memory statistics
hosts/smp1/PagSp/...     Paging space statistics
hosts/smp1/Disk/...      Disk and CD ROM statistics
hosts/smp1/LAN/...       LAN Interfaces
hosts/smp1/Proc/...      Process statistics
hosts/smp1/Syscall/...   System call statistics
hosts/smp1/SysIO/...     System IO statistics
hosts/smp1/IPC/...       Inter Process Communication statistics
hosts/smp1/FS/...        File system statistics
hosts/smp1/IP/...        Internet Protocol statistics
hosts/smp1/TCP/...       Transmission Control Protocol statistics
hosts/smp1/UDP/...       User Datagram Protocol statistics
hosts/smp1/RPC/...       NFS Remote Procedure Call statistics
hosts/smp1/NFS/...       Network File System statistics
hosts/smp1/DCE/...       Statistics for DCE
hosts/smp1/Spmi/...      Statistics for Spmi
hosts/smp1/DDS/...       Dynamic Data Supplier Statistics
```

Figure 124. Selecting Central Processor Statistics

For the sake of our example, we want to monitor CPU statistics for our SMP; so we selected **Central Processor Statistics**

You will then get a new window which looks like Figure 125.

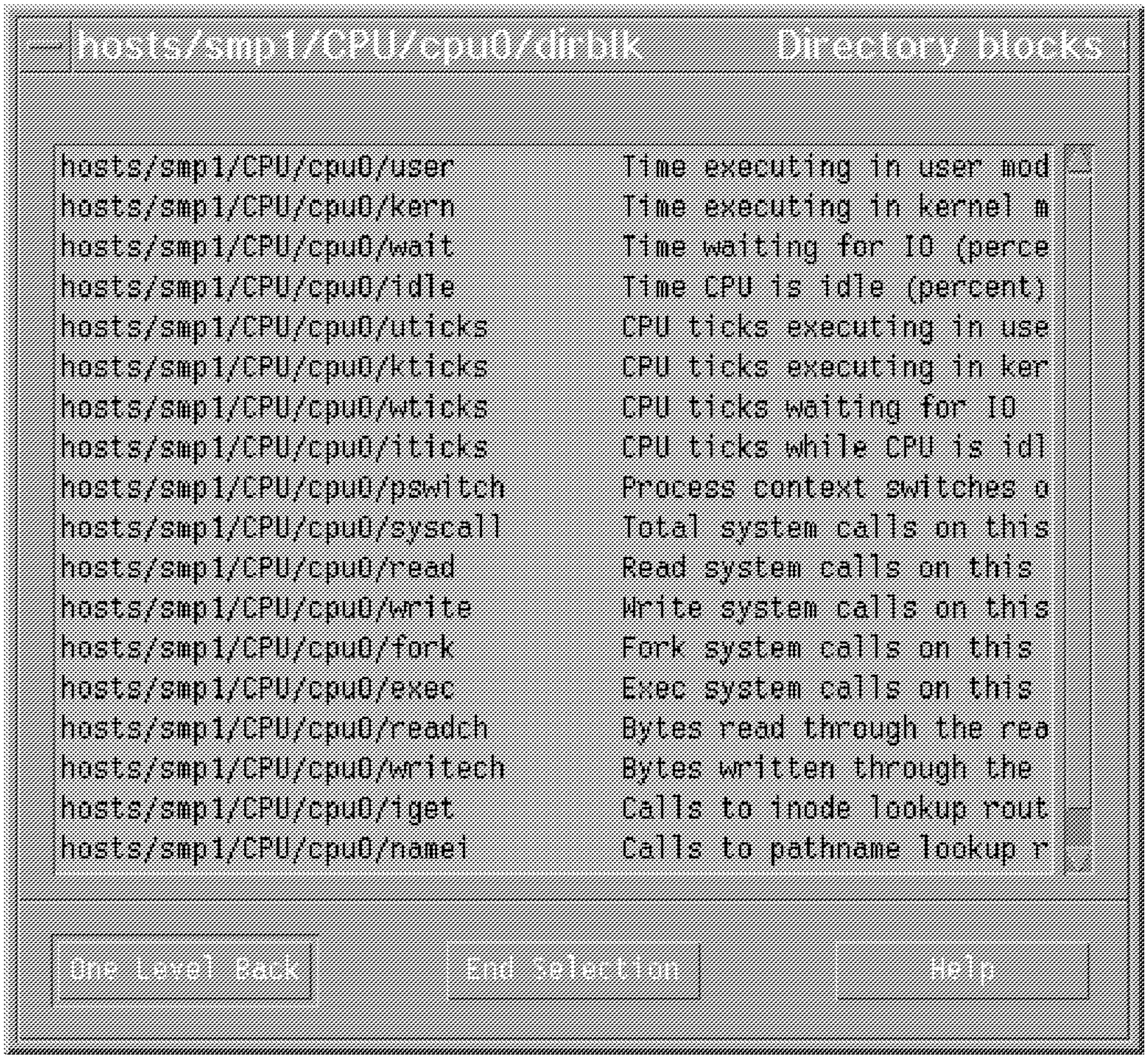


hosts/smp1/CPU/cpu3/...   Statistics for processor # 3

```
hosts/smp1/CPU/gluser       System-wide time executing in user mode (percent)
hosts/smp1/CPU/glkern       System-wide time executing in kernel mode (percent)
hosts/smp1/CPU/glwait       System-wide time waiting for IO (percent)
hosts/smp1/CPU/glidle       System-wide time CPU is idle (percent)
hosts/smp1/CPU/gluticks     System-wide CPU ticks executing in user mode
hosts/smp1/CPU/glkticks     System-wide CPU ticks executing in kernel mode
hosts/smp1/CPU/glwticks     System-wide CPU ticks waiting for IO
hosts/smp1/CPU/gliticks     System-wide CPU ticks while CPU is idle
hosts/smp1/CPU/cpu0/...     Statistics for processor # 0
hosts/smp1/CPU/cpu1/...     Statistics for processor # 1
hosts/smp1/CPU/cpu2/...     Statistics for processor # 2
hosts/smp1/CPU/cpu3/...     Statistics for processor # 3
```

Figure 125. Selecting Statistics

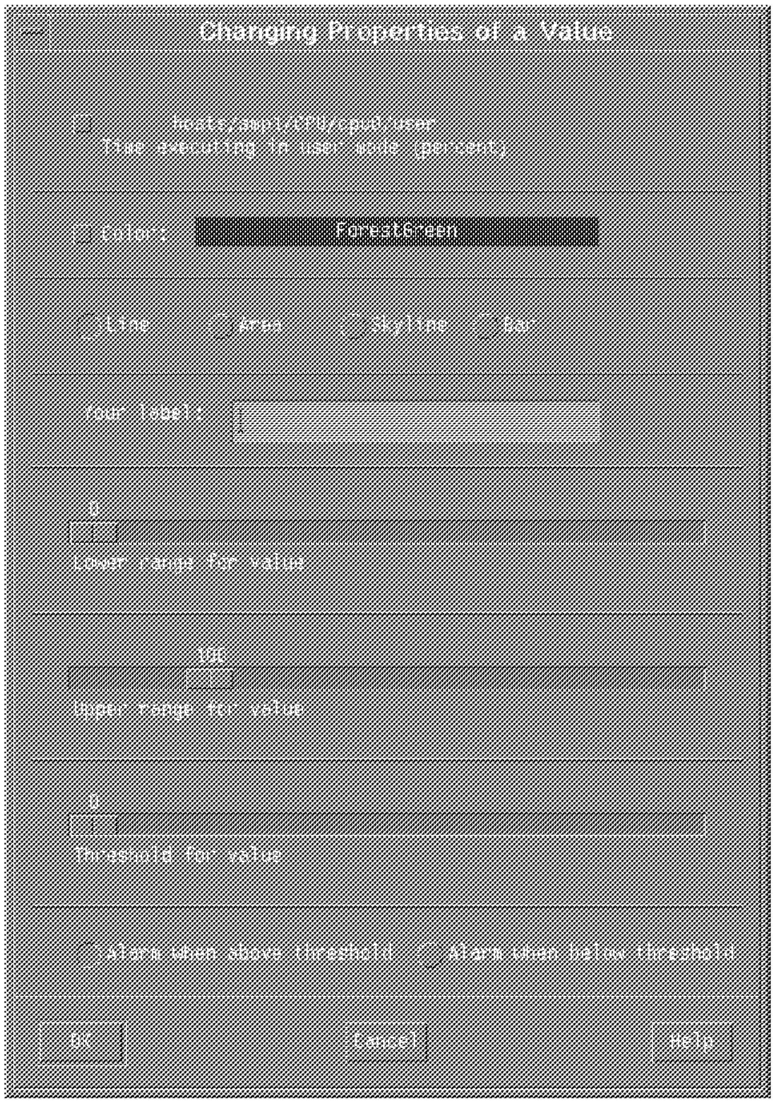
At this point, you can choose global statistics or statistics for a specific processor. You cannot do multiple selections at the same time. You can choose for example, Statistics for processor # 0. You will then get the following screen:



```
hosts/smp1/CPU/cpu0/user         Time executing in user mod
hosts/smp1/CPU/cpu0/kern         Time executing in kernel m
hosts/smp1/CPU/cpu0/wait         Time waiting for IO (perce
hosts/smp1/CPU/cpu0/idle         Time CPU is idle (percent)
hosts/smp1/CPU/cpu0/uticks       CPU ticks executing in use
hosts/smp1/CPU/cpu0/kticks       CPU ticks executing in ker
hosts/smp1/CPU/cpu0/wticks       CPU ticks waiting for IO
hosts/smp1/CPU/cpu0/iticks       CPU ticks while CPU is idl
hosts/smp1/CPU/cpu0/pswitch      Process context switches o
hosts/smp1/CPU/cpu0/syscall      Total system calls on this
hosts/smp1/CPU/cpu0/read         Read system calls on this
hosts/smp1/CPU/cpu0/write        Write system calls on this
hosts/smp1/CPU/cpu0/fork         Fork system calls on this
hosts/smp1/CPU/cpu0/exec         Exec system calls on this
hosts/smp1/CPU/cpu0/readch       Bytes read through the rea
hosts/smp1/CPU/cpu0/writech      Bytes written through the
hosts/smp1/CPU/cpu0/iget         Calls to inode lookup rout
hosts/smp1/CPU/cpu0/namei        Calls to pathname lookup r
```

Figure 126. 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 127 on page 232 shows how to customize the properties of the values you want for your instrument.


Changing Properties of a Value

hosts/smp1/CPU/cpu0/user
Time executing in user mode (percent)

Color: ForestGreen

Line Area Skyline Bar

Your label:

0
Lower range for value

100
Upper range for value

0
Threshold for value

Alarm when above threshold Alarm when below threshold

OK Cancel Help

Figure 127. 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 at any time your console by selecting the File pulldown 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 using the Monitor pull-down menu.

Figure 128 on page 233 shows an example of a customized SMP console.

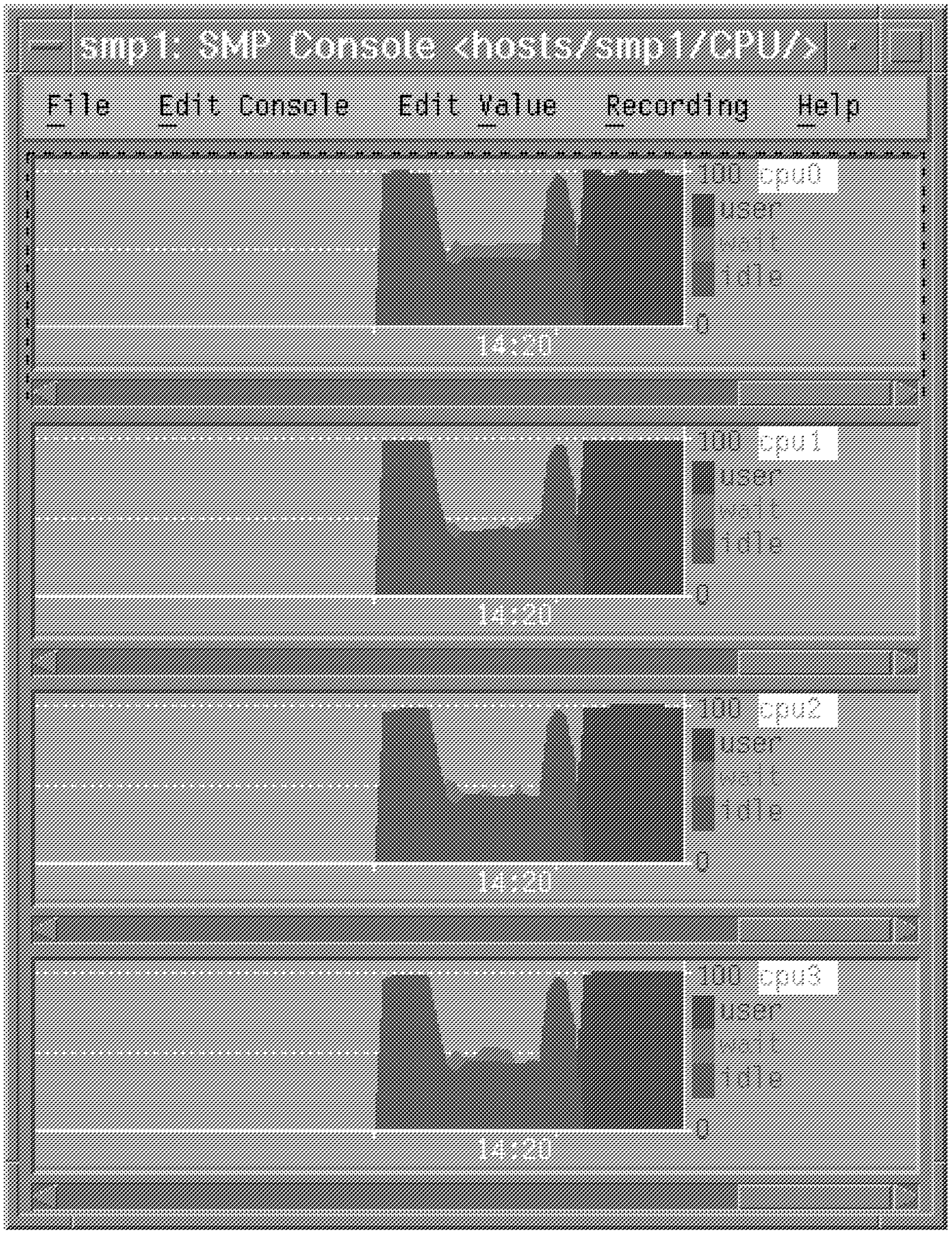


Figure 128. SMP Console Example

8.9.3 Monitoring an SMP with 3dmon

`3dmon` provides a quick method of producing the same results in a three-dimensional view for important performance values.

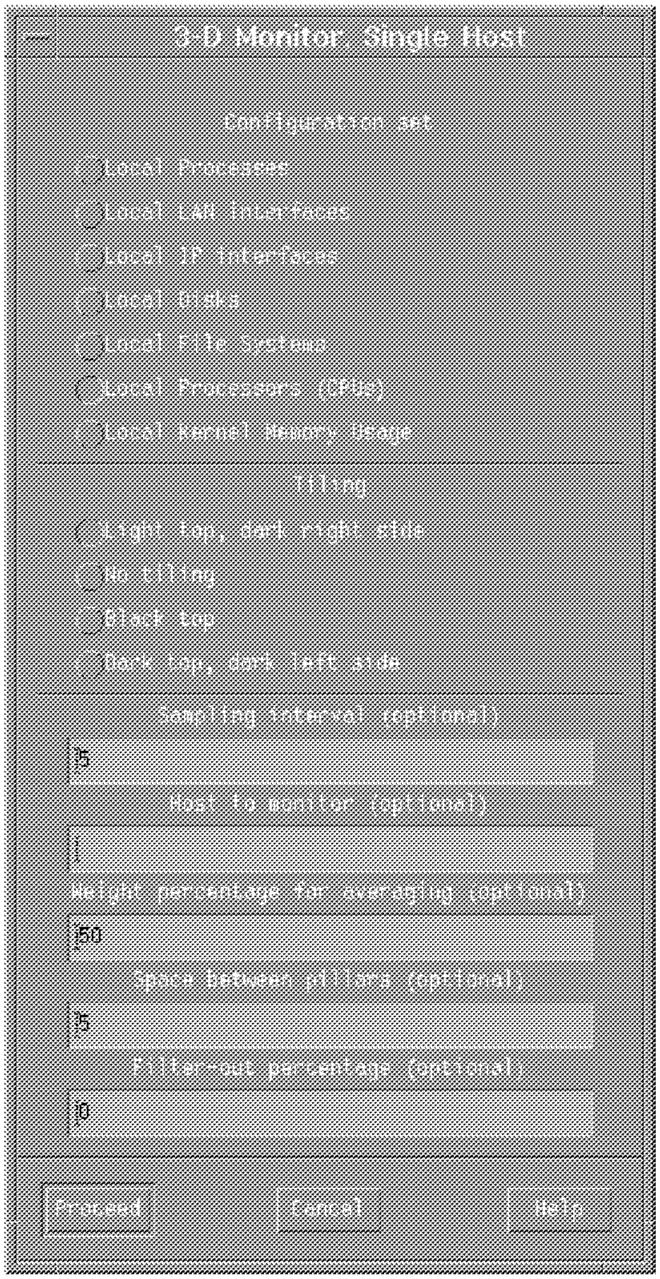
This monitor may be invoked by going to the utilities menu in the main `xmperf` window. Inside this menu, you will find both the 3d Monitor Local and 3d Monitor

Remote sub-menus. 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`.

`3dmon` 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:



Sampling interval (optional)
5
Host to monitor (optional)

Weight percentage for averaging (optional)
50
Space between pillars (optional)
5
Filter-out percentage (optional)
0

Figure 129. 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:

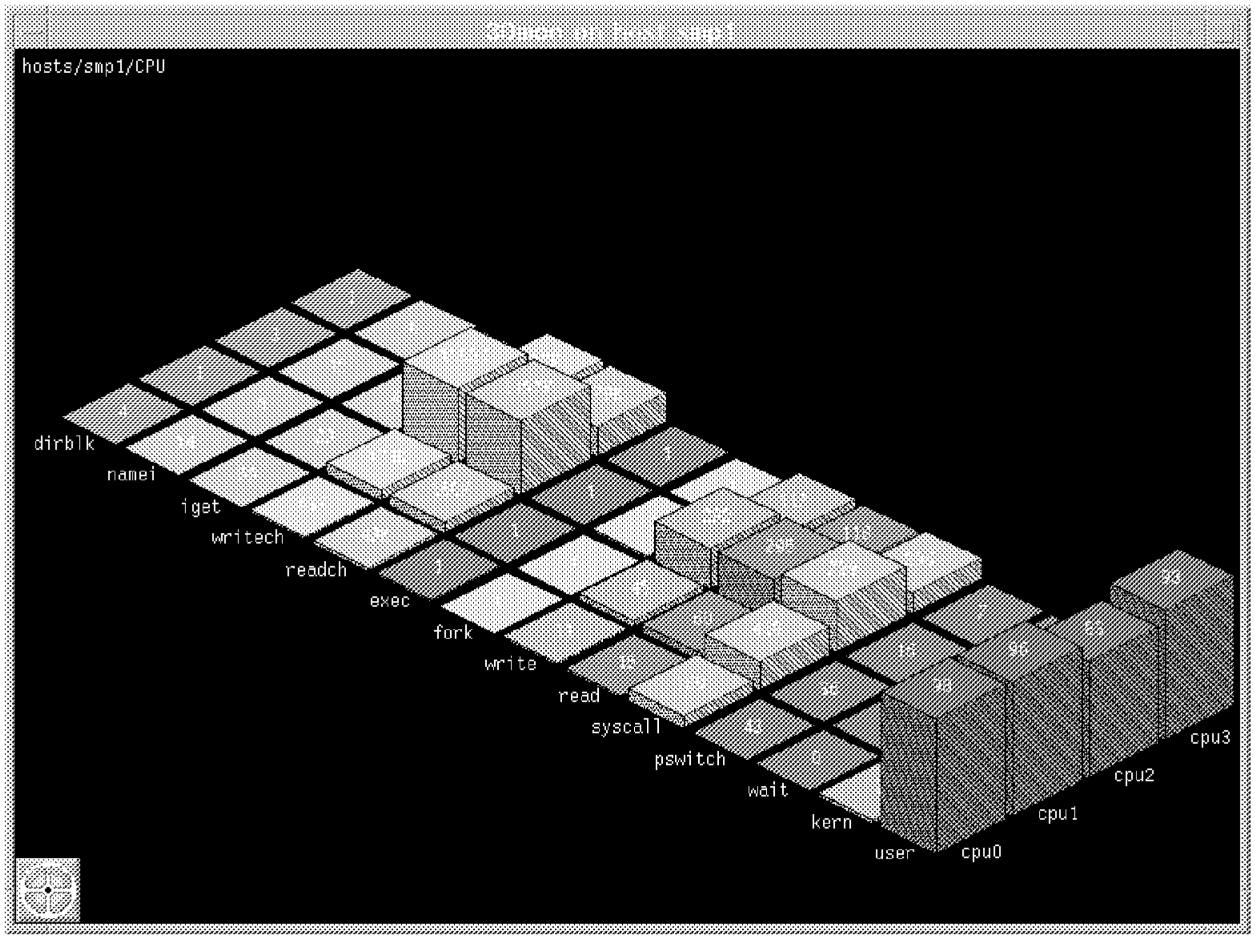


Figure 130. 3dmon Output on a Four-way SMP

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.

Appendix A. SystemGuard Remote Operation Configuration

In order to utilize the remote operation capabilities of SystemGuard and to also allow console mirorring, you need to have flags, parameters and TTY configurations properly enabled. 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.

```
[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                                   disable
  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]
```

This is the tty1 configuration for the S2 port.

```
[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                                   disable
  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                       [rts]
  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]
```

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/7851
2       Service Line Speed                        9600
3       Protocol Inter Data Block Delay           5
4       Protocol Time Out                         60
5       Retry Number                              2
6       Customer ID
7       Login ID
8       Password ID
```

The `Modem Parameter File Name` value should be set to the file name of your modem configuration file. The service line speed should be set to your modem and tty capabilities (9600 is recommended).

- 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
```

- Diagnostics flags:

```
mpcfg -df

Index   Name                                    Value
1       Remote Authorization                    1
2       Autoservice IPL                         0
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
```

- Phone numbers:

```
mpcfg -dp

Index   Name                                    Value

1       Service Center Dial-Out (1)             18008301041
2       Service Center Dial-Out (2)
3       Customer Hub Dial-Out (1)
4       Customer Hub Dial-Out (2)
5       System Dial-In
6       System Operator Voice
```

The phone number in the `Service Center Dial-Out` field represents the U.S IBM RETAIN number. It should be set as appropriate to the geography. Other phone numbers should be provided based on account-related information.

A.3 Modem Configuration Files

If you want to attach a modem to the S2 port to allow automatic problem reporting from SystemGuard or dial-in access from a remote location, 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.

Two modems have been tested, the IBM 7851 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 model of modem. If you do not use any modem for connecting the Service Console, you will need a modem file. An example of the modem file without a modem is provided below.

This is a sample /usr/share/modems/mir_modem file for console mirroring without using modems.

```
ICDelay 1
DefaultTO 10
condout: done
connect: done
retry:   done
disconnect:    done
condin:  done
condwait:      done
waitcall:      done
page:    done
```

This is a sample /usr/share/modems/mir_modem file for console mirroring using an IBM 7851 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
```

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
```

A.4 Initializing a Modem

Once flags, parameters and configurations have been enabled, the modem can be initialized to accept incoming calls. This can be done in the following manner:

- Place the System Key to Normal.
- Issue a `ps -ef|grep mirrord` command.
- Obtain mirrord process ID.
- Issue a `kill -9 <mirrord_pid>`.
- Issue `/usr/sbin/mirrord <modem_filename>`.
- Place the System Key into the Service position.

This will initiate the mirrord process. The `disconnect` and `condin` parameters will be read from the appropriate modem file, and the modem will be initialized for dial-in activity. After the mirrord daemon is activated, the System Key should be placed in the Normal position.

A.5 Testing Dial-Out

Dial-out or automatic problem reporting can be tested from the SystemGuard Maintenance Menu using the Offline Test for Dial-out. Successful connection and transmission of data will result in an OK completion.

During the test, the modem configuration file will be read, the modem initialized properly and data transmitted. Comparing the modem configuration file to line activity will show disconnect, condout, connect, transmit of data, disconnect and condin. Thus, the modem is initialized for dial-out, data is transmitted, and the modem is initialized to allow dial-in.

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
```

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
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
PReP	PowerPC Reference Platform
PLL	Phase Lock Loop
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
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

ITSO Technical Bulletin Evaluation
RED000

International Technical Support Organization
IBM RISC System/6000 SMP Servers
Architecture and Implementation
November 1995

Publication No. SG24-2583-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

____

____

What other topics would you like to see in this Bulletin?

____

____

What other Technical Bulletins would you like to see published?

____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

______________________ ______________________
Name Address

______________________ ______________________
Company or Organization

______________________ ______________________
Phone No.

ITSO Technical Bulletin Evaluation **RED000**

SG24-2583-00

IBM®

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 948, Building 821
Internal Zip 2834
11400 BURNET ROAD
AUSTIN TX
USA 78758-3493

Fold and Tape **Please do not staple** Fold and Tape

SG24-2583-00

Cut or Fold
Along Line