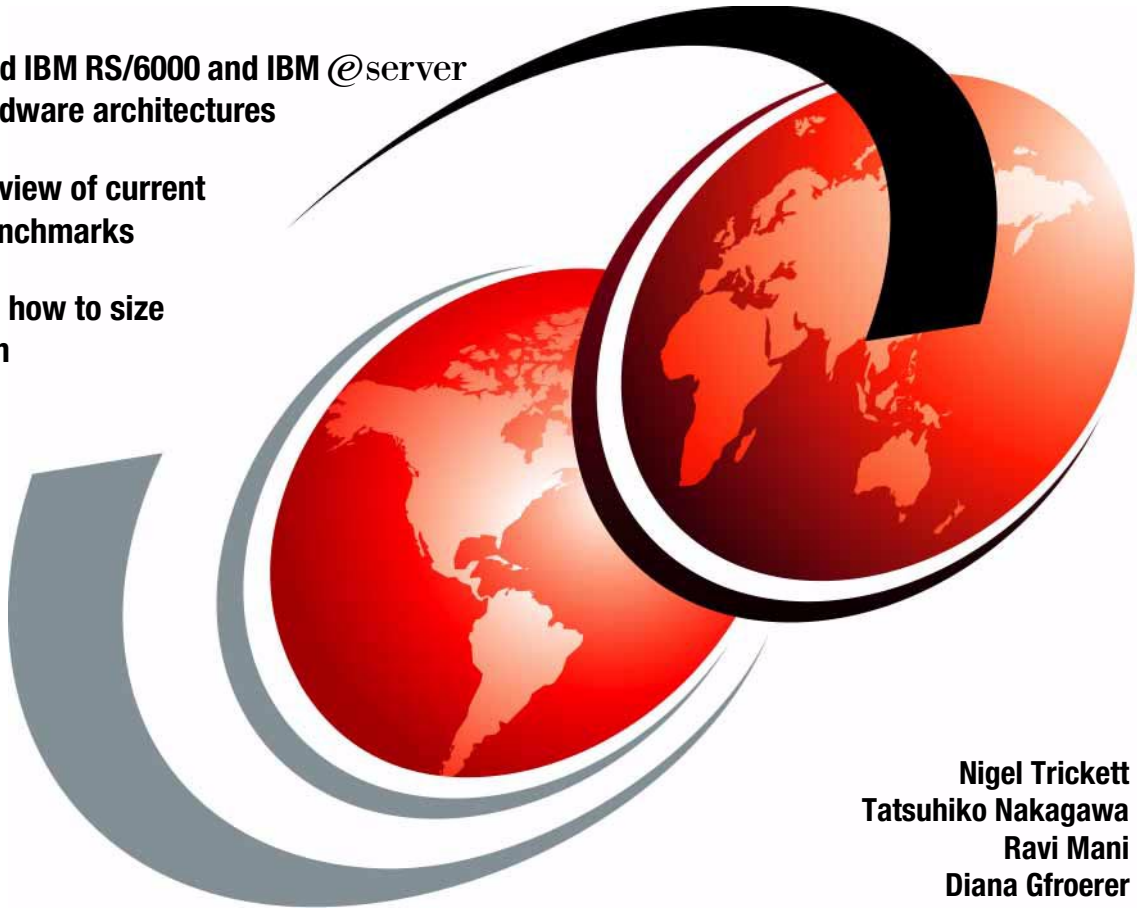
IBM

Understanding IBM @server pSeries Performance and Sizing

- Comprehend IBM RS/6000 and IBM @server pSeries hardware architectures
- Get an overview of current industry benchmarks
- Understand how to size your system



Nigel Trickett
Tatsuhiko Nakagawa
Ravi Mani
Diana Gfroerer

Redbooks

ibm.com/redbooks

SG24-4810-01

International Technical Support Organization

Understanding IBM @server pSeries Performance and Sizing

February 2001

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix A, “Special notices” on page 377.

Second Edition (February 2001)

This edition applies to IBM RS/6000 and IBM @server pSeries as of December 2000, and Version 4.3.3 of the AIX operating system.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

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

Contents

Preface

Contained in this redbook is a close-up, performance related view of the different hardware architectures IBM offers in its RS/6000 and @server pSeries systems, including system, processor, memory, storage, and network architectures. One chapter is dedicated to general sizing rules for a number of environments such as database sizing, IBM HTTP server sizing, Net.Commerce sizing, and Lotus Domino sizing. The reader will also find a description of the Industry benchmarks that are performed on IBM systems as well as an overview on AIX performance tools.

This redbook is an update to the successful first Edition of “Understanding IBM RS/6000 Performance and Sizing,” that was published in 1997. Obsolete information was taken out, still relevant information was updated, and new information was added to this new Edition. While the book was produced, IBM RS/6000 was re-branded to IBM @server pSeries. As a conclusion to that, the given information applies to both brands, even though only one of the brand names might be mentioned, except for the processor and system architectures, which only apply in part to the IBM @server pSeries models.

The team that wrote this redbook

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

Diana Gfroerer is an International Technical Support Specialist for AIX Performance at the International Technical Support Organization Austin Center. She writes extensively and teaches IBM classes worldwide on all areas of AIX Performance and Tuning. Before joining the ITSO in 1999, Diana Gfroerer worked in AIX pre-sales Technical Support in Munich, Germany. She was leading the World Wide Technical Skills Community for AIX and PC Interoperability.

Nigel Trickett is a Software Support Specialist at IBM New Zealand. He joined IBM in 1995, working on software calls providing both onsite and telephone support. He has worked with Unix since 1984 and has had several roles since then, including hardware and software support and systems administration. Nigel Trickett's primary responsibilities are to resolve performance issues with AIX and to analyze system dumps. He also works on many types of software issues. Nigel holds a New Zealand Certificate of Computer Technology.

Tatsuhiko Nakagawa is an I/T specialist at IBM Japan. He joined IBM in 1991 and has been working with AIX ever since. He has been working with RS/6000 SP since 1996. He has extensive experience in providing solutions for the banking industries. He has written extensively on sizing methods. He holds a B.S. degree in Electrical Engineering from Waseda University, Tokyo, Japan.

Ravi Mani has been working as a Technical Specialist at IBM India Ltd., Bangalore since February, 1996. He has been associated with RS/6000 brand team for more than three years, providing AIX and Hardware support. His responsibilities include benchmarking of various RDBMSs on RS/6000 systems.

Thanks to the following people for their invaluable contributions to this project:

IBM Austin
Matt Accapadi, Bill Brantley, Bill Britton, Chij-Mehn Chang, DaeSung Chung, Richard Cutler, Herman Dierks, Dixin Gu, Hong Hua, Ernest A. Keenan, Warren Maule, John McCalpin, Andy McLaughlin, Augie Mena III, Stephen Nasypany, Lilian Romero, Joe St Clair, Rakesh Sharma, Tina Tsao, Scott Vetter, Nina Vogl-Wilner

IBM Dallas
Roger Leukie, John Tesch

IBM France
Laurent Vanel

IBM Germany
Dr. Hans-Jürgen Kitzhöfer

IBM Japan
Jun Nakano, Eiichi Yamamoto

IBM Pougkeepsie
Ella Buslovich

IBM Rochester
Karl R. Huppler

IBM San Francisco
Dale Martin

IBM San Jose
John Aschoff

IBM Toronto
Don Bourne

IBM UK
Nigel Griffiths, Simon Woodcock

IBM Waltham
Richard Hooker

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 411 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Introduction

Good performance of a system is a relative term because everybody has a different perception of it. It might be defined by:

- The response time for interactive users
- The complete time for batch jobs
- The number of reports finished each day
- The time a system needs for recovery after a failure
- No complaints about poor performance
- The graphical image can be redrawn a certain amount of times every second
- The system never needs changing to fix poor performance

More broadly, a system displays good performance if it meets its performance requirements. Therefore, a clear definition of the performance requirements is mandatory before optimizing system performance, both in business terms (what the users of the system actually see) and technical terms (in quantifiable and measurable numbers).

Good performance is achieved in three phases:

1. The system is adequately sized.
2. The system is initially set-up to yield the maximum performance from the resources available.
3. Regular monitoring and tuning is performed, including upgrading if necessary.

This redbook brings together all the important information that is needed to comprehend IBM RS/6000 and IBM @server pSeries performance and sizing for all three phases.

In this fast changing Information Technology industry, computer design, architecture, configuration rules, and technology are a moving target. It is easy to get left behind as the current state-of-the-art moves endlessly forward. For example, it is now normal to expect high end UNIX systems to include:

- All or a mixture of Symmetric Multi Processing (SMP), Massively Parallel Processing (MPP), and Non-Uniform Memory Access (NUMA) machines
- Level 1, Level 2, and soon Level 3 memory caches

- Multiple terabytes of disk storage

The above was not true just a few years ago. Also, old theories, unsubstantiated rumors, and myths are often perpetuated due to a lack of in-depth understanding of the latest computing advances. To counteract this, everyone in the IT industry needs to keep up to date and refresh their understanding of the latest trends in computer design and technology. This is a key goal of this redbook, and Chapter 2, “Background” on page 5, Chapter 3, “IBM RS/6000 and IBM pSeries architectures” on page 39, and Chapter 4, “IBM RS/6000 and IBM pSeries products” on page 77 cover these aspects in details.

This redbook will naturally have a wide range of readers with a wide range of backgrounds and needs:

- The system designer and architect. If they understand what is available in the latest range of RS/6000 and IBM @server pSeries machines, they can make early choices in the architecture and design to maximize the cost effectiveness of their solution.
- The technical specialist who is sizing a particular configuration needs to understand how to use the industry standard benchmarks and to compromise costs with performance to achieve a balanced system with upgrade potential.
- The system administrator wishing to maximize the performance of the installed machine, spot real or potential performance bottlenecks and reduce them, or perhaps recommend effective upgrades to maintain performance levels.
- The performance tuning expert who understands generic UNIX performance tuning concepts and methods, but needs in-depth RS/6000 and IBM @server pSeries specific information to carry out tuning.
- The application or database specialist who needs to understand the underlying platform to ensure maximum performance of their application and that it is efficiently using the resources available.
- Those new to performance and sizing looking for a grounding in the basics of modern computer design, and who need to apply them immediately to the RS/6000 and IBM @server pSeries range.

Although this redbook contains information for all of the above people, they may wish to consult more in-depth redbooks on individual topics. For example, the sizing specialists and system administrators may need specific knowledge of particular RS/6000 and IBM @server pSeries machines that can be found in the RS/6000 and @server pSeries system handbooks, or

those doing performance tuning might want to consult the redbook *Performance Tools in Focus*, SG24-4989 for more details of the specific tuning tools, their options, and outputs.

When comparing the performance of computer systems it is easy to oversimplify the points of comparison, such as if you compared two fast cars by comparing the official top speeds. The higher the number, the better tuned the car and therefore the faster it will go. However, cars are complex machines, and factors such as brakes, suspension, steering, tires, and regular maintenance and tuning, among many other things, can profoundly affect performance.

When comparing computers it is easy to fall into the same trap - just using one simple measure to determine which is best. For example the MHz rating of the CPUs is a simple measure, but within the RS/6000 there are two models that only differ in the size of the Level 2 memory cache, yet one model is 35 percent faster under a certain workload. Computers are complex machines - just like cars - and it is important to understand that all the components of the computer combine to create good performance. A modern computer needs:

- Fast CPUs
- Fast memory caches
- High band-width memory controllers and system bus
- Fast memory
- Multi-path I/O channels and disk subsystems
- High throughput networks and rapid graphics
- A robust operating system tuned for the hardware

This is available with RS/6000 and IBM @server pSeries and the AIX operating system, but each part of the system has its part to play in the outstanding performance that these machines make available. This redbook explains each component and how it fits in the overall picture.

For the above reasons, this redbook covers the standard industry benchmarks in some detail so that they can be fully understood and used appropriately. The industry benchmarks, performed by the different hardware vendors, can give a guideline on how a certain hardware system performs under a certain workload. However, it is important to understand two aspects:

- Hardware vendors tune their systems to the extreme levels to achieve top performance numbers for a published benchmark.

- Your application will never act exactly the same way as a benchmark application does.

Your best bet is to compare your application workload to the closest benchmark workload as a basis for selecting a hardware system. Chapter 6, “Benchmarks” on page 215 provides an overview on the industry benchmarks that IBM performs on its @server pSeries systems.

In order to size a system, you need to understand the behavior of the application and the workload that it produces. Sizing can only be as good as the information that is available for sizing the system. If there is very little information, then sizing becomes an educated guess as all sorts of assumptions have to be made, introducing inaccuracy. Sometimes application vendors provide sizing recommendations for their applications that can be used to size a suitable hardware system. Otherwise, Chapter 7, “Sizing” on page 255 offers general rules for sizing systems in different environments.

AIX offers a vast amount of performance tools to monitor performance of an IBM RS/6000 or IBM @server pSeries system, including the AIX Performance Toolbox, a graphical performance monitoring tool. Chapter 8, “Performance tools” on page 333 gives a brief overview of the available AIX performance monitoring and tuning tools.

Chapter 2. Background

The objective of this chapter is to review the major theoretical notions that are useful when considering performance or sizing of RS/6000 machines. For the actual architecture and implementation of the RS/6000, refer to Chapter 3, “IBM RS/6000 and IBM pSeries architectures” on page 39 and Chapter 4, “IBM RS/6000 and IBM pSeries products” on page 77.

Understanding hardware architecture or implementation is important for sizing. For example, you need to understand architectures such as PCI and memory bus of RS/6000 models to size or tune your system.

IBM also offers various system architectures such as SMP, MPP, and NUMA today. These are discussed in this chapter as well as software architectures like the AIX Kernel and Monterey/64.

This chapter does not contain a complete description of hardware architectures and UNIX systems. Only performance-related concepts are presented. If you need to know RS/6000 models in detail, the RS/6000 and pSeries Handbooks are helpful. Go to:

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

and search for *Handbook* in order to get a list of current handbooks.

2.1 Performance of processors

The overall performance of a current processor can be calculated like this:

Figure 1. CPU execution time

Execution Time =	**Number of Instructions**	*	**Number of Cycles of Instructions**	*	**Clock Cycle**

The different factors affecting execution time are:

- **Number of instructions**

  The number of elementary operations needed to complete a program in a result of the compilation. This is called the path length.

- **Cycles per instruction**

  This number depends on the complexity of the instructions. The more complicated the instructions are, the higher the number of cycles consumed. But, on the other hand, there are fewer total instructions. This deals with the material discussed in Chapter 2.2.1, “RISC/CISC concepts” on page 6.

- **Clock cycle**

  The smaller the clock cycle, the faster the processor, but the more expensive its production cost.

2.2 Hardware architectures

This section discusses the processor concepts of Complex Instruction-Set Computer (CISC) and Reduced Instruction-Set Computer (RISC), as well as CPU functions that are essential to performance such as pipelining and parallelism. It also examines memory management, including cache and virtual memory concepts.

Further different system architectures, such as the PCI system bus, the different MP concepts, NUMA, and LPAR are discussed.

2.2.1 RISC/CISC concepts

Two different CPU designs have been implemented since the mid-’70s; CISC and RISC.

The first one, complex instruction-set computer (CISC), is the traditional design featuring a large and highly functional instruction set (more than 200 instructions). These instructions need several cycles to complete.

The need for complex instructions existed because, at that time, computers were equipped with small quantities of slow RAM. Complex instructions result in fewer instructions per program, so less memory was needed. But studies showed that only a small percentage of CISC instructions (around 10 percent) were commonly utilized by programs.

Later, as progress in semiconductor technology reduced the difference in speed between memory and processor, and as high-level languages replaced assembly language, the major advantages of CISC decreased.

The reduced instruction-set computer (RISC) concept was first defined by IBM Fellow John Cocke in 1974. It has some basic characteristics:

- A very simple architecture with an optimized set of machine instructions.

  The instruction set consists only of elementary operations (less than 100 instructions) to reduce the complexity of the instruction decoder. Therefore, the CPU can execute with maximum speed and efficiency. The software generates other, more complex operations by combining several simple machine instructions. All these instructions have a fixed length (necessary for superscalar architecture, as seen later in Chapter 2.2.2, “Superscalar architecture: pipeline and parallelism” on page 7).

- A very high instruction execution rate

  The objective of the RISC architecture is to be able to execute an average of one instruction per machine cycle. The execution time can be reduced to less than one instruction per machine cycle using the superscalar architecture, as explained in Chapter 2.2.2, “Superscalar architecture: pipeline and parallelism” on page 7.

- Compiler optimization

  The performance of the RISC architecture heavily depends on the compiler optimization. The compiler has to be able to exploit the hardware architecture by generating instruction sequences that take advantage of the capabilities and performance of the processor.

- Load/store architecture

  Memory access is separated from data manipulations in RISC architectures so that the CPU is not stalled by slow memory access. Data is prefetched into registers, and instructions work within those registers, which are the fastest memory available. Working with registers also allows the compiler to better organize data fetching according to data dependency.

In comparison, CISC tries to reduce the number of instructions for a program, whereas RISC tries to reduce the cycles per instruction.

Nowadays, both of these designs have evolved. RISC architectures, which are commonly utilized in the UNIX world, in particular are benefitting from the superscalar concept.

2.2.2 Superscalar architecture: pipeline and parallelism

A pipeline is a hardware feature, similar to an assembly line, designed to increase instruction throughput through internal parallelism. Different units of the CPU perform, in parallel, the various operations required for fetching, decoding, and executing instructions. Several instructions can be executed in the CPU at the same time. The instructions go along the pipeline stages in

synchronization with the CPU clock. This means that, if everything goes well, each time a new instruction enters the pipeline, an older one is exiting. This results in one instruction per pipeline and per cycle. Thus, although the time it takes to complete each instruction is not directly affected, pipelining increases the overall rate at which instructions complete.

When pipelining works as intended, performance is optimized. However, there are some potential problems; branch instructions and data conflicts. A pipeline normally holds a number of instructions in different stages of execution. Consider the case where one of these is a conditional branch, dependent on the condition code to be produced by a not-yet-executed instruction coming through the pipeline. Should it later turn out that the branch is to be taken, the system has to discard all the instructions prefetched after the branch and continue from the branch target address instead. A “bubble” in the pipeline will develop, leading to wasted CPU cycles.

A true data dependency arises when an instruction entering the pipeline needs the result still to be produced by an instruction further ahead in the pipeline. This case cannot be resolved by register renaming, the technique employed to avoid data conflicts. The instruction simply has to wait on the newer one to produce the result.

While true data conflicts are uncommon, branches are frequently encountered. In fact, branch instructions constitute about 20 percent of the instructions in most computer architectures. Branch target prediction as used in the RS/6000 alleviates the problem to a certain degree. The basic problem that remains is that very complex software, like kernel code and database systems, suffers a slowdown of CPU speed in the pipeline because of the high percentage of conditional branch instructions that are typical for these environments. Simpler applications are less affected by this problem.

Next, came the idea of making several pipelines in order to implement further parallelism, which is called superscalar architecture. The instructions had to be distributed between the different pipelines and no more sequential treatment was possible. That is why compilers are so important in the RISC superscalar architecture; complexity no longer lies in the instruction itself but in the compiler. But the advantage of a compiler is its ability to be optimized continuously, quickly, and much more easily than hardware code. Superscalar implies several independent execution units, like branch units, fixed-point units, or floating-point units.

Superscalar allows more than one instruction to complete in a clock cycle. The objective is to achieve the highest number of instructions per cycle.

While the superscalar architecture aims at issuing more than one instruction per cycle, this goal is achieved only when the proper mix of instructions and data is sent through the pipeline. Some benchmarks will perform at several instructions per cycle, but the throughput might go down to less than one in other applications. This has nothing to do with instruction length because the processor can handle the large percentage of floating-point instructions typical of technical and scientific applications. Actually, this instruction mix promotes parallelism because load and store operations and loop counting are handled by the fixed-point unit. The challenge for superscalar, highly pipelined RISC architectures lies in complex commercial applications that use the fixed-point and branch units only. These applications tend to have very short sequential execution paths and poor locality (as discussed in Section 2.2.3.2, “Locality concept” on page 13).

Simplified scheme of superscalar CPU architecture

Figure 2 on page 10 shows a model of a three-pipelined architecture, the independent processor units being the branch, the fixed-point, and the floating-point processor units.

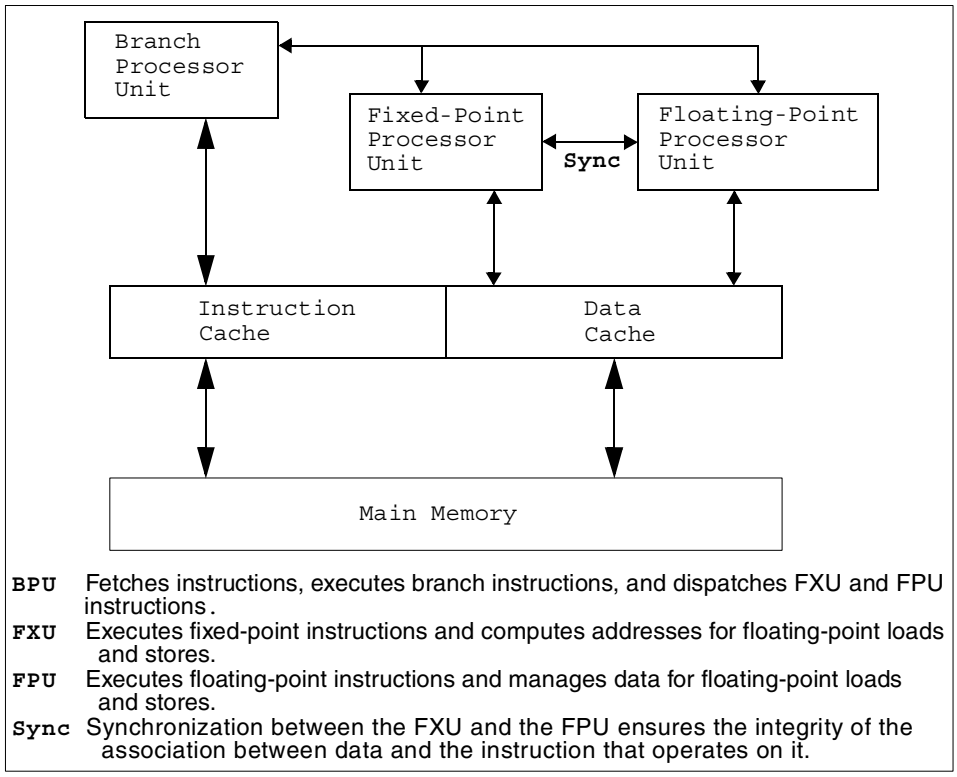


Figure 2. Pipelined Architecture

2.2.2.1 CPU performance enhancements

IBM researchers have announced breakthrough results in developing a new family of experimental high-speed computer circuits that run at test speeds up to five times faster than today's top chips.

The new circuits employ an innovative design - called "Interlocked Pipelined CMOS (IPCMOS)" - to reach speeds of 3.3 - 4.5 billion cycles per second (3.3 - 4.5 GHz) using new copper based transistors while dramatically reducing power consumption. IBM researchers estimate that chips made with IPCMOS circuits would require only half the power used by a standard high-performance chip.

2.2.2.2 Speeding up the clock

The key to the IPCMOS design is a distributed *clock* function. In computer chips, the clock paces the speed of the circuits. Standard designs use a centralized clock to synchronize the operations of an entire chip, ensuring that all operations run at the same interval, or cycle. The clock waits for all the

operations on a chip to finish before starting the next cycle, so the speed of the entire chip is limited to the pace of the slowest operation. To increase the speed, the IBM researchers decentralized the clock, using locally generated clocks to run smaller sections of circuits. This locally generated clock has two significant advantages:

- Speed

  Faster sections of circuits are free to run at higher cycles without needing to wait for slower operations to catch up.

- Power

  The distributed IPCMOS clocks send signals locally only when an operation is being performed, significantly reducing power requirements where as centralized clocks send a signal to the entire chip. The synchronizing function can use as much as 2/3 of the total power consumed.

2.2.2.3 Reference

Additional information can be located on the web at the following URL:

- http://www.research.ibm.com/news

2.2.3 Memory management

Efficient memory management can increase system performance. There are several layers and concepts involved.

2.2.3.1 Memory hierarchy

Memory hierarchy is often referred to as having four levels spreading from disk to CPU, but in advanced microprocessor architectures, this scale can be extended to many more levels, including cache levels L2 and L3, which are most common, or a multiple level hierarchy in real memory itself (local memory and remote memory).

Figure 3 on page 12 shows a typical memory hierarchy. At the apex of the pyramid, memory is expensive but the access times are fast, whereas at the base of the pyramid memory is considerably less expensive but the access times are much slower.

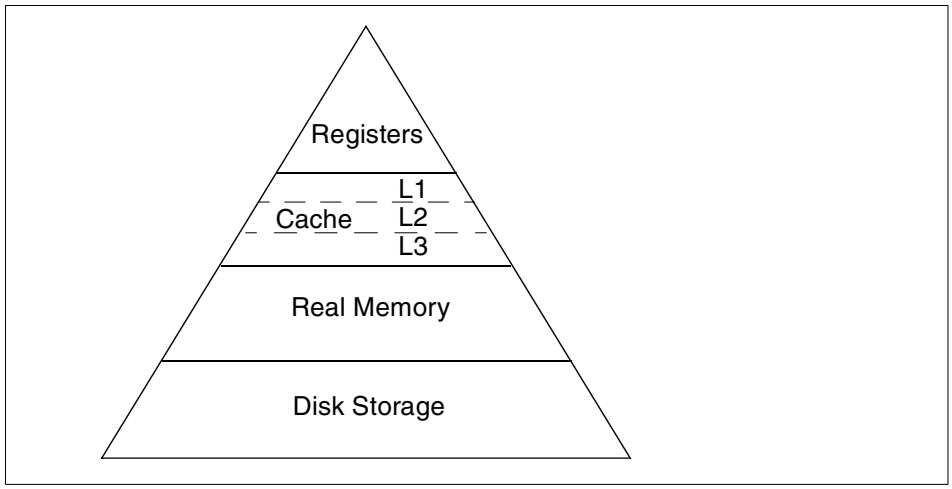


Figure 3. Memory Hierarchy

Each level in the pyramid is scarcer and more expensive than the one below.

- **Registers**

  Registers are storage cells within the specialized units inside the CPU pipelines. This is the fastest memory available, but there are only a few registers. Access is immediate.

- **Cache**

  Cache is a high-speed memory containing only a subset of main memory. This element is of great importance regarding performance considerations. Indeed, if the CPU accesses the cache instead of main memory for the most-frequently utilized instructions and data, it will gain many clock cycles.

  There are usually three different types of cache; levels 1, 2, and 3.

  On-chip caches (usually L1, sometimes also L2) are located next to the pipelines, and are the smallest.

  Generally, there are one or two cache levels that are off-chip.

  L3 cache storage capacity is bigger than that of L2, but its access time is slower. It can be a superset of the L2 cache. When L3 is implemented, L1 cache generally is put on the chip for performance reasons.

- **Real memory**

  If the data is not in the cache, the data is fetched from main memory.

- **Disk**

  If the data is not in main memory, a page fault takes place and the data is retrieved from hard disk. This is by far the slowest way to get data.

2.2.3.2 Locality concept

One of the basic principles defining how hardware and software interact is the concept of locality. Hardware expects that programs will exhibit patterns of address reference that are local both in time and space. To put it another way, it is assumed that programs access instructions and data according to the following models:

- **Locality in time.** This means that, if an address is referenced, it is likely that it will be referenced again soon.
- **Locality in space**. This implies that, if an address is referenced, it is likely that nearby addresses will also be accessed in the near future.

The principle of locality has given rise to the concept of working sets. The working set of a process is the collection of memory addresses that the process is currently using. This means addresses that the process has recently referenced or is likely to use in the near future. The working set thus comprises those memory ranges that the process needs to have access to without any significant delay in order to achieve maximum performance.

Inherent in the concept of working set is the observation that active address ranges normally do not shift gradually but rather tend to be replaced entirely in phase transitions. Most programs behave so that they remain in one area of memory for some time, then suddenly move to another area, remain there for some time, and so on.

Although locality and working sets are rather vague concepts based on empirical observations rather than strict laws, they are the rationale behind two very powerful architectural features of today's computers; caches and virtual memory.

2.2.3.3 Cache

As explained before, cache memory sits between the CPU and main memory.

The L1 cache memory is nowadays typically divided into two sections, one for data (D-cache) and one for instructions (I-cache). In this way, for example, while the arithmetic units work on numeric data in the data cache, the branch processor can simultaneously load new instructions from the instruction cache, which increases parallelism. Lower level caches are normally common caches.

Caches exploit locality on a smaller scale and offer much faster access times than main memory or disk.

Cache can be either integrated within the memory management unit (MMU) or located outside the processor (external cache). Most modern RISC architectures now implement both internal and external caches in order to reduce access to main memory by having a bigger global cache size. In terms of performance, the nearer to the pipelines the cache gets, the smaller the access time is.

Data organization
As the cache only contains a subset of main memory data, its data needs to be referenced for the CPU to find it.

Data is organized in lines because too much space would be used to reference each byte otherwise. So each line begins with a tag containing the main memory address of the first byte and some control information like the valid bit. Then comes the real data, made of contiguous words.

When a cache miss happens, the whole line must be fetched from memory because there is only one tag to reference the line.

The line size has some consequences on performance. Indeed, if you choose a small line size like 32 bytes, then a higher percentage of cache space is occupied by tags. This results in a smaller amount of cache, but data transfers between cache and main memory are almost immediate. On the other hand, if you choose a long line size like 128 or 256 bytes, it results in a larger amount of cache available for data, but transfers between main memory and cache are slower because you need to fetch the whole line from main memory. For this kind of implementation, dividing a line into several sublines, each independent and with its own valid bit, may improve transfer time.

Hit ratio
Among the various factors influencing performance, one of the most important in determining processor throughput is the cache hit-to-miss ratio. To achieve optimal performance, the CPU needs to achieve a high percentage of cache hits, meaning that the instructions or data required are present in cache memory. If not, the processor will have to wait for the information to be loaded from main memory, which implies a performance degradation of as much as 50 percent. Effectively, while page faults cause either context switches or I/O waits, cache misses actually force the CPU into a wait state, and this forces idling while the requested data or instructions are fetched from memory or, in the worst case, disk.

The CPU wait state is forced because access to real memory is slower than access to the caches by more than an order of magnitude. Furthermore, the RISC architecture and the highly sophisticated pipelines found in the RS/6000 design work at top efficiency only when they can access code and data at a rate of two to four words per CPU cycle.

In addition to the cache hit-to-miss ratio, there is another important factor to be considered; the miss penalty, defined as the number of cycles the CPU must wait while the cache miss is being resolved by the memory subsystem. The cost of a cache miss, in terms of performance, is the product of the cache miss ratio and the miss penalty.

In general, the instruction cache is smaller than the data cache because programs are typically executed in chunks of four to five sequential instructions before the next branch instruction is encountered. Also, the hit rate is usually higher and the average access is faster than for the data cache because the instruction cache is never written to, and the consequences of a cache miss are more severe because the CPU pipelines are immediately stalled.

Cache access

The first goal of a cache is to access data faster than memory. Therefore, cache searching must be very quick and efficient.

Generally, it utilizes a hashing algorithm to index the CPU addresses to locations in the cache (except for fully associative caches). Hashing implies that different CPU addresses can have the same index. The cache line tags with this index will then have to be compared to the CPU address to find out if it's a hit or a miss. The hashing algorithm has been chosen because it is an efficient way of limiting the search to only a few lines (the ones that refer to the same index).

Several cache organizations follow:

- **Direct mapped cache**

  The index refers to only one line of the cache where data may be stored. This is the simplest organization. However, as hashing will produce the same index for many different addresses, it can end up in cache thrashing. This happens when the same lines are continuously replaced by new ones before being reused.

- **n-way set associative cache**

  This organization is aimed at reducing the probability of cache thrashing. The idea is to group several lines (n) and to refer to them with one index.

Each line is independent of the others in its set and has its own tag. Thus, when the CPU looks for an index, it has just n tags to compare to its own address. These comparisons are made in parallel to avoid reducing performance. Cache thrashing is less likely, as several lines are provided for each index.

- **Fully associative cache**

  This is a particular case of the preceding organization, when n equals the total number of lines in the cache. It means that there is only one set of lines. So no hashing is implemented. All the lines are looked through in parallel for each search. This is the most expensive cache organization. That explains why it is used only for small caches such as translation lookaside buffers (TLB).

  When new data has to come into the cache, some existing line or subline must be put aside. This replacement policy, by which data is selected for removal, is usually done according to the least recently used (LRU) algorithm, which is easier to implement than techniques used for main memory such as page aging.

  Another extremely important policy is the update policy. The CPU has to store data. It can do this either to main memory or to cache. If the latter option is chosen, it increases the cache hit ratio because of locality, and the store time is decreased. That is why, in most cases, CPUs store data to cache.

But to ensure data integrity, cache needs to be consistent with main memory. Two options exist. First, write the data both to cache and memory. This is called the write-through policy. The advantage is complete coherency with memory. But it ignores the locality concept and always wastes a memory cycle. The other policy, called the write-back policy, asks the CPU to write only to cache. Data will be written to main memory just before it would be discarded (due to the replacement policy) or if the operating system requests it. Performance enhancement is quite clear, as fewer writes to main memory will occur, but this is done at the expense of main memory consistency. This policy is widely used throughout the different implementations.

Performance considerations

- The bigger the cache is, the less main memory will be accessed.
- The write-back policy yields better performance than the write-through policy.
- For small caches, it is generally better to have large sets of lines so that the replacement policy will not induce too much cache thrashing.

- Due to spatial locality, the line size should be as large as possible. But very large line sizes will add some overhead when loading lines from memory.

2.2.3.4 Virtual memory concepts

Virtual memory has two technical meanings:

- The system can behave as though it has access to more physical memory than actually exists on the system. For example, a 32-bit system is limited to 4 GB of real memory. However, AIX uses a virtual memory manager model that can support as much as 4 PB (4 Petabytes = 4,000 Terabytes) of virtual memory. This is accomplished by implementing a 52-bit virtual address.
- Process text and images are given effective addresses by the compiler, as opposed to real addresses. Because they have effective addresses, they can be loaded at any real memory location. Virtual memory allows many programs to occupy memory at the same time.

Swapping

Originally, UNIX systems used a technique called swapping to provide virtual memory. In a swapping environment, entire process images are loaded into real memory. Therefore, when a process is not needed in real memory (such as when it is sleeping), its image is transferred out to a secondary storage device. This secondary storage device is usually a disk partition known as the swap space. This swap space provides a backing store that allows the system to appear to have more physical memory than it actually has. The drawback to swapping is its slow mechanism, as the entire image of the process must be moved from real memory to swap space and back.

Paging

A newer virtual memory management technique is paging. In a paging environment, only the most popular pages of a process occupy memory at any given time. A page is a small chunk of code or data that has a fixed size throughout the system. For example, AIX Version 4 uses a 4 KB page size.

Like swapping, paging utilizes a secondary storage device, called the paging space, for backing store. When available real memory space for pages becomes scarce, the system moves the least popular (usually least-recently accessed) pages out of memory to the paging space, making paging completely independent of any process.

There is also a hybrid approach to managing virtual memory. Paging is the standard method, but when real memory becomes overcommitted, the system begins to swap processes. Usually, only sleeping processes will be

swapped out. The swapped out processes must then be put back into real memory before they can be made ready to run. This approach is utilized by AIX Version 4.

Performance considerations
When dealing with memory, a couple of issues come up:

- **Thrashing**: The system spends more time handling page ins and page outs than performing computational tasks. Thrashing occurs when there is so much demand on the real memory that it becomes over-committed. It is a direct result of not having enough real memory to handle the workload. Thrashing is often characterized by a sudden slowdown of system response time and a large amount of disk activity.
- **Running out of paging space**: If not enough paging space is defined, it causes the kernel to prevent new processes from starting. The *SIGDANGER* signal is sent to most processes in alert. If the condition persists, the kernel may be forced to terminate processes.

I/O buses
So far, we have reviewed notions dealing with the internals of a processor, cache, and memory management. Another performance-related factor that we need to consider is the I/O bus.

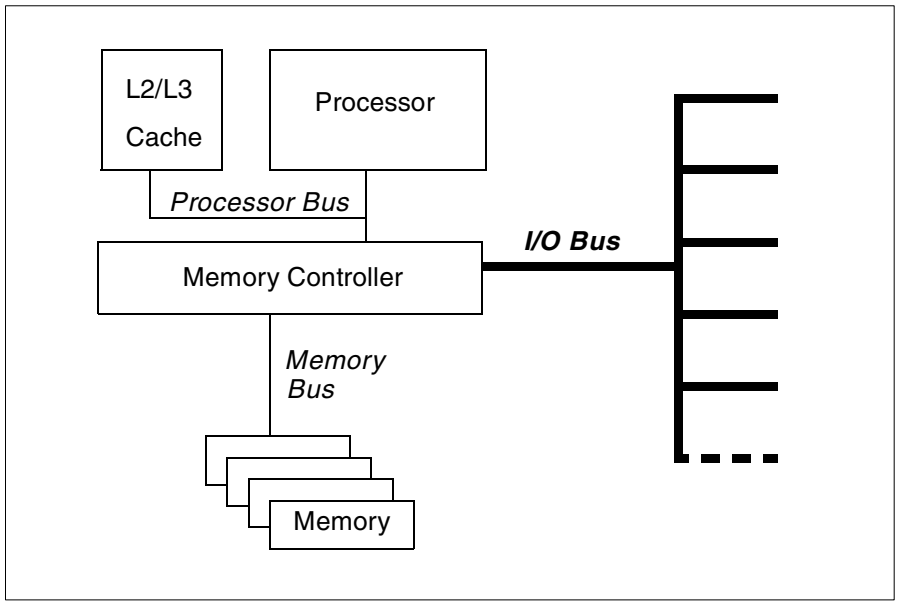


Figure 4. Simplified System Architecture: Focus on Buses

2.2.4 PCI

The Peripheral Component Interconnect (PCI) local bus specification was developed by the PCI Special Interest Group (PCI-SIG), led by a group of companies including Compaq, IBM, Intel, Digital, and NCR. Introduced in 1992, the PCI bus architecture has quickly gained widespread industry acceptance.

The goal was to provide a common system-board bus that could be used in personal computers, from laptops to servers. It was envisioned as a local system board bus that would serve as a common design point, supporting different system processors as the various processors evolved over time. This is much like operating systems that have defined Application Binary Interfaces (ABIs) so that applications need not change with each generation of the operating system. The PCI Local Bus would serve as a common hardware interface that would not change with different versions of microprocessors.

The group defined PCI to support the high-performance basic system I/O devices, such as the graphics adapter, hardfile controller, and/or LAN adapter. In the original definition, these would be mounted on the planar and would communicate through the PCI bus. Current I/O buses (ISA, EISA, and Micro Channel) would be used to attach pluggable features to configure the system for the desired use. The first release of PCI Specification was made available in June of 1992.

The PCI Special Interest Group (SIG) soon realized that the PCI bus needed the capability to support connectors. For example, display controller evolution doesn't necessarily match planar development, so providing for an upgrade of the display controller became a requirement. The next release of the PCI Specification (Version 2.0 in April of 1993) included upgrade capability through expansion connectors.

The original design for the PCI bus was to move high bandwidth peripherals closer to the CPU for performance gains. This need for more bandwidth has compelled system vendors to find ways of increasing the throughput of the PCI bus, and the system.

The PCI bus is a clock-synchronous bus that runs at up to 33 MHz for standard operations. It can transfer either 32-bit or 64-bit data. This yields a peak local bus performance of 132 MB/s for 32-bit transfer and 264 MB/s for 64-bit transfer at a clock speed of 33 MHz. PCI allows low-latency random access such that at 33 MHz; as little as 60 nanoseconds are required for a master on the bus to access a slave register.

2.2.4.1 PCI features and benefits

The PCI bus architecture has many advantages involving the following:

- High data transfer speed
- Processor independence
- Cross-platform compatibility
- Plug and Play
- Investment protection

High data transfer speed

The high-speed data transfer is implemented by the following functions:

- Buffering and asynchronous data transfer

  The PCI chip can support the processing and buffering of data and commands sent from the processor or from the peripherals in case the peripheral or the processor is not yet ready to receive the information.

- Burst mode transfer

  Variable length linear or toggle mode bursting for both reads and writes improves write-dependant graphics performance.

- Caching

  To reduce the access time, the PCI bus architecture supports caching of frequently used data.

- DMA

  The Direct Memory Access (DMA) function is used to enable peripheral units to read from and write to memory without sending a memory request to the processor. This function is very useful for peripherals that need to receive large amounts of data, such as video adapters, hard disks, and network adapters.

Processor independence

Processor independence allows manufacturers to implement PCI buses on any computer. Any PCI-compliant peripheral will work on any PCI-compliant bus implementation.

Cross-Platform compatibility

The key to cross-platform compatibility is processor independence. Until PCI, different systems used different buses, such as ISA, EISA, NuBus, and so forth. Now, different systems can use one bus.

Multi-bus support
An important aspect to PCI-based system architecture is support for multiple PCI buses, operating transparently to existing software.

Plug and play
PCI peripherals, following the PCI standard, load the appropriate set of installation, configuration, and booting information to the host CPU without user intervention. This provides a greater ease of use for the system integrator or end-user.

Investment protection
The PCI bus is designed for 64-bit addressing support.

Summary of I/O Bus Capabilities
Table 1 summarizes the capabilities of the I/O Bus.

Table 1. I/O Bus Capabilities

Feature	Capability
Data Path Width (bit)	32 or 64
Data Bus Speed (MHz)	33
Data Transfer Rate (MB/s)	132 or 264
Data Rate Implemented	132

2.2.4.2 References
Additional information can be located in the following Redbooks:

- *Technical Introduction to PCI-Based RS6000 Servers*, SG24-4690
- *Understanding IBM RS/6000 Performance and Sizing*, SG24-4810

2.2.5 MP implementation specifics
Different types of Multiprocessor (MP) technologies coexist. The three major ones are:

- **Shared Memory MP**

  A symmetric multiprocessor, also known as a shared memory or tightly coupled MP, has multiple processors that have their own cache and can each address the shared memory and all devices. User processes on any processor see the full machine. If two or more processors access the same word in memory, hardware keeps the caches consistent, invisible to application processes. Compared to other multiprocessor types, the

advantage of SMPs is their use of the same programming model as uniprocessors.

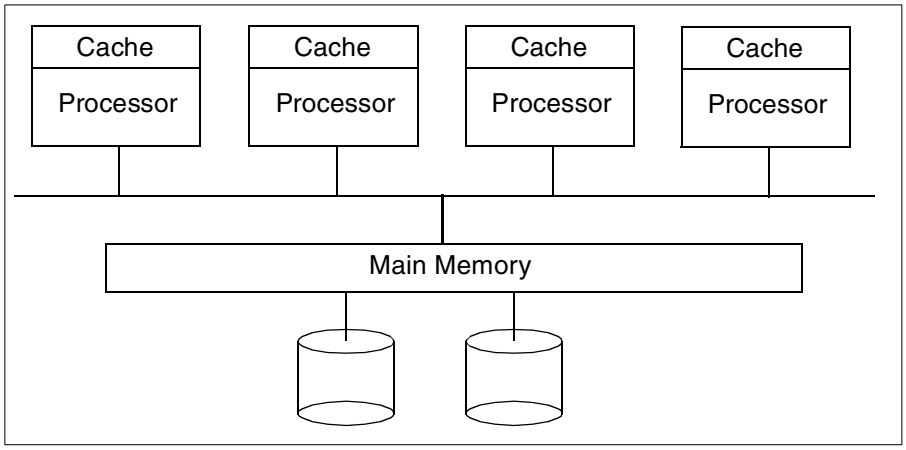


Figure 5. Shared Memory MP

- **Shared Nothing MP**

  All processors have their own memory and disks. Uniprocessor programs must be changed to use the parallelism of this configuration because they must pass messages across an interconnect in order to use the multiple processors.The IBM RS/6000 SP is an example of this kind of architecture.

  Shared nothing MPs generally scale better than SMPs because they have no memory bus contention and no cache coherency problems among the processors.

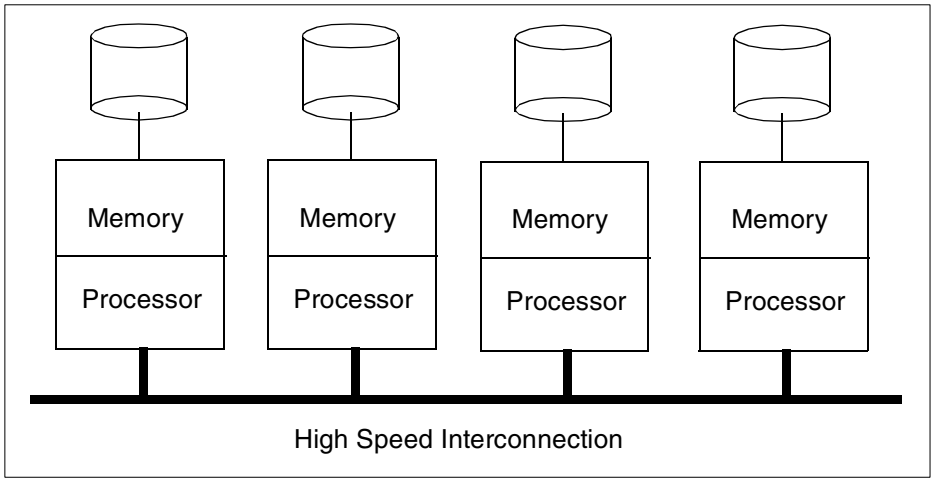


Figure 6. Shared Nothing MP

- **Shared Disk MP**

  Unlike the SMP, each processor on a shared disk multiprocessor has its own memory. That is why the shared disk multiprocessors, like the shared nothing multiprocessors, have no memory bus contention or cache coherency problems among the processors. However, a centralized locking scheme is used to control access to the disks. This locking scheme requires changes to some applications (such as databases), and generally offsets the performance advantages of no memory bus contention or the cache coherency problem.

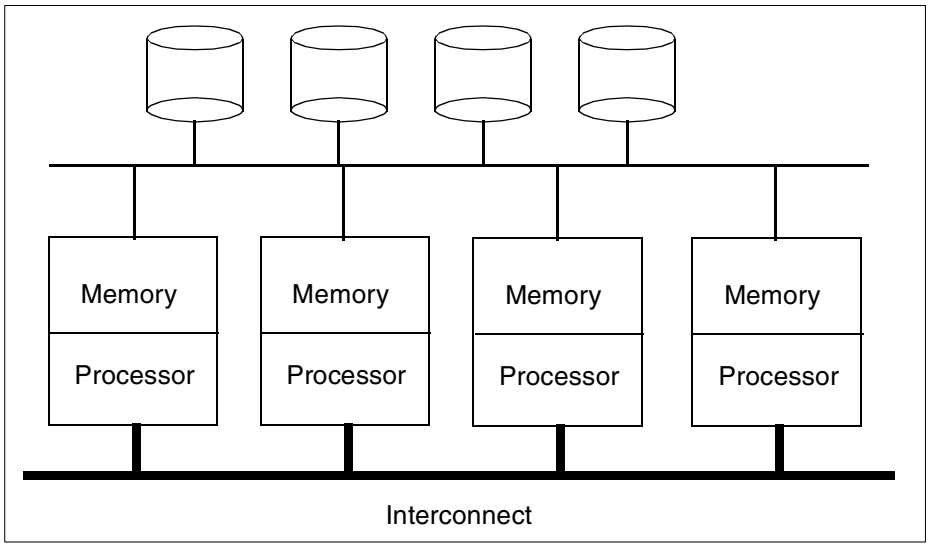


Figure 7. Shared Disk MP

2.2.6 NUMA

Non-Uniform Memory Access (NUMA) was developed to offer better scalability for large servers. The demand for scalability has increased due to the requirements of large databases and decision support systems such as e-business applications where server load is a key issue.

IBM has done significant work and research on NUMA technology for many years, and when Sequent joined IBM, they brought with them a lot of experience in NUMA performance and tuning.

There has been a steady increase in demand for systems that offer higher CPU power. If the system has performance problems, the logical solution would be to add additional CPU power to the system, but this solution does not address the issues of memory accessing that can quickly erode any performance increases of additional CPU power.

Figure 8 on page 26 shows some of the system components that limit scalability of non-NUMA architectures.

Architecture plays an important role in how a system performs. Architecture must take advantage of the CPU and marketplace technologies and offer scalability. To enhance CPU throughput, the following technologies were developed:

- Symmetrical Multi-Processor (SMP) - share everything
- Massively Parallel Processors (MPP) - share nothing resource.

There are advantages and disadvantages to both philosophies, and each is suited to different environments. With SMP, the programming model is easier.

MPP provides very high performance for compute-intensive workloads, but requires data partitioning and is therefore not a good choice for running some commercial applications.

With SMP's easy programming model, SMP has been very popular and over time the need for performance of SMP has increased, but with SMP's architectural limitations, you cannot just add CPUs and expect the equivalent gain in performance.

To take advantage of faster CPUs, physically shorter busses and busses with fewer central interconnects are needed to be able to reap the benefits of memory with ever decreasing latency times.

With NUMA, the concept is to combine these areas to offer program simplicity and the flexibility of SMP while providing low-latency, high- multiprocessing for commercial applications.

Software that runs on an SMP system will run on NUMA systems. Some by their nature will even run efficiently, but others will need to understand the NUMA characteristics in order to perform well on a NUMA system. So even if certain software runs well in a large SMP, that is no guarantee it will run well in a NUMA environment.

NUMA combines the resources of a group of systems and allows sharing of data between them. For example, the memory on multiple servers appears as one.

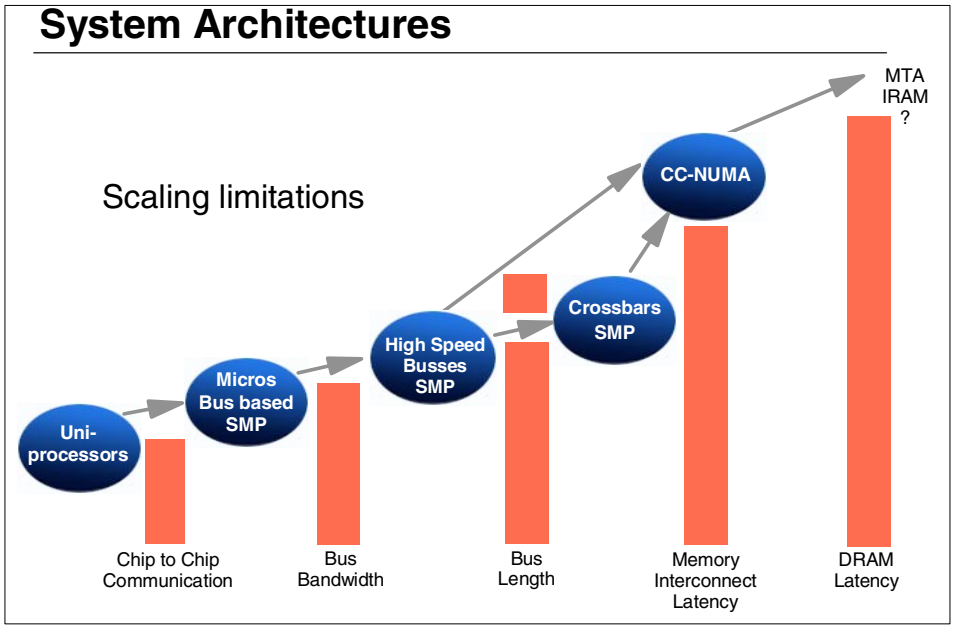


Figure 8. System Architectures

2.2.7 Logical partitioning (LPAR)

Logical Partitioning (LPAR) allows the individual allocation of resources (processors, memory, I/O adapters) in any combination and runs a separate operating system on each allocated partition. Each partition can run a different level of operating system. This can be done on SMP systems or NUMA nodes.

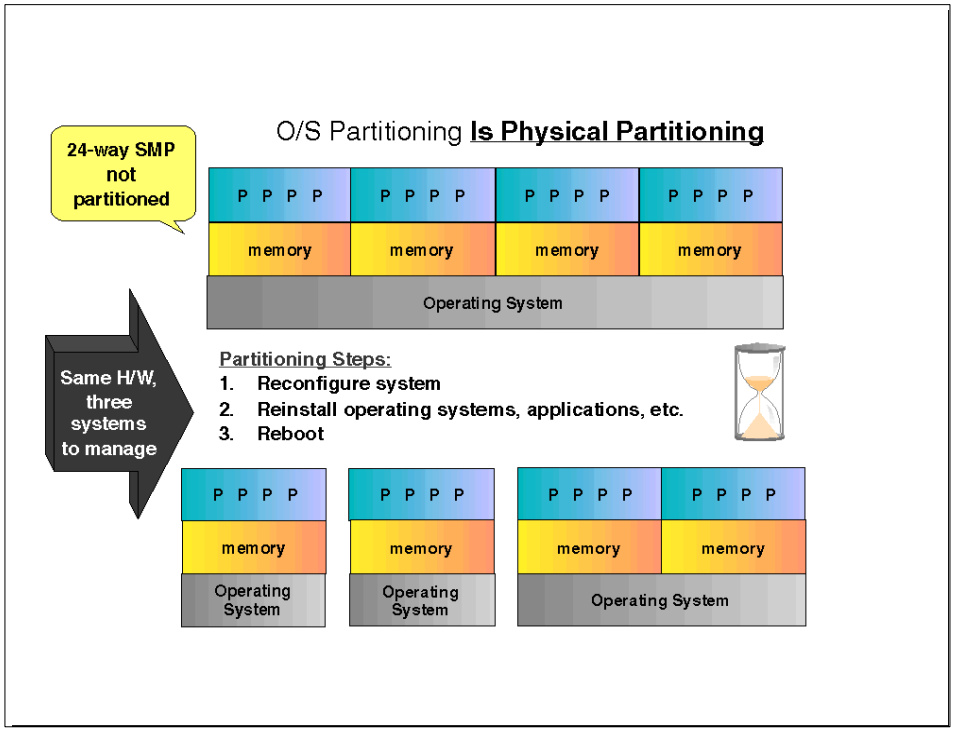


Figure 9. LPAR

Each partition has its own memory, processors, and I/O.

Partitioning can be used to solve several problems, like running production and test versions of an application or operating system on different partitions for verification or certification purposes. It can also be used for operating system fault isolation. Also, application failures in one partition do not affect other partitions. It does not protect from global hardware faults, however.

Compared to an unpartitioned system, extra resources are needed due to the fact that each partition requires it's own operating system that has to be managed as an individual system.

Resources may be wasted because the granularity of control is done on hardware boundaries, such as individual processors. Because resources cannot easily be switched from one partition to another, free resources on one partition will be wasted.

A more flexible solution to this problem is provided by various workload management products, such as the AIX Workload Manager (WLM). Detailed information on WLM is available from the redbook *AIX 5L Workload Manager (WLM)*, SG24-5977.

2.3 AIX kernel

UNIX is a multiuser and multitasking operating system. The kernel is the core of the operating system.

2.3.1 Description

The kernel is a linked object file similar to a user application, and provides the following functions:

- Mechanism for creation and deletion of processes
- CPU scheduling. Because UNIX is a time-sharing operating system, the kernel implements scheduling routines to fairly allocate processor time slices to processes.
- Memory management. The kernel allocates and de-allocates virtual memory for all active processes.
- I/O Handling. The kernel provides the I/O path between applications and the system hardware. I/O support includes device and file I/O. The kernel provides the mechanisms for the creation and management of files via file systems. The UNIX system establishes file names to represent logical and physical devices. These file system abstractions are found, by convention, in the /dev directory. This concept allows applications to access devices for I/O as if they were ordinary files. Thus, the file system provides the application interface for device I/O.
- Handling the loading and execution of programs.
- Synchronization tools.
- Communications tools for interprocess communication.

The AIX Version 4 kernel is preemptable. A preemptive kernel means that a thread that is running in kernel mode can be interrupted and, upon return from the interrupt, the preempted process can retain control of the CPU.

Other higher priority process can get control of the CPU. AIX Version 4 permits processes to be preempted even if they are in the midst of a system call. Kernel locks are provided to safeguard kernel data integrity.

2.3.1.1 The AIX Version 4 kernel is pageable

A pageable kernel means that portions of the kernel can be paged out to paging space. This allows more real memory for applications. Of course, some critical portions of the AIX kernel, such as interrupt handler code and data, are pinned in memory. A pageable kernel means that the system time needed for a system call can vary depending on whether the pages called are in memory or in the paging space.

Most other kernels do not allow the kernel to be paged out, so that the entire kernel must be loaded and pinned into memory, limiting memory availability for other processes. AIX defines kernel extensions as entities added to the base kernel. These extensions can be added to the kernel dynamically, without the need to reboot the system.

Kernel extensions are:

- Device drivers
- System calls
- Virtual file systems (journaled file system, network file system, CD-ROM file system)
- Streams modules (an AT&T creation that facilitates the creation and implementation of character I/O mechanisms)

2.3.2 Executable file formats

AIX defines the format of a compiled, executable file as Extended Common Object File Format (XCOFF). XCOFF is based on the AT&T definition of COFF. Programs compiled by the AIX C, C++, Pascal, or Fortran compilers generate XCOFF files. One of the major interests in XCOFF is its ability to dynamically resolve references to shared libraries and other external objects. COFF, on the other hand, can only resolve references statically.

In AIX, the kernel allows the process to share resources simultaneously among many processes and users. Many other, non-UNIX operating systems allow only single access to the machine resources.

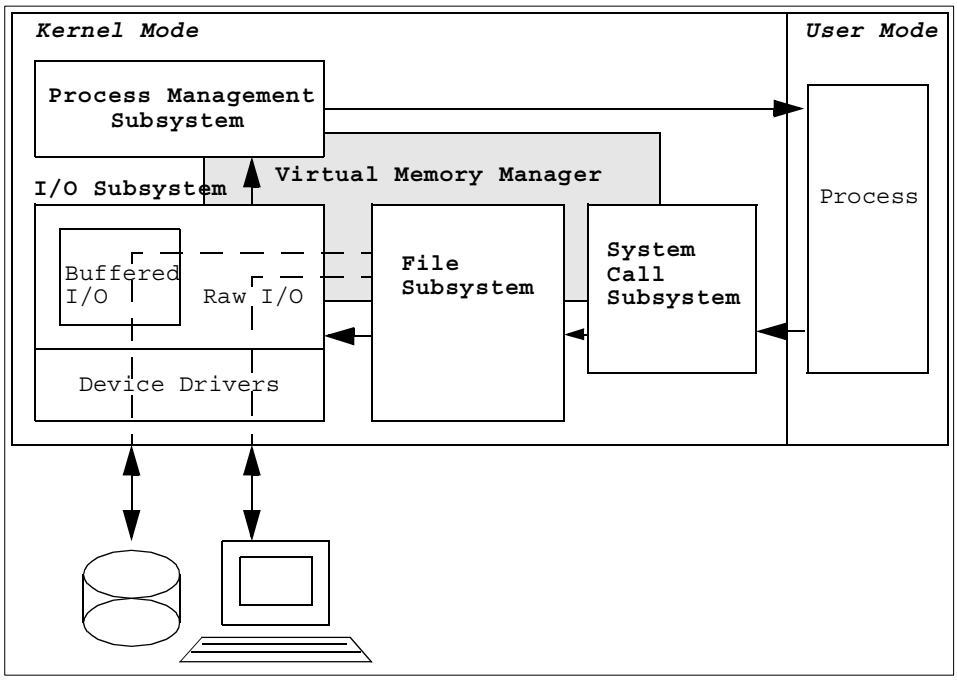


Figure 10. AIX Version 4 Kernel Subsystems

2.3.3 Kernel and user mode

When the system is executing user code, it is said to be running in user mode. When the system is executing system calls or other kernel code, such as interrupt handlers in the device drivers, it is said to be running in kernel mode. User applications run in user mode, but if they require the use of a system call, (for example to read or write from disk) this is done in kernel mode. A process in kernel mode cannot be interrupted by the user with a signal to the running process, but it can be preempted.

2.3.4 I/O

Processes access the I/O and process management subsystems via the system call subsystem. The process management subsystem is responsible for scheduling and dispatching processes. The kernel uses two types of interfaces to devices; buffered I/O and raw I/O. Buffered I/O (like hard disk drives or floppy diskette drives) is performed in blocks of data. The blocking factors and schemes used by block devices are controlled by device drivers. Raw devices (such as printers or terminals) perform I/O one character at a

time. The virtual memory manager supports both process management and device and file I/O.

2.3.5 Context/Thread switches

In AIX Version 4, the scheduled entity is the thread (as opposed to process in Version 3). There are 128 levels of priority (0-127). The lower the number, the higher the thread priority. The scheduler recalculates the priorities each second. Each clock tick (time slice), the priority for the currently running thread is recalculated and the dispatcher chooses the thread to run.

A context switch occurs when the execution of the current process is stopped and replaced by the next one (determined by the scheduling policy). The system must store the state and data of the current process and then load those of the next process to be executed. This will slow system performance.

A thread switch is the corresponding action when dealing with threads inside the same process. Because the same process is still executing, less information needs to be safely stored before switching to the next thread. Therefore, this mechanism is much faster than the context switch.

2.3.6 Virtual address space

Not to be confused with physical memory (the amount of memory contained in the physical memory chips in the system), the virtual address space is the range of addresses that a process (or the kernel) is allowed to reference.

A 32-bit effective address on AIX is broken into 16 segments, each 256MB in size. The total 32-bit effective address space is therefore 4 gigabytes.

A 64-bit effective address on AIX is broken into 16^{23} segments, each 256MB in size. The total 64-bit effective address space is therefore 16 exabytes.

Processes have access to a limited range of virtual addresses given to them by the kernel.

The AIX Virtual Memory Manager (VMM) tries to keep the physical memory as full as possible. By doing so, when a page frame is required there is more chance that it will be in memory. The virtual memory manager keeps track of which pages are clean and which are dirty so the page stealers can obtain memory without paging.

The virtual memory manager can be tuned with AIX's `vmtune` command. Please refer to the *AIX Performance Management Guide* for further information.

2.3.7 Demand paging

Demand paging occurs when a page is retrieved from either a disk or a page space and is transparent to the user. Demand paging saves much of the overhead of creating new processes because the pages for execution do not have to be loaded unless they are needed. If a process never uses part of its virtual space, valuable physical memory will never be used.

The virtual memory manager can be tuned with AIX's `vmtune` command. Please refer to the *AIX Performance Management Guide* for further information.

2.3.8 Kernel scalability enhancements

Kernel scalability enhancements have greatly increased OLTP throughput. Because the bigger machines are capable of handling large memory configurations, AIX 4.3.3 supports multiple lists of free memory frames. A frame is a 4 KB unit of real memory. It maps 1:1 to a 4 KB page of virtual memory. The latest kernel also supports multiple page replacement daemons. These constantly-running threads manage real memory by deciding whether the contents of a location in memory should remain where they are or be moved to disk where it will take more time to retrieve them in the future. Allowing multiple lists and having multiple daemons reduces memory contention and latency (the time needed to retrieve data or instructions needed by a processor).

In another kernel enhancement, runnable threads are assigned to local run queues on a per-processor basis. This simplifies the dispatcher's decision about which thread to run next by reducing lock contention and eliminating time-consuming calculations needed to maintain affinity between a processor and its cached data. The algorithms used by the dispatcher have been tuned to provide better transaction throughput on busy SMP systems. At AIX 4.3.3, user threads generate less cache interference and maintain a greater affinity to a single processor.

2.3.9 References

Additional information can be located in the following publications:

- AIX 4.3 Kernel Internals Workshop (Course Code Q998C).
- RS/6000 S-Series Enterprise Servers Handbook, SG24-5113

2.4 64-bit architecture

Rapid advances in chip technology have enabled very large and complex chips to be manufactured. This has made possible the move from 32-bit to 64-bit chips. 64-bit chips will typically be up to twice the area of 32-bit chips. This area increase is due to data buses that are twice as wide as before, to the many registers that are twice as wide as before, and to the arithmetic and logical units that now need to be able to process data of twice the previous width. Thus, a 64-bit architecture enables the more efficient handling of 64-bit data types and the utilization of more physical memory.

2.4.1 Concepts

From an operational point of view, an architecture is said to be 64-bit when:

- It can handle data 64-bits in length; in other words, a contiguous block of 64-bits (8 bytes) in memory is defined as one of the elementary units that the CPU can handle.This means that the instruction set includes instructions for moving a 64-bit long data, and arithmetic instructions for performing arithmetic operations on 64-bit integers.
- It generates 64-bit addresses, both as effective addresses (the addresses generated and utilized by machine instructions) and as physical addresses (those that address the memory cards plugged into the machine memory slots). Individual processor implementations may generate shorter physical addresses, but the architecture must support up to 64-bit addresses.

RS/6000 64-bit architecture, however, has some important characteristics that make it different from other processor architectures. As opposed to some of its competitors, it was designed as a 64-bit architecture (with 32-bit mode as a functional subset), and 64-bit capability is not an adaptation or remodeling of an existing 32-bit architecture. This design also makes binary compatibility easier to maintain. Binary compatibility with the current processors is an important aspect of the 64-bit version of PowerPC. From the standpoint of the 32-bit and 64-bit specifications, there are several differences. As shown in Figure 11 on page 34, the number of General Purpose Registers (GPR) remains the same, but these registers are 64 bits long instead of 32 bits long. A few other control registers move from 32- to 64-bits in length. As shown, floating point registers always remains the same, as they conform to industry standards for floating-point, which require 32-bit or 64-bit length data.

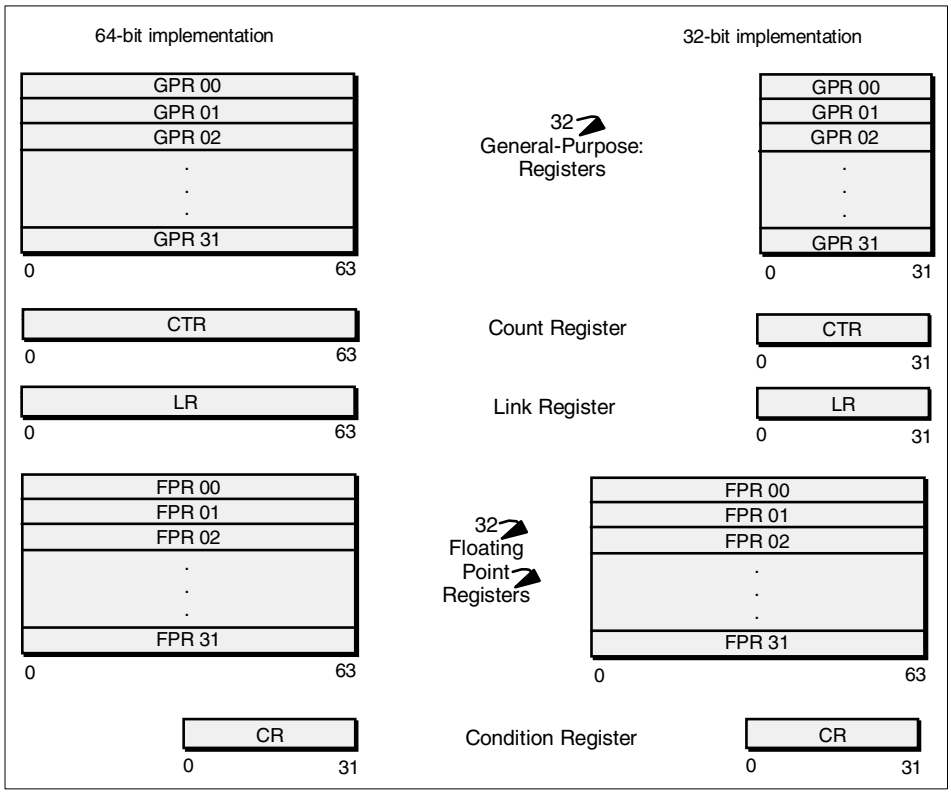


Figure 11. Register Implementation of 32 and 64-bit PowerPC Processors

2.4.2 Addressability

Addressability is the most important aspect, as complex applications (large databases, large numeric applications, and multimedia environments) will need operate on larger data sets. Table 2 shows the size of the address space that can be managed as a function of the length of the address that the CPU generates. A 64-bit architecture can address a huge address space.

Table 2. Size of Address space

Address Length	Address Space
8 bit	256 bytes
16 bit	64 kilobytes
32 bit	4 gigabytes
52 bit	4,000 terabytes

Address Length	Address Space
64 bit	16,384,000 terabytes

2.4.3 Advantages of 64-bit architecture

The following are the key benefits of the 64-bit architecture:

2.4.3.1 Large file support

The ability to address data in files larger than 2 GB requires that a program be able to specify file offsets larger than a 32-bit number. This capability is generally considered to be 64-bit computing function, even though it does not require 64-bit hardware support. AIX Version 4.2 provided this capability for 32-bit programs, and AIX Version 4.3 provides it for 64-bit programs as well. Because it does not depend on 64-bit hardware, this function can be used on any RS/6000 system running the appropriate release of AIX. There is, however, synergy between large file support and 64-bit hardware capabilities, in that a 64-bit program can have much larger portions of 64-bit files in its address space, as well as in system memory, at one time, than a 32-bit system could provide. Application enablers, such as DB2 UDB for AIX, support this capability. Advanced applications using high-quality graphics and sound or managing huge databases are going to become more common. They will greatly benefit from systems that can address and manage such huge amounts of data without forcing the application to worry about limits in data size.

2.4.3.2 Large physical memory support

Sufficient system memory is crucial for sustaining overall system performance. As a system's processor capacity grows with the speed and number of processors in the system, so does the requirement for system memory. Memory is a key element of having *balanced resources*, a requirement that applies whether the workload consists of 32-bit or 64-bit applications. For many customer environments, system memory capacity beyond 4 GB will be needed for optimum performance on an 6-way Model S80, with even more needed on a 12-way configuration. The scalability introduced by 64-bit technology is the opportunity for some programs to keep very large amounts of data in memory, both resident in physical memory and accessible in their 64-bit virtual memory address space. While exploiting this capability can significantly improve performance for some applications, there are relatively few types of applications that have evolved to make use of such techniques today. Those that have, however, most often make use of very large memory, that is, multiple gigabytes of system memory, just for one application program.

2.4.3.3 Large 64-bit application virtual address spaces

In 32-bit systems, an individual program or process may typically have between 2 GB and 4 GB of virtual address space for its own use to contain instructions and data. With 64-bit computing, applications may run in a 64-bit address space, where an individual program's addressability becomes measured in terabytes (TB). Some database management programs use a large address space for scalability in order to maintain very large data buffers in memory, reducing the amount of disk I/O they need to perform. Using a large address space, they can supply data to client applications at the pace needed to sustain the high transaction rate potential afforded by many of the new processors in the industry. In certain cases, database management programs or customer applications may benefit from keeping an entire database or large file immediately accessible in memory. Read-only data lends itself most readily to this scenario. Significant improvements in response time or transaction rates are possible. Certain types of applications are able to directly attack larger problems by organizing larger arrays of data to be computed upon. Computer simulation of a physical phenomenon, such as aircraft flight or a nuclear reaction, are frequently cited examples.

2.4.3.4 64-bit integer computation

The ability to use very long integers in computations is a feature that can be very helpful in specialized applications. Some applications need to deal with integers or bit strings larger than 32 bits.

For example, programs that perform data matrix manipulation can deal with large sets in potentially half as many references and logical/arithmetic operations as before. Programs that perform software operations on bit areas (graphics) in virtual storage can deal with twice as much data per operation as before.

2.4.4 Performance of 64-bit architecture

Although 64-bit architectures deliver all the benefits just described (larger programs, data, files, and physical memory), a common misunderstanding is that a 64-bit processor per se increases performance. This is not so for the following reasons:

- The larger address space means that larger applications can be developed and executed, without relation to performance.
- The support for larger physical memory gives no advantages without having a larger memory module. Having large real memory for huge applications will reduce paging, an issue more related to software than to hardware.

- The ability to manage larger disk spaces is a great functional and programming advantage if the software is modified to exploit this ability. In itself, 64-bit hardware will not influence performance.
- A 64-bit architecture can do 64-bit arithmetic, so if an application needs this function, programmers can avoid writing a library. Again, this ability is an indirect, software related, performance improvement. It is of no benefit to most integer calculation, because text is 8-bit, audio is 16-bit, and full-color graphics is 24- or 32-bit. It may have advantages in video compression, multimedia, and cryptology.
- It will double the size of every pointer and address, which makes the operating system and the programs larger. In other words, the memory requirement will be higher. This will impact cache, bus, and memory, creating demand for a higher memory bandwidth. This may actually decrease performance if it is not designed very well.

2.4.5 Software considerations for 64-bit architecture

Software or application exploitation in 64-bit systems is not automatic. Several things have to be considered:

- The operating system needs to support the 64-bit processor. This enables support for more than 4 GB memory.
- Compilers need to incorporate 64-bit support.
- Applications need to be 64-bit enabled if they want to take advantage of 64-bit architecture. If the application is written in C language, there are several things to be taken care of, for example:
  - Specific integer data types will be 64-bit, as defined by industry standards.
    - Pointer size is 64 bits in the 64-bit environment
    - *long* data type is 64 bits in the 64-bit environment
  - C data structures will be aligned to 64-bit.

2.4.6 64-bit operating system capabilities

Building a 64-bit machine around PowerPC processors is fairly straight forward, so there is no lack of capable hardware. However, to really exploit large size processors, the entire system has to support 64-bit.The AIX operating system enhancements exploit large memory, exposes a standard-compliant 64-bit Application Programming Interface (API) to applications and middle ware, and maintains binary compatibility with current 32-bit applications that are able to run concurrently.

The UNIX98 specification includes 64-bit computing features without actually defining it, and does not specify any dependency on 64-bit hardware. The specification defines a programming environment for large files that allows both 32-bit and 64-bit programs to have this capability. It also cleans up a few APIs that carried implications of 32-bit data types, allowing a 64-bit programming environment the same opportunity to have standard conformance as a 32-bit programming environment. The nature of the specification is such that a conforming 64-bit environment might exist on a system that supports only 64-bit programs, or on a system that supports other environments (such as 32-bit binary compatibility) in addition to 64-bit programs.

A system with 64-bit computing will complement 32-bit computing in different ways for different customers, and each customer will exploit the various elements of 64-bit computing at different speeds.

Chapter 3. IBM RS/6000 and IBM pSeries architectures

In February 1990, IBM introduced the first RISC System/6000 (RS/6000) with the first Performance Optimization With Enhanced RISC (POWER) architecture. Since that date, several POWER architectures have been designed for the RS/6000 models.

In 1991, with the alliance of Apple and Motorola, IBM started a plan for the future that would span a range from the small, battery-operated computer to very large supercomputers and mainframes. The PowerPC family of microprocessors, a single-chip implementation jointly developed by Apple, IBM, and Motorola, established a rapidly expanding market for RISC-based hardware and software. IBM has several successful lines of PowerPC-based products for workstations and servers. Motorola introduced a broad range of desktop and server systems, and other companies such as Bull, Canon, and FirePower have announced or shipped PowerPC-based systems. Apple has Power Macintosh systems, and companies such as Daystar, Pioneer, Power Computing, and Radius also have announced Power Macintosh-compatible systems.

With these successes the alliance ended, leaving IBM to continue building on it's CPU architecture and design, which can be seen with the introduction of the powerful copper technology deployed in the S80 servers.

3.1 POWER2 Super Chip

In October 1996, IBM announced the RS/6000 Model 595. This was the first machine to be based on the POWER2 Super Chip (P2SC) processor. As its name suggests, this is a single chip implementation of the POWER2 architecture, enabling the clock speed to be increased further. Currently the P2SC processors are employed only in the RS/6000 SP Thin4 nodes, where they run at clock speed of 160 MHz with a theoretical peak speed of 640 MFLOPS.

The POWER2 Super Chip (P2SC) is a compression of the POWER2 eight-chip architecture into a single chip with increased processor speed and performance. It retains the design of its predecessor, the POWER2.

The initial models had clock speeds of 120 MHz and 135 MHz. High-density CMOS-6S technology allows each to incorporate 15,000,000 transistors.

The most significant change is a halving of the size of the data cache and the data TLB, which now are 128 KB and 256 KB, respectively. These changes were required to fit the eight-chip processor onto a single chip.

The P2SC delivers the processing and dual floating-point power needed for large, numeric-intensive tasks as well as the integer and transaction performance for commercial applications. The P2SC contains on-chip 32 KB instruction cache and 128 KB data cache, and is full binary compatible with the POWER2 architecture

SP2 Thin nodes are the only current systems that use the POWER2 chips.

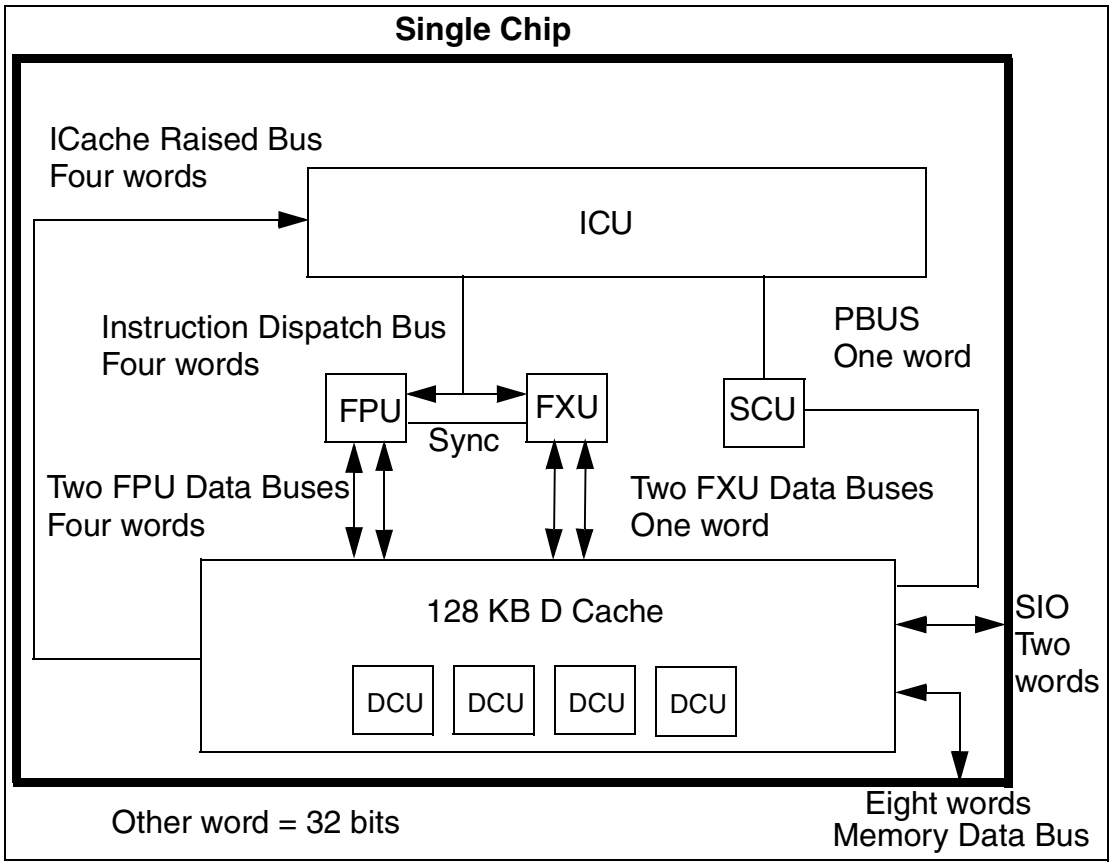


Figure 12. POWER2 Super Chip Module

3.2 POWER3

The POWER3 microprocessor introduces a generation of 64-bit processors especially designed for high performance and visual computing applications. POWER3 processors are the replacement for the POWER2 and POWER2 Super Chips (P2SC) in high-end RS/6000 workstations and technical servers.

The POWER3 microprocessor is a single chip implemented with 0.25 micron CMOS technology. It operates at a 200 MHz clock cycle. The POWER3 microprocessor has eight execution units, and allows concurrent operation of fixed point instructions, load/store instructions, branch instructions, and floating point instructions. The processor can dispatch up to four instructions at a time and execute them out of order, but is designed to ensure in-order completion and precise interrupts to provide program integrity. There is a 32 KB instruction cache and a 64 KB data cache on the chip, both parity protected. There is 256-bit external interface to a 4 MB L2 cache, which operates at 200 MHz and is ECC protected (Single Error Correction, Double Error Detection).

The POWER3 processor was designed to provide high performance floating point computation. For example, there are two floating point execution units, each supporting a 3-cycle latency, 1-cycle throughput Multiply-Add execution rate. This allows the POWER3 to execute four floating point operations per cycle, resulting in a peak throughput of 800 MFLOPS. The POWER3 processor essentially brings together the POWER2 architecture, as currently implemented in the P2SC processor, with the PowerPC architecture. It combines the excellent floating-point performance delivered by P2SC's two floating-point execution units, while being a 64-bit, SMP-enabled processor ultimately capable of running at much higher clock speeds than current P2SC processors.

The POWER3 implementation of the PowerPC architecture provides significant enhancements compared to the POWER2 architecture. The SMP-capable POWER3 design allows for concurrent operation of fixed-point instructions, load/store instructions, branch instructions, and floating-point instructions. The POWER3 is designed for ultimate frequencies of up to 600 MHz when fabricated with advanced semiconductor technologies such as copper metallurgy and silicon-on-insulator (SOI). In contrast, the P2SC design has reached its peak operating frequency at 160 MHz. The first POWER3 based system, RS/6000 43P 7043 Model 260, runs at 200 MHz.

Capable of executing up to four floating-point operations per cycle (two multiply-add instructions), the POWER3 maintains the emphasis on floating-point performance and memory bandwidth that has become the

hallmark of POWER2 based RS/6000 systems. Integer performance has been significantly enhanced over the P2SC with the addition of dedicated integer and load/store execution units, thus improving its SPECint95 performance relative to the 160 MHz P2SC by about 50 percent at 200 MHz. This gives the POWER3 far more balanced performance, which is especially notable in graphics intensive applications.

The POWER3 is a 64-bit PowerPC implementation with a 32-byte backside L2 cache interface (private L2 cache bus), and a 16-byte PowerPC 6XX bus, as shown in Table 13 on page 43. The POWER3 has a peak execution rate of eight instructions per cycle (compared to six for the P2SC) and a sustained performance of four instructions per cycle.

Significant investments in the chip's data flow, instruction routing, and operand buffering have been made in order to sustain a high computational and corresponding data rate. The POWER3's level-one (L1) data cache is an efficient interleaved cache capable of two loads, one store, and one cache line reload per cycle. Although half the size of the P2SC's cache, the L1 is effectively supplemented by a dedicated second level (L2) cache, which may be from 1 MB to 16 MB in size. Data and instruction prefetching mechanisms improve the memory access performance by hiding memory latency. Also, the large 128 byte line size takes advantage of the locality of reference (spacial reuse) characteristic of large engineering and scientific data reference patterns.

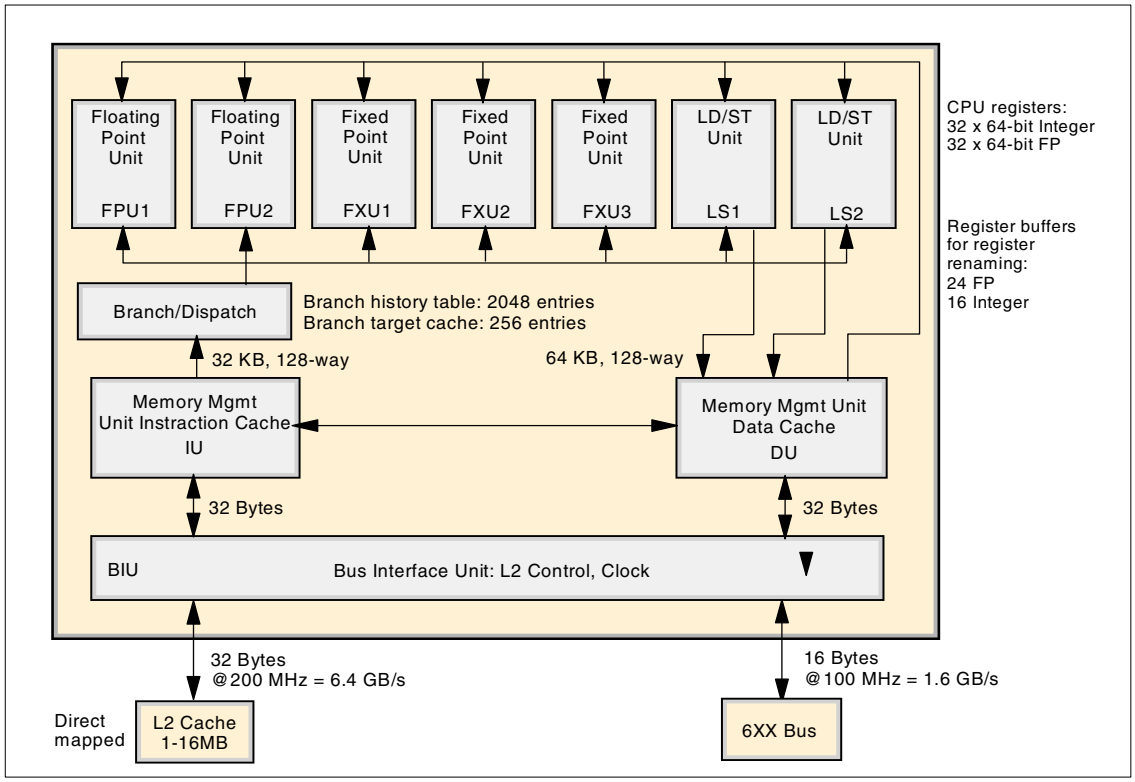


Figure 13. POWER3 Processing units (Model 260)

3.2.1 POWER3 execution core

Unlike some competitive chips, which need several pipeline stages before instructions enter the first execution stage, POWER3 keeps this front end of the pipeline short, using only three stages. POWER3 needs only one cycle to access the instruction cache, one cycle to decode and dispatch the instructions to different execution units, and one more cycle to access the operands. POWER3's relatively short pipeline keeps its mispredicted branch penalty to only three cycles, up to 24 cycles shorter than its competitors. Up to eight instructions (two floating-point, two load/store, two single-cycle integer, a multi-cycle integer, and a branch) can be in execution in each cycle. Ready instructions are issued out of order from the issue queues, allowing instructions of different types, as well as of the same type, to execute out of order. The load/store and branch instructions are issued in program order.

For branch instructions whose conditions are not known in the decode stage, POWER3 uses a 2,048-entry branch history table (BHT) to predict the branch direction. Because a branch is often resolved in the decode stage or soon

thereafter, the benefit of the BHT when used to predict the current encounter of the branch is less in POWER3 than in designs with deeper pipelines. To better use the BHT, however, POWER3 uses the BHT to predict both the current and the next encounter of each conditional branch, using a branch target address cache (BTAC).

POWER3 uses rename registers for the general-purpose registers (GPR), floating-point registers (FPR), and the condition-code register (CCR) to allow out-of-order and speculative execution of most instructions. The few exceptions are stores and certain move-to-special-register instructions that are difficult to undo. Although instructions can be issued out-of-order, and thus their operands can be read out-of-order from the registers, the rename registers eliminate anti- and output-dependencies by enabling the registers to be updated in program order.

POWER3 has two identical FPUs, each delivering up to two floating-point operations per cycle. POWER3's FPUs execute multiply-add instructions, as shown in Table 3, taking only one cycle throughput to calculate the frequently used (a*b+c) operation.

Table 3. POWER3's Low Execution Latencies

Instruction	Number of 32-bit Cycles	Number of 64-bit Cycles
Integer Multiply	3-4	3-9
Integer Divide	21	37
FP Multiply or Add	3-4	3-4
FP Multiply-Add	3-4	3-4
FP Divide	14-21	18-25
FP Square Root	14-23	22-31

3.2.2 Memory access section

The non-blocking caches support four outstanding L1 data demand requests and two outstanding L1 instruction demand requests in order to reduce the memory subsystem latency. The L1 cache also supports hits under misses; it allows a fifth demand request to proceed even when there are four previous outstanding misses to the data cache. In comparison, the POWER2 architecture allows only one outstanding cache miss without blocking. Cache hits are satisfied within a single cycle. The writeback data cache implements a four-state MESI cache coherence protocol (possible states: **m**odified, **e**xclusive, **s**hared, and **i**nvalid) to support SMP environments.

POWER3 uses instruction- and data-prefetch mechanisms to reduce pipeline stalls due to cache misses. The instruction cache is two-way interleaved on cache-line boundaries, allowing one bank to be accessed for instruction fetches while the other bank is accessed for the next cache line. When the former access hits in the cache but the latter access does not, a prefetch request for this next cache line is issued to the L2 cache. Because the prefetch is still speculative, the request is not propagated to the main memory. If it misses in the L2 cache, this allows the request to be canceled upon detecting a mispredicted branch instruction. An instruction prefetch takes six cycles from the 200 MHz L2 cache.

For the data cache, the Model 260 can prefetch up to four streams of data from memory or L2 cache into L1 cache. To establish a prefetch stream, the prefetch mechanism monitors every access that misses in the data cache, searching for cache-miss references to two adjacent cache lines. For this purpose, a stream address filter queue of depth 10 is used, which contains the guessed next stream addresses. The filter is maintained by a least recently used (LRU) mechanism in order to age out seldom used prefetch streams. Upon finding such a pair of succeeding cache misses, it initiates a prefetch request for the next cache line. The stream addresses, along with the ascending or descending prefetch direction, is kept in a four-entry stream address buffer. Once a prefetch stream is identified, the address of every data-cache access is checked with the addresses in the stream address buffer. When a match is found, a prefetch request for the next cache line is made, and the address in the matching entry is updated with the address of the new prefetch request. A simplified view on the prefetch hardware is given in Table 14 on page 46.

When initially predicting the direction of a prefetch stream, it is assumed that if the word that causes the cache-miss occurs in the bottom half of the cache line, the next higher line will be required, but if the miss occurs in the top half, then the next lower line will be required. Then data is being prefetched in sequentially in either a forwards or backwards direction. If the initial prediction is wrong, the direction is corrected for the subsequent stream.

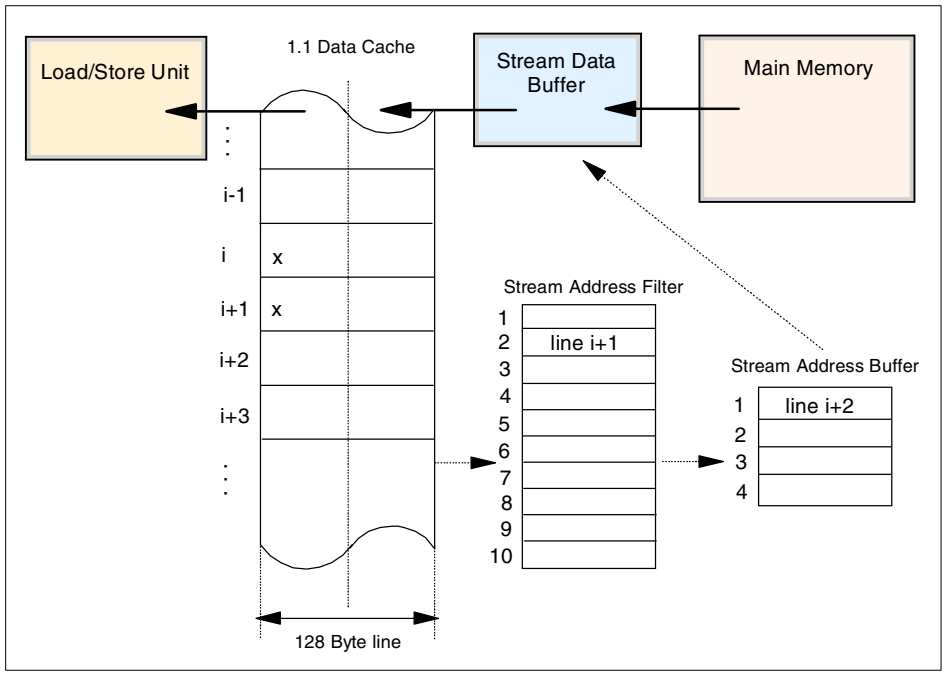


Figure 14. Data prefetch overview

The 64-bit address space is managed by using 80-bit virtual addresses and 40-bit real memory addresses, which support up to 1 terabyte. A 256-entry two-way set associative translation lookaside buffer (TLB) based on a least recently used replacement algorithm is used to access 4 KB memory pages.

The performance of many technical applications is mainly determined by the performance of the memory subsystem. POWER3 systems are designed to deliver industry leading memory bandwidth, which has already been a strength of the POWER2 architecture. The bandwidth, as listed in Table 4 on page 47, in terms of GB/s depends on the actual clock frequency. As an example the DAXPY (subroutine that computes a constant times a vector plus a vector) operation, y(i)=y(i)+a*x(i), yields a sustained memory bandwidth of 1.3 GB/s, close to the peak bandwidth of 1.6 GB/s of a POWER3 Model 260 system.

The load latency, due to either a data or instruction L1 miss that hits the L2 cache, amounts nine CPU cycles. A data access that misses the L1 and L2

cache causes a latency of about 35 cycles on a Model 260. However, this does not depend on the processor only, but also on the system.

Table 4. RS/6000 43P 7043 Model 260 Memory Bandwidth

Access	Interface Width (Bit)	Clock Frequency (MHz)	Bandwidth (Byte/cycle)	Bandwidth (GB/s)
Load Register from L1	128	200	2*8	3.2
Store Register to L1	64	200	8	1.6
Load/Store L1 from/to L2	256	200	4*8	6.4
Load/Store L1 from/to Memory	128	100	2*8	1.6

3.2.3 POWER 3 II chip

The POWER3 II is a third generation superscalar design that is used for 64-bit technical and scientific applications. The processor functional diagram of the POWER3 and the POWER3-II are similar. However, the use of copper in the POWER3-II represents a new generation of processing power. Table 5 on page 48 lists some of the differences between the POWER3 and the POWER3-II processors. Also, the chart indicates the direction being taken by this technology. The number of transistors is increased in the POWER3-II due to the enhanced L2 cache controller and minor updates to handle different bus requirements.

The POWER3 II processor supports the following characteristics:

- Up to 8 instructions per cycle.
- 64-bit SMP implementations
- 23 million transistors
- 0.22 micron lithography
- 6 layers of metal copper interconnect
- Hardware memory prefetch
- Operation within 333 MHZ- 400 MHz
- Caches:
  - Integrated L1

        - 32 KB Instruction cache, 128-way, 128 B line
        - 32 KB Data cache, 128-way, 128 B line
    - Integrated 4/8 MB L2 controller
        - 32 B/beat
        - 32 B bus @ 200-250 MHz (6.4-8.0 GB/s bandwidth)
- PowerPC 6XX Bus
    - 16 B bus @94-100 MHz (1.5-1.6 GB/s bandwidth).

Table 5. Differences between POWER3 and POWER3-II Processors

Description	POWER3	POWER3-II
Chip Die Size	270 mm ^{2}	163 mm ^{2}
Transistors	15 million	23 million
Power Avg/Max	Power Avg/Max 39W/46W@200 MHz	26W/33W@375 MHz
CMOS Technology	6S2, 5 layers metal	7S, 6 layers metal, copper interconnect
Lithography	0.25 µm	0.22 µm

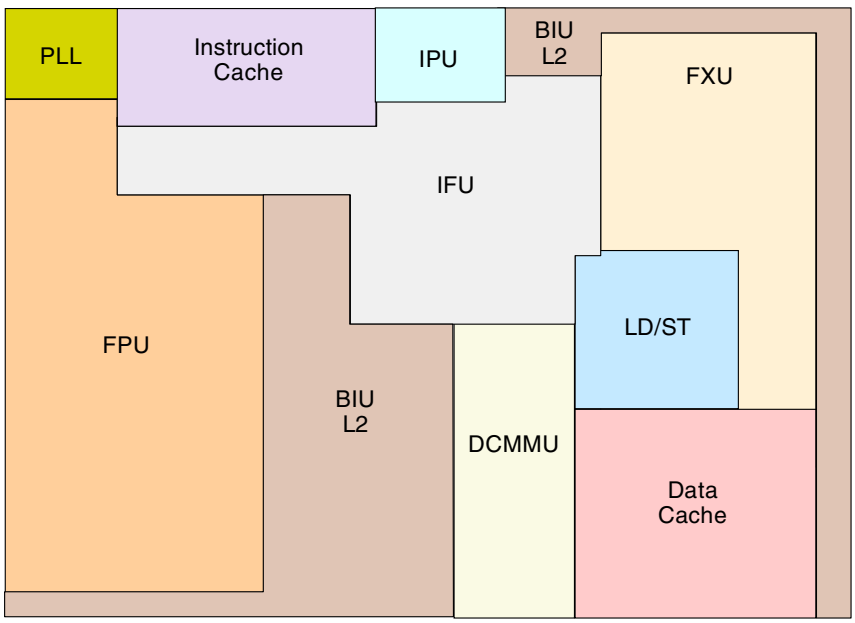


Figure 15. The POWER3 II Processor

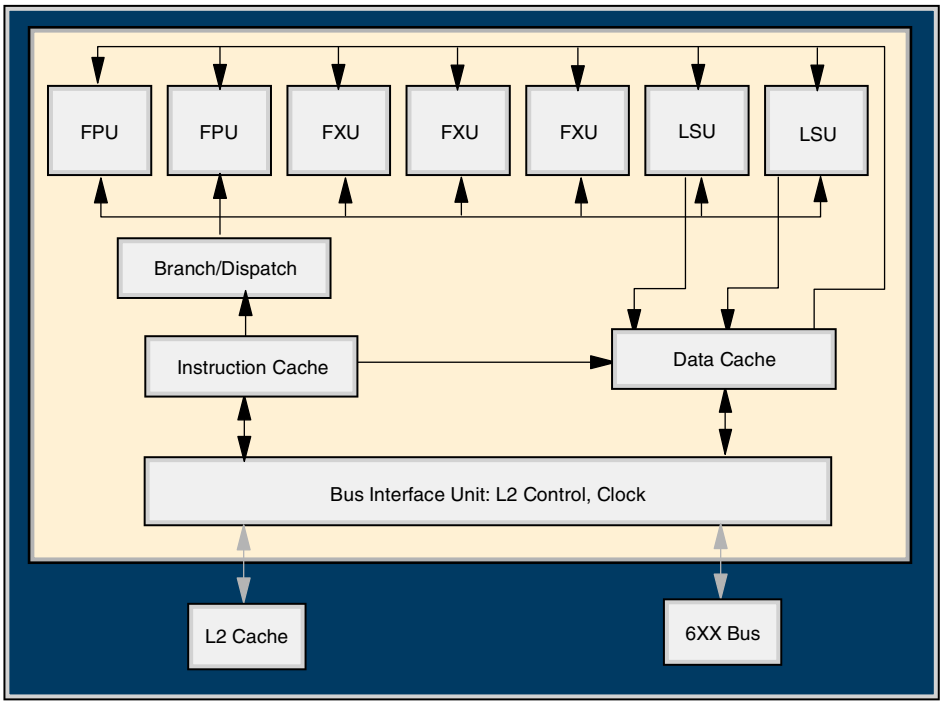


Figure 16. POWER3 II Processor

3.2.3.1 Copper and CMOS technology

Copper is a superior conductor of electricity, making it possible to shrink the electronic devices even further while increasing performance. It has less resistance than aluminum and, therefore, allows designs that transmit electrical signals faster. However, it does not mix as well with silicon, the base material of semiconductor chips. The IBM researchers found a way to put a microscopic barrier between the copper and silicon in a way that actually reduced the number of steps needed to complete a chip. With this development, IBM is able to squeeze down the widths of copper wires to the 0.2-micron range from the current 0.35-micron widths - a reduction far more difficult for aluminum. A single POWER3-II chip contains about 400 meters of copper wiring.

This technology, called *CMOS 7S*, is the first to use copper instead of aluminum to create the circuitry on silicon wafers. Copper wires conduct electricity with about 40 percent less resistance than aluminum. That translates into a speed up of up to 15 percent in processors that contain copper wires.

3.3 PowerPC

The PowerPC family of processors was started by the alliance between Apple and Motorola, IBM in 1991. This alliance established a rapidly expanding market for RISC-based hardware and software. The section discusses the PowerPC processors in use today.

3.3.1 PowerPC 604 and 604e

The PowerPC 604 microprocessor family is a 32-bit implementation of the PowerPC. The 604 gives the performance needed to support graphics, computation, and multimedia-intensive applications. The early 604 implementation reaches new levels of performance by issuing four instructions per cycle, thus achieving balanced execution of integer and floating-point operations.

The 604 uses a superscalar design to provide six independent execution units: one branch unit, three fixed-point units, one floating-point unit, and a load/store unit. The 604 chip also uses dynamic branch prediction techniques to enhance instruction pre-fetching as well as speculative execution techniques to take advantage of the improved instruction pre-fetching and multiple execution units.

On-chip, the 604 features 16 KB instruction and 16 KB data caches coupled to a high-performance, 64-bit system bus. The microprocessor takes advantage of instruction-level parallelism found in today's application programs.

The 604 fetches, dispatches, and completes up to four instructions per cycle. It can hold up to eight instructions for dispatch and 16 more in various stages of execution. Of the six execution units, three are pipelined to sustain a four-instructions-per-cycle rate for those applications that offer a high degree of parallelism, and a total of six pipeline stages are used to achieve its 100 MHz initial design. The stages are: *fetch, decode, dispatch, execute, complete,* and *write-back*.

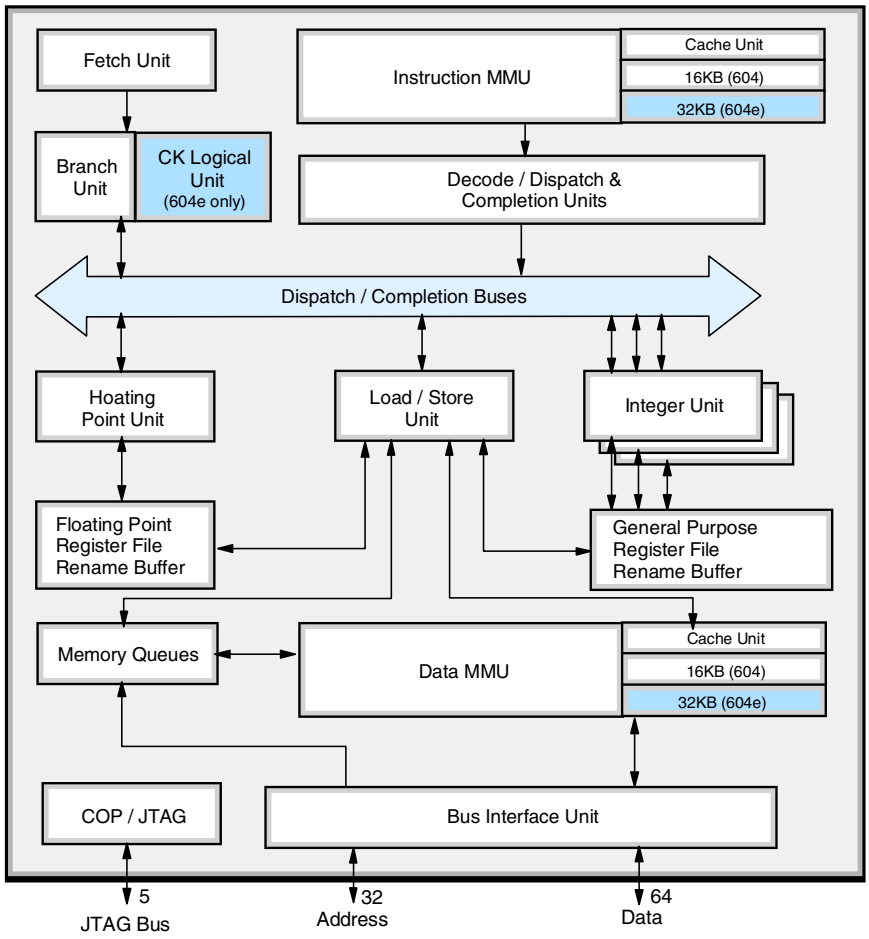


Figure 17. PowerPC 604 and PowerPC 604e Block Diagram

3.3.2 Differences between 604 and 604e processors

There are many ways to increase performance from microprocessors. Of all the techniques available, the migration from PowerPC 604 to PowerPC 604e takes advantage of two. First, the PowerPC 604 was re-mapped into a higher performance CMOS technology. That alone increased speed just from the physics of the transistor design. Second, the PowerPC 604e design team adopted a philosophy of identifying and reducing internal bottlenecks. Along the way, the main intent for both techniques was maintaining backwards compatibility. The result; the PowerPC 604e is a truly seamless migration of

the PowerPC 604 core architecture into a state-of-the-art CMOS technology that improves the internal flow of instructions without impact to PowerPC 604 software or hardware implementations. The *e* was added to the PowerPC 604 name to emphasize the enhancements over its predecessor.

Externally, PowerPC 604e is fully compatible with PowerPC 604. Internally, the block diagrams look almost the same. The difference is the enhancements, which include:

- Full hardware support of misaligned Little-Endian accesses
- 32KB split instruction and data caches (double the size of PowerPC 604)
- Data cache line-fill buffer forwarding; additional cache copy-back buffers
- Additional processor/bus ratios
- No DRTRY mode
- Performance monitor enhancements
- Coherent instruction fetch mode
- Split Branch and Condition Register execution units

In almost every case, the design philosophy was to examine areas where instructions or information was getting held up, and invisibly eradicate the cause for the bottlenecks. In addition to the above list, a mode was added to the bus interface unit for timing purposes. The idea of the mode, called out in the CPU specifications as *Fast Out mode*, was to provide more time in each bus cycle to accommodate faster bus timings. While this is different from PowerPC 604, the default mode of the PowerPC 604e bus is fully compatible with existing PowerPC 604 designs.

3.3.3 RS64 II processor

IBM's RS64 II superscalar RISC microprocessor integrates high-bandwidth and short pipe depth with low latency and zero cycle branch mispredict penalty into a fully scalable 64-bit PowerPC-compatible symmetric multiprocessor (SMP) implementation. Based on PowerPC architecture, the first in the RS64 Series of microprocessors, the RS64 II processor contains the fundamental design features used in the newly available RS/6000 server systems targeted at leading edge performance in commercial applications.

3.3.3.1 Design point

The RS64 II microprocessor design objectives were to provide more performance, reliability, and functional robustness than previous 64-bit PowerPC commercial/server processor designs while reducing product and development costs.

The basic design philosophy was to reuse as much as possible from the previous design point, adding enhancements only if they were simple or resulted in significant improvements to the design objectives. The result of this approach is a microprocessor with the following attributes:

- 4 way superscalar
- 5 stage deep pipeline
- Branch mispredict penalty of zero or one cycle
- 64 Kilo Byte (KB) on-chip Level One (L1) instruction cache
- 64 KB on-chip L1 data cache with one cycle load-to-use latency
- Support for up to a 32 Mega Byte (MB) Level Two (L2) cache with a 5 cycle load-to-use latency
- 8.4 Giga Byte (GB) per second L2 cache bandwidth
- 32 byte wide on-chip busses
- 262 MHz operating frequency
- 162 mm^2 die size
- 27 Watts maximum power

Major differences from the previous design point include switching from BiCMOS technology into CMOS technology, consolidating the processor from five chips into one, reducing the power by a factor of 5, and adding support for an external L2 cache.

The technology strategy of the RS64 II design was to produce a high performance low cost microprocessor in an advanced, but well-established technology (CMOS6S2). A well defined interface was developed between the processor core logic and the Bus Interface Unit, allowing for future reuse of the processor core with various upgrades and enhancements to the memory subsystem from the level two cache and beyond. The first derivation of RS64 II is called RS64 III. RS64 III leverages IBM's cutting-edge semiconductor copper technology (CMOS 7S) to increase the operating frequency to 450 MHz. The higher densities available with this technology also permitted doubling the on-chip L1 instruction and data cache sizes to 128 KByte each.

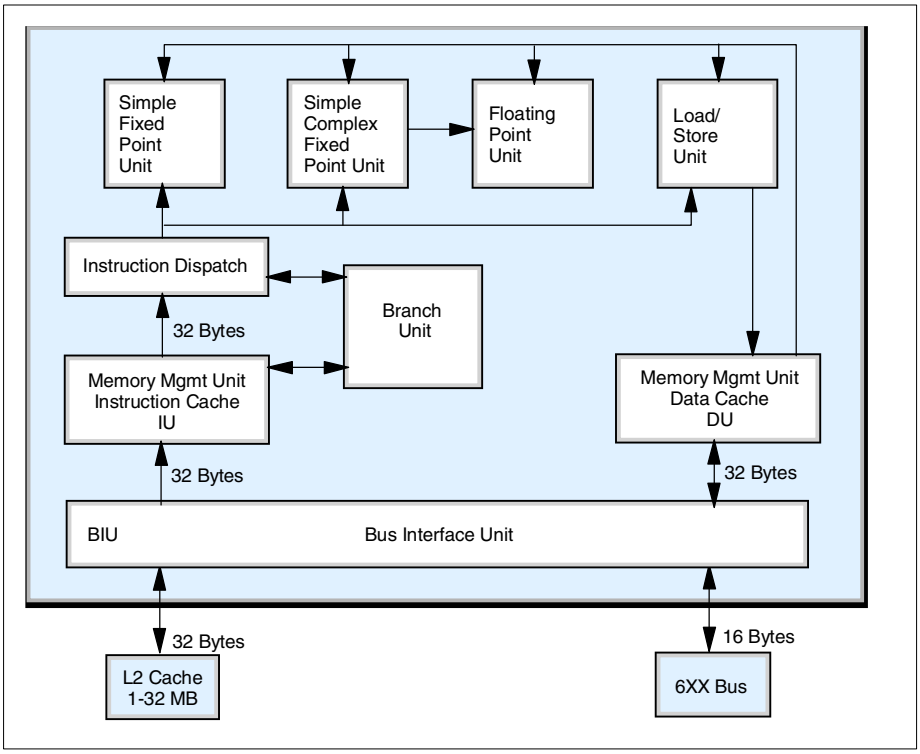


Figure 18. RS64 II block diagram

3.3.3.2 Processor overview

The RS64 II processor block diagram shown in Figure 18 focuses on server performance with emphasis on conditional branches with zero or one cycle mispredict penalty, contains 64 KB L1 instruction and data caches, has a one cycle load-to-use penalty on the L1 data cache, enhanced string support, and four superscalar fixed point and one floating point pipelines. There is an on board bus interface unit (BIU) that controls both the L2 cache interface and the main memory bus interface.

3.3.3.3 Description of pipe stages

Figure 19 on page 55 is a pictorial view of the five RS64 II pipe stages described in the following text.

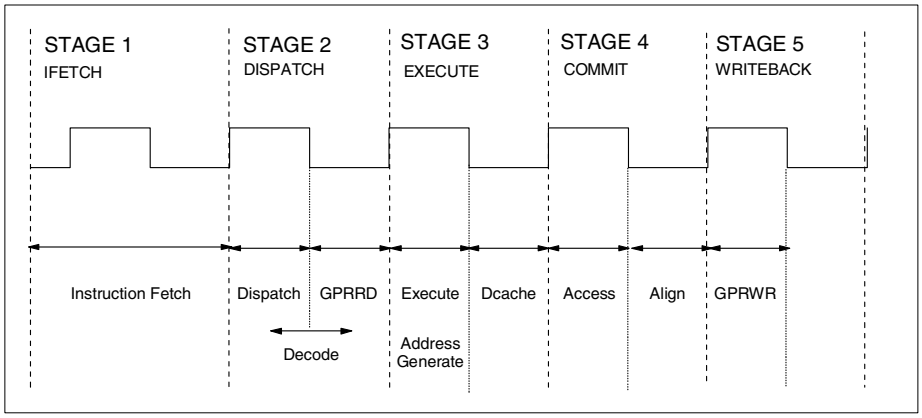


Figure 19. RS64 II's Series Pipeline

- Instruction Fetch stage:

  In the instruction fetch stage, the L1 instruction cache array is accessed with the address generated by the branch unit and 32 bytes of instructions are output and written into either the 16 entry sequential instruction buffer or the eight entry branch buffer. The sequential instruction buffer and the branch buffer are used for conditional branch processing, as explained in Section 3.3.3.4, "Branches and instruction cache address generation" on page 56.

- Dispatch stage:

  The dispatch stage is responsible for decoding and dispatching up to four 4-byte PowerPC instructions each cycle. Instructions are dispatched in order from either the sequential instruction buffer or the branch buffer. During dispatch, operands are read from architected registers, completion buffers, and result busses. The instruction cache branch target address is generated in the dispatch stage unless there are dependencies on instructions that have not yet executed.

- Execute stage:

  During the execute stage, the arithmetic, rotate, and data cache address generation functions are performed. All results are bypassable to all execution units for use in the next cycle by subsequent instructions. Condition register lookahead logic based on the arithmetic and rotate zero detect and sign bits bypasses into the conditional branch logic. If any of the input operands are invalid due to dependencies, then the execute stage for that pipeline stalls.

- Commit stage:

  The commit stage holds execution results for each pipeline. Taken branches, exceptions, and page faults can cause commit (and execute) results to be discarded. Commit stage results are bypassable to the execution stage. Cache fetch data is bypassable to the execution stage through an aligner.

- Writeback stage:

  The writeback stage of the pipeline writes the instruction results into architected registers once all branch and exception conditions have been resolved.

3.3.3.4 Branches and instruction cache address generation

When it comes to keeping the pipeline full of instructions, conditional branches pose a special problem. Many processor designs solve this problem with branch prediction logic. Software code executed in the commercial environment has fewer code loops, making branches harder to predict with high accuracy. The RS64 Series of processors takes a different approach in solving the conditional branch problem by minimizing the branch mispredict penalty to zero or one cycle. This is accomplished with a combination of techniques.

The wide 32 byte instruction fetch path from the instruction cache paired with the 16 entry sequential instruction buffer and the 8 entry branch instruction buffer allow instructions at the branch target to be prefetched while instructions are being executed out of the sequential instruction buffer.

The first cycle of branch processing begins prior to the dispatch stage. The branch logic looks ahead, up to six instructions, into the dispatch queue to find a branch instruction. The first branch instruction found is decoded and its branch target address is generated. In the second cycle of branch processing, the instruction cache array is accessed and the aligned output is written into an instruction buffer.

By default, branches are assumed *not taken*, that is, instructions are dispatched and executed down the *not taken* or *sequential* path prior to the outcome of the branch instruction being known. This is equivalent to predicting branches as not taken. Once the outcome of the branch instruction is known and if the branch is taken, the instructions dispatched after the branch instruction are canceled. The *branch taken* logic switches dispatch from the sequential instruction buffer to the branch instruction buffer and cancels instructions dispatched down the *not taken* path. The *branch taken* logic is some of the most timing critical logic in the processor. The zero detect

and sign bits in the execution stage of the fixed point units are bypassed into the *branch taken* generation logic.

Branch penalty is defined as the time from dispatch of a branch instruction to the dispatch of the target of a branch instruction. When the instruction cache branch address is generated ahead of the dispatch stage and the branch condition is known at the end of the dispatch stage, there is no branch penalty. This is known as a *zero cycle* branch. When the Instruction cache address is generated during dispatch stage (instead of earlier) or when an instruction modifying the condition register is dispatched in parallel with the conditional branch, a one cycle branch penalty is incurred.

3.3.3.5 Fixed and floating point units

Two of the four superscalar units are fixed point units (FXUs) and have single cycle execution for the bulk of the integer arithmetic instructions. One of the two FXUs is specialized to also execute multi-cycle integer instructions, such as multiply and divide.

Although a commercial processor, it was deemed necessary to implement a simple and efficient pipeline for floating point arithmetic. The floating point unit (FPU) is fully independent and contains hardware for square root and division as well as for the fused multiply-add instruction. The FPU is fully pipelined with four cycle latency, single cycle throughput.

The load store unit includes a custom dynamic adder to allow for high speed cache address generation.

3.3.3.6 L1 data cache, L2 cache, and bus interface unit

Minimizing L1 data cache latency is key to high performance without complexity. The RS64 Series of processors are microarchitected so that the L1 data cache access has a 1 cycle load-to-use penalty. The L1 data cache was designed to be as large as possible without increasing the load-to-use penalty to 2 cycles.

The L1 data cache data bypasses directly into the execution units. It is 2 way set associative with a 16 byte interface to the execution units and a 32 byte interface for cache line replacement.

The L1 data cache was designed with four single port arrays. Chip area was saved by using single port arrays instead of multi-port arrays to increase the number of entries in the cache while minimizing cache access latency. Cache line replacements and stores normally done with a second cache port are accomplished by queuing them in a line fill buffer and a store buffer. The fills and stores are done either during background cycles when the instruction

stream is not accessing the data cache or simultaneously with instructions that operate on 8 bytes of data or less. The majority of instructions operate on bytes of data or less, and these instructions use at most one half of the available L1 data cache interface. The line fill buffer holds seven cache lines and has the characteristics of an L0 (Level Zero) cache in that any portion of an incoming line can be stored to or read from. A high speed bypass path around the line buffer exists for the first data transfer coming from L2 cache or main store going directly to the execution units.

The on-chip BIU contains the interface logic to support up to a 32 MB L2, 6XX system bus protocols, and dedicated hardware to hide latency to memory. While the RS64 II chip supports up to 32 MB of L2, a maximum 8 MB of L2 was installed on the first systems using the RS64 II processor due to considerations external to processor support. L2 latency is right behind L1 latency when it comes to impact on performance, so various innovative techniques were used to minimize the L2 load-to-use latency to a total of 5 cycles. L1 data cache accesses are speculatively forwarded to the L2 and canceled if an L1 cache hit is detected. The L2 SRAM clocking logic on RS64 II is designed to tolerate a wide range of access delays caused by SRAM process variation without adding latency to the access path. In the first incantation of the RS64 Series, both the L2 cache and L2 cache directory are implemented with SRAMs external to the processor chip. The external L2 cache is 4-way set associative. Associativity in the L2 results in higher L2 cache hit rates for most commercial workloads.

Instructions that gate SMP performance, such as those related to locks, TLB (Translation Lookaside Buffer), cache management, and synchronizing, are optimized for performance in the storage control microarchitecture. Lock information is bypassed between pipeline stages to prevent pipeline stalls. The TLB table walk routine is implemented in circuits instead of microinstructions to reduce table walk time. The cache coherency scheme implemented by the RS64 Series processors does not require synchronizing instructions to be broadcast on the system bus, minimizing the performance impact due to synchronization.

Containing 12.5 million transistors, the RS64 II processor die is shown in Figure 23 on page 69. It is manufactured in IBM's 0.35 micron hybrid CMOS 6S2 technology, with five levels of interconnect metallurgy.

3.3.3.7 System implementation

A key challenge of the RS64 II processor was to design a high bandwidth system interface required to support the high miss rates driven by commercial processing. RS64 II leveraged IBM's advanced packaging technology to

implement separate, independent 16 byte memory bus and 32 byte L2 bus, each with separate address, data, and control lines, achieving 8.4 GB (Giga Bytes) per second to the L2 at 262 MHz. This was achieved with a total of 2030 chip I/Os, of which 985 are signal I/Os.

The system interface is designed to allow flexibility in system implementation from low cost, bus-based systems to more complex switch-based configurations, providing greater address and data bandwidth.

The RS64 II processor design supports Modified Exclusive Shared Invalid (MESI) snoop-oriented SMP cache coherence along with remote processor bus protocols for increased throughput and large system topologies.

One characteristic of transaction processing is a high rate of data sharing between processors. The RS64 Series of processors provides improved performance in this environment by allowing cache lines to be transferred directly between processors with a technique called intervention. This results in shorter cache miss latencies compared to retrieving all L2 cache miss data from main store.

3.3.3.8 Scalability

The RS64 Series processors provide logic support for greater than a 12 way SMP. This capability will be leveraged in future systems.

3.3.3.9 Error correction, detection, and isolation

The commercial processing environment requires both high data integrity and high availability. On-chip arrays comprise the largest portion of chip area and are also the most susceptible to failures. For this reason, RS64 Series processors have built in recovery for single bit array failures. If an error is detected in the instruction cache, instruction cache directory, data cache directory, or the TLB, the entry in error is invalidated or marked unusable and its correct contents refetched. Separate L1 data directories for processor use and SMP snooping are implemented to provide adequate bandwidth. These separate L1 data directories are exact copies of each other, resulting in built in redundancy that is used to recover from errors in either directory.

The L1 data cache policy is store-in and may hold the only copy of modified data in the system. For this reason, the L1 data cache is implemented with an ECC scheme that can detect double bit errors and correct single bit errors. The off-chip L2 data cache and the L2 Directory are covered by the same ECC scheme as the L1 data cache.

Various parity schemes are integrated into the control and data flow logic on the processor chip. Whenever a recoverable or non-recoverable error is

detected, information pertaining to the error is recorded by the hardware and made available to the system diagnostics to isolate the failing circuits.

3.3.3.10 Performance

The RS64 II processor excels in real applications precisely because its many facilities combine to cover the wide spectrum of demands that characterize commercial computing. Applications may be limited by the rate of computational speed or by the rate of data delivery to the computational units. They are primarily fixed point intensive, but floating point was not ignored and performs quite acceptable for commercial applications without being a burden to the chip's cost and area. RS64 II's well balanced design handles these challenges with its five execution units, wide data paths and short pipe length.

3.3.3.11 Summary

In summary, the RS64 Series processors are very robust, delivering real performance on real applications for the next generation of 64-bit RISC commercial and server processors, all while retaining optimum chip size and power. It achieves high performance on real application because of its low latency design and IBM's superior silicon technology. The RS64 Series can be expected to lead the commercial and server benchmarks for years to come

3.3.3.12 Reference

Additional information may be obtained from the web:

`http://www.austin.ibm.com/resource/technology/nstar.html`

3.3.4 RS64 III processor

The RS64 III superscalar processors are optimized for commercial workloads. Target environments are characterized by heavy demands on system memory, both in the form of very large working sets and latency-sensitive serial dependencies. As a result, the design of the RS64 III processors, which run at 450 MHz, has focused on large cache sizes and data paths having high bandwidth and low latency.

The RS64 III processor has separate internal 128 KB L1 caches for instructions and data. Both L1 caches have doubled in size over the RS64 II. It contains an L2 cache controller and a dedicated 32-byte interface to a private 4-way set associative 8 MB L2 cache. The L2 interface runs at half processor speed, but transfers data twice per cycle to provide 14.4 GB/s of bandwidth. RS64 III internal data paths are 32 bytes wide. The RS64 III processor also has a separate 16-byte system bus interface.

The RS64 III has a total of five pipeline execution units. There is a Branch Unit, a Load/Store Unit, a Fixed-point Unit, a Complex Fixed-point Unit, and a Floating-point Unit. The processor has a 32-byte interface to the dispatch logic. There is a current instruction stream dispatch buffer that is 16 instructions deep. The RS64 III has an eight deep branch buffer. The RS64 III can sustain a decode and execution rate of up to four instructions per cycle.

All of the arrays in the RS64 III have either redundancy and ECC or parity and retry to support the high requirements desired for customer reliability, availability, and integrity. This enables full fault detection and correction coverage.

The RS64 III S80 processor card has six processors with an associated L2 cache contained on each card. There are 8 MB of L2 per processor. Each processor card has the six processors on a set of two SMP system buses and that dual bus interface is presented to the S80 backplane. The systems processor cards all need to use the same type and speed of processor.

The IBM RS64 III superscalar RISC microprocessor integrates high bandwidth, short pipe depth with low latency, large caches, and zero-cycle branch mispredict penalty into a fully scalable, 64-bit, PowerPC-compatible symmetric multiprocessor (SMP) implementation. The RS64 III processor contains the fundamental design features used in the newly available RS/6000 server systems targeted at leading-edge performance in commercial applications.

This paper provides an overview of how the processor microarchitecture, silicon technology, packaging technology, and systems architecture were leveraged to produce outstanding high performance in commercial applications and server markets.

3.3.4.1 Design point

The RS64 III is a 64-bit Commercial Processor and is the second in a line of microprocessors. The first microprocessor in this line is known on the RS/6000 system as the RS64 II.

The RS64 III microprocessor powers the RS/6000 S80 server systems. The RS64 III microprocessor design objectives were to provide more performance, reliability, and functional robustness than previous 64-bit PowerPC commercial/server processor designs while reducing product and development costs.

The *RS64 II* processor was used as a base for the design of the RS64 III. The operating frequency of the RS64 III processor was increased to 450 MHz over

its predecessor's 262 MHz debut. The increase in frequency was accomplished by leveraging IBM's new copper technology (CMOS 7S) along with redesign of timing critical paths. The size of the Level One (L1) instruction and data caches were doubled to 128 Kilobytes (KB) each. Innovative custom circuit design techniques were used to maintain the one cycle load-to-use latency for the L1 data cache. The branch mispredict penalty relating to the L1 instruction cache was also kept at zero or one cycle, as discussed later in this paper. The RS64 II processor's off-chip Level Two (L2) cache directory was integrated into the new RS64 III chip. A new IBM silicon technology with higher density used in conjunction with copper technology allowed these new functions to be added to the RS64 III processor chip while shrinking the die size from RS64 II's 162 mm^2 to RS64 III's 140 mm^2.

The silicon technology used by RS64 II requires a 2.5 Volt (V) power supply. IBM's new copper technology, used in the new chip, uses a 1.8 V power supply. The lower power supply voltage coupled with the smaller circuit dimensions resulted in 22 watts of maximum power at 450 MHz for RS64 III compared to its predecessor's 27 watts at 262 MHz.

A new generation of IBM's leading-edge SRAM technology was required to support the increased L2 data cache bandwidth requirements of the new processor. The L2 SRAM technology used for RS64 II was single data rate while IBM's new SRAM technology used by RS64 III is double data rate. Double data rate technology provides two transfers of data on the 32-byte wide L2 data bus every SRAM clock cycle. The L2 SRAM clock cycle time is 225 MHz, resulting in a L2 data cache bandwidth of 14.4 Gigabytes/Second (GB/s) for the new processor. The new SRAM technology also reduced L2 access latency as measured by nanoseconds. The reduced L2 SRAM access latency resulted in an L2 load-to-use latency of 7 cycles at 450 MHz.

The RS64 III processor has the following attributes:

- 128 KB on-chip L1 instruction cache
- 128 KB on-chip L1 data cache with one cycle load-to-use latency
- On-chip L2 cache directory that supports up to 8 MB of off-chip L2 cache
- 14.4 GB/s L2 cache bandwidth
- 32-byte wide on-chip busses
- 450 MHz operating frequency
- 140 mm^2 die size
- 22 watts maximum power
- 4-way superscalar
- 5 stage deep pipeline
- Branch mispredict penalty of zero or one cycle

The next step in the roadmap is to map the RS64 III design with its large caches into IBM's newest technology breakthrough called Silicon On Insulator (SOI) to create a microprocessor with a frequency in excess of 500 MHz. Laboratory hardware testing is currently underway for the first SOI-based systems. Plans exist for another *RS64 Series* microprocessor that uses a future IBM SOI technology with a target product frequency of over 600 MHz.

3.3.4.2 Processor overview

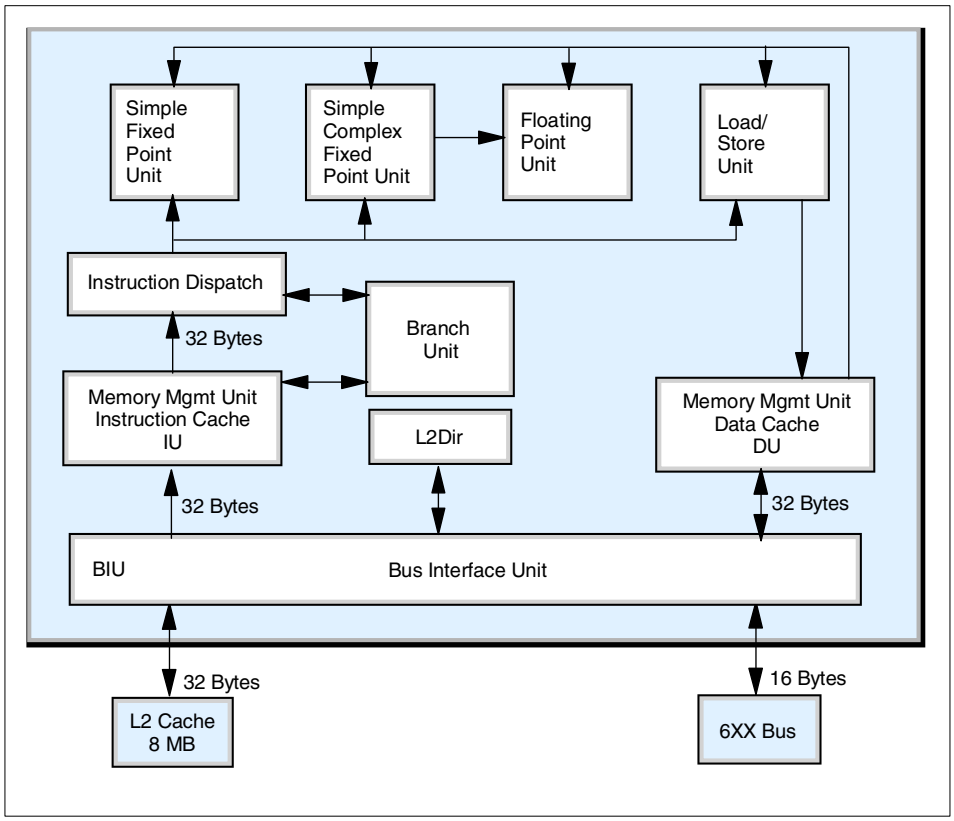


Figure 20. RS64 III Block Diagram

The RS64 III processor block diagram shown in Figure 20 focuses on server performance with emphasis on conditional branches with zero or one cycle mispredict penalty, contains 128 KB L1 instruction and data caches, has a one cycle load-to-use penalty on the L1 data cache, enhanced string support, and four superscalar fixed-point and one floating-point pipelines. There is an on board bus interface unit (BIU) that controls the L2 cache interface and the main memory bus interface.

Description of pipe stages

Figure 21 is a pictorial view of the five RS64 III pipe stages.

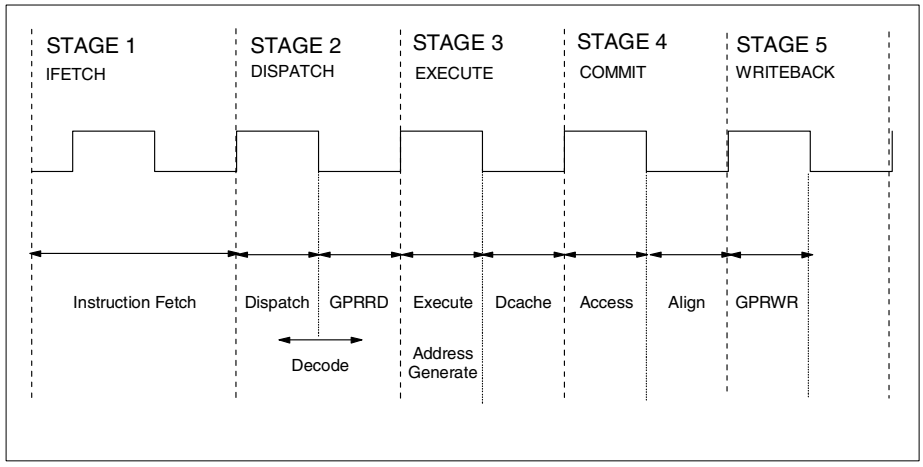


Figure 21. RS64 III's Pipeline

- Instruction fetch stage:

  In the instruction fetch stage, the L1 instruction cache array is accessed with the address generated by the branch unit and 32 bytes of instructions are output and written into either the 16-entry sequential instruction buffer or the 8-entry branch buffer. For enhanced performance, RS64 III's L1 instruction cache was doubled in size to 128 KB and made two-way set associative. The sequential instruction buffer and the branch buffer are used for conditional branch processing, explained the Section “Branches and instruction cache address generation” on page 65.

- Dispatch stage:

  The dispatch stage is responsible for decoding and dispatching up to four 4-byte PowerPC instructions each cycle. Instructions are dispatched in order from either the sequential instruction buffer or the branch buffer. During dispatch, operands are read from architected registers, completion buffers, and result busses. The instruction cache branch target address is generated in the dispatch stage unless there are dependencies on instructions that have not yet executed.

- Execute stage:

  During the execute stage, the arithmetic, rotate, and data cache address generation functions are performed. All results can be bypassed to all

execution units for use in the next cycle by subsequent instructions. If any of the input operands are invalid due to dependencies, then the execute stage for that pipeline stalls.

- Commit stage:

  The commit stage holds execution results for each pipeline. Taken branches, exceptions, and page faults can cause commit (and execute) results to be discarded. Commit stage results can be bypassed to the execution stage. Cache fetch data is bypassed to the execution stage through an aligner.

- Writeback stage:

  The writeback stage of the pipeline writes the instruction results into architected registers once all branch and exception conditions are resolved.

Branches and instruction cache address generation

When it comes to keeping the pipeline full of instructions, conditional branches pose a special problem.

Many processor designs solve this problem with branch prediction logic, but software code executed in the commercial environment has fewer code loops, making branches harder to predict with high accuracy. The RS64 Series of processors takes a different approach in solving the conditional branch problem by minimizing the branch mispredict penalty to zero or one cycle. This is accomplished with a combination of techniques.

The wide 32-byte instruction fetch path from the instruction cache paired with the 16-entry sequential instruction buffer and the 8-entry branch instruction buffer allow instructions at the branch target to be prefetched while instructions are being executed out of the sequential instruction buffer.

The first cycle of branch processing begins prior to the dispatch stage. The branch logic looks ahead, up to six instructions, into the dispatch queue to find a branch instruction. The first branch instruction found is decoded and its branch target address is generated. In the second cycle of branch processing, the instruction cache array is accessed and the aligned output is written into an instruction buffer.

By default, branches are assumed *not taken*; that is, instructions are dispatched and executed down the *not taken* or *sequential* path prior to the outcome of the branch instruction being known. This is equivalent to predicting the branches as not taken. Once the outcome of the branch instruction is known and if the branch is taken, the instructions dispatched

after the branch instruction are canceled. The *branch taken* logic switches dispatch from the sequential instruction buffer to the branch instruction buffer and cancels instructions dispatched down the *not taken* path. The *branch taken* logic is some of the most timing critical logic in the processor. The zero detect and sign bits in the execution stage of the fixed-point units are bypassed into the *branch taken* generation logic.

Branch penalty is defined as the time from dispatch of a branch instruction to the dispatch of the target of a branch instruction. When the instruction cache branch address is generated ahead of the dispatch stage and the branch condition is known at the end of the dispatch stage, there is no branch penalty. This is known as a *zero cycle* branch. When the instruction cache address is generated during dispatch stage (instead of earlier) or when an instruction modifying the condition register is dispatched in parallel with the conditional branch, a one cycle branch penalty is incurred.

Fixed- and floating-point units
Two of the four superscalar units are fixed-point units (FXUs) and have single-cycle execution for the bulk of the integer arithmetic instructions. One of the two FXUs is specialized to also execute multi-cycle integer instructions, such as multiply and divide.

Although a commercial processor, it was deemed necessary to implement a simple and efficient pipeline for floating-point arithmetic. The floating-point unit (FPU) is fully independent and contains hardware for square root and division as well as for the fused multiply-add instruction. The FPU is fully pipelined with four-cycle latency, single-cycle throughput. The load store unit includes a custom dynamic adder to allow for high speed cache address generation.

L1 data cache, L2 cache, and bus interface unit
Minimizing L1 data cache latency is key to high performance without complexity. The RS64 Series of processors are microarchitected so that the L1 data cache access has a one cycle load-to-use penalty. Innovative custom design circuit techniques were used to double RS64 III's L1 data cache to 128 KB while still maintaining a one cycle load-to-use penalty.

The L1 data cache data bypasses directly into the execution units. It is two-way set associative with a 16-byte interface to the execution units and a 32-byte interface for cache line replacement.

The L1 data cache was designed with four single-port arrays. Chip area was saved by using single-port arrays instead of multi-port arrays to increase the number of entries in the cache while minimizing cache access latency. Cache

line replacements and stores normally done with a second cache port are accomplished by queuing them in a line fill buffer and a store buffer. The fills and stores are done either during background cycles when the instruction stream is not accessing the data cache or simultaneously with instructions that operate on 8 bytes of data or less. The majority of instructions operate on 8 bytes (64 bits) of data or less, and these instructions use at most one half of the available L1 data cache interface. The line fill buffer holds seven cache lines and has the characteristics of an L0 (Level Zero) cache in that any portion of an incoming line can be stored to or read from. A high-speed bypass path around the line buffer exists for the first data transfer coming from L2 cache or main store going directly to the execution units.

The on-chip BIU contains the L2 cache directory, interface logic to support up to an 8 MB L2 cache, 6XX system bus protocols, and dedicated hardware to hide latency to memory. The L2 directory contains an entry for each 128-byte cache line held in the 8 MB L2 data cache. L2 latency is right behind L1 latency when it comes to impact on performance, so various innovative techniques were used to minimize the L2 load-to-use latency to a total of seven 450 MHz cycles. L1 data cache accesses are speculatively forwarded to the L2 and canceled if an L1 cache hit is detected. The L2 SRAM clocking logic on RS64 III tolerates a wide range in access delays caused by SRAM process variation without adding latency to the access path. RS64 III's L2 cache is implemented with double data rate SRAMs external to the processor chip. The external L2 cache is 4-way set associative and directory information for accessed in parallel. Associativity in the L2 results in higher L2 cache hit rates for most commercial workloads.

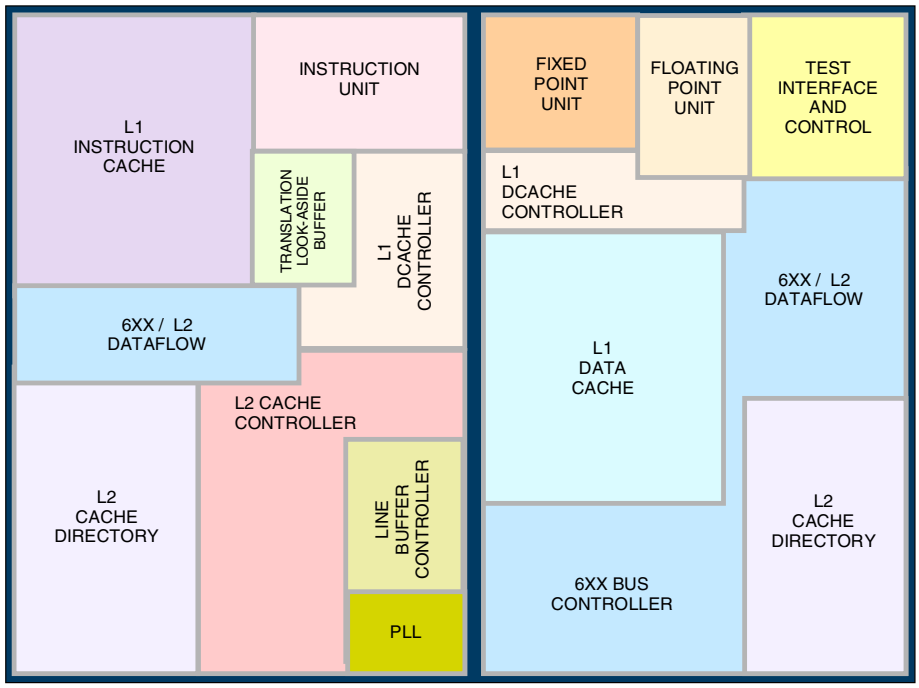


Figure 22. RS64 III Processor Chip

Containing 34 million transistors, the RS64 III processor die is shown in Figure 22. It is manufactured in IBM's 0.22 micron copper CMOS 7S technology, with six levels of copper interconnect.

Instructions that gate SMP performance, such as those related to locks, TLB (Translation Lookaside Buffer), cache management, and synchronizing, are optimized for performance in the storage control microarchitecture. Lock information is bypassed between pipeline stages to prevent pipeline stalls. The TLB table walk routine is implemented in circuits instead of microinstructions to reduce table walk time. The cache coherency scheme implemented by the RS64 Series processors does not require synchronizing instructions to be broadcast on the system bus, minimizing the performance impact due to synchronization.

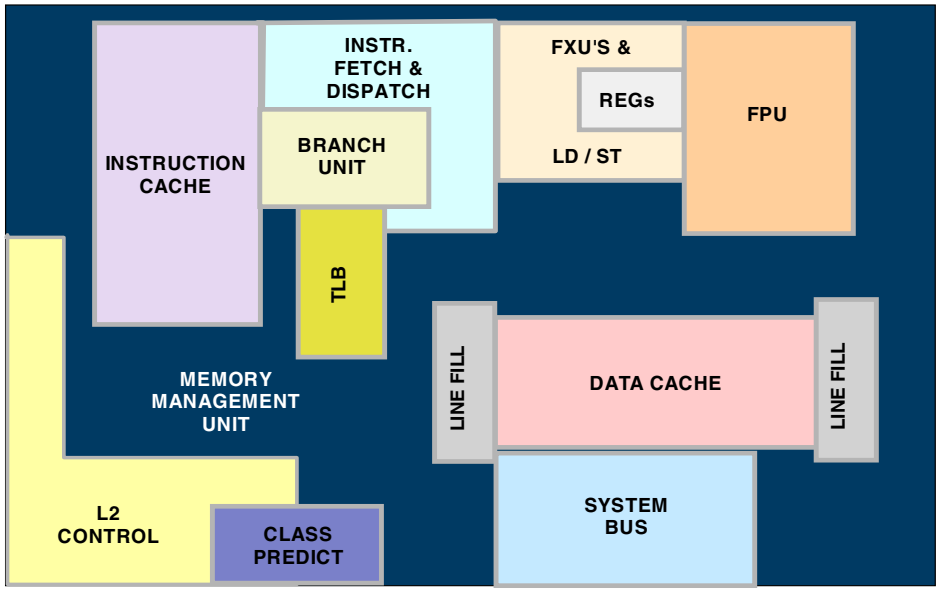


Figure 23. RS64 II Processor Chip

Containing 12,500,000 transistors, the RS64 II processor die is shown in Figure 23. It is manufactured in IBM's 0.35 micron hybrid CMOS 6S2 technology, with five levels of interconnect metallurgy.

3.3.4.3 System implementation

A key challenge of the RS64 III processor was to design a high-bandwidth system interface required to support the high miss rates driven by commercial processing. RS64 III leveraged IBM's advanced packaging technology to implement separate, independent 16-byte memory bus and 32-byte L2 bus, each with separate address, data, and control lines, achieving 14.4 GB/s to the L2 at 450 MHz. This was achieved with a total of 2030 chip I/Os of which 985 are signal I/Os.

The system interface is designed to allow flexibility in system implementation from low cost, bus-based systems to more complex switch-based configurations, providing greater address and data bandwidth. The 6XX memory bus architecture implemented in RS64 Series processors provides the scalability required to support large SMP systems.

The RS64 III processor design supports Modified Exclusive Shared Invalid (MESI) snoop-oriented SMP cache coherence along with remote processor bus protocols for increased throughput and large system topologies.

One characteristic of transaction processing is a high rate of data sharing between processors. The RS64 Series of processors provides improved performance in this environment by allowing cache lines to be transferred directly between processors with a technique called intervention. This results in shorter cache miss latencies compared to retrieving all L2 cache miss data from main store.

Error correction, detection, and isolation

The commercial processing environment requires high data integrity and high availability. On-chip arrays comprise the largest portion of chip area, and are also the most susceptible to failures. For this reason, RS64 Series processors have built in recovery for single-bit array failures. If an error is detected in the instruction cache, instruction cache directory, data cache directory, or the TLB, the entry in error is invalidated or marked unusable and its correct contents refetched. The L1 data cache directory was duplicated to provide adequate bandwidth required to support the processor pipelines and SMP snooping. These separate L1 data directories are exact copies of each other, resulting in built-in redundancy that is used to recover from errors in either directory.

The L1 data cache policy is store-in and may hold the only copy of modified data in the system. For this reason, the L1 data cache is implemented with an ECC scheme that can detect double-bit errors and correct single-bit errors. The off-chip L2 data cache and the L2 Directory are also covered by ECC.

Various parity schemes are integrated into the control and data flow logic on the processor chip. Whenever a recoverable or non-recoverable error is detected, information pertaining to the error is recorded by the hardware and made available to the system diagnostics to isolate the failing circuits.

3.3.4.4 RS64 roadmap for the future

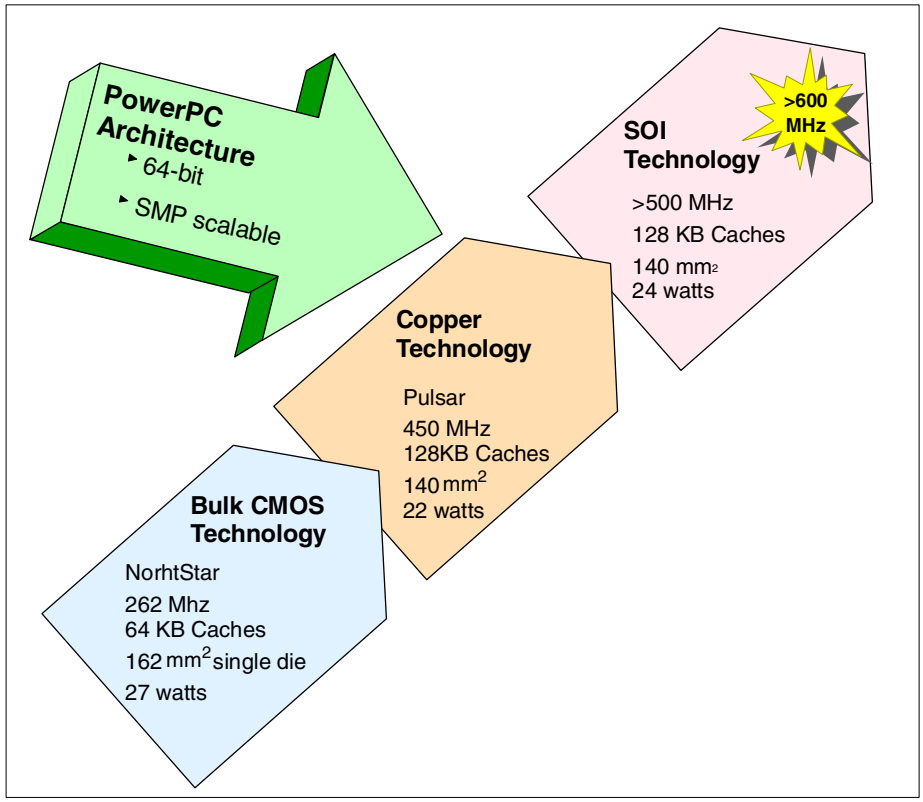


Figure 24. RS64 Series

Technology maps and performance tuning are planned to rapidly scale the RS64 Series family of processors in excess of 600 MHz in order to sustain top performance benchmarks in the commercial environment. Additional functionality for commercial and server applications is also planned to be added to the future microprocessors.

The design point after RS64 III implements IBM's industry-leading CMOS 7S Silicon On Insulate (SOI) technology that provides performance gains associated with shrinking channel lengths to 0.12 micron and a reduction in capacitance.

3.3.4.5 Summary

In summary, the *RS64 Series* processors are very robust, delivering real performance on real applications for the next generation of 64-bit RISC

commercial server processors while retaining optimum chip size and power. They achieve high performance on real applications because of the low latency design and IBM's superior silicon technology. These processors be expected to lead the commercial server benchmarks for years to come.

3.3.4.6 Reference

Additional information may be obtained from:

`http://www.rs6000.ibm.com/resource/technology/pulsar.html`

3.3.5 POWER4

The POWER4 processor as shown in Figure 25 on page 73 was designed to operate at speeds of over 1 GHz and can handle commercial and technical workloads. The server workload characteristics of Commercial and Technical computing are:

- Commercial
  - Large database footprints
  - Small record access
  - Random access patterns
  - Sharing / Thread communication
- Technical
  - Structured data
  - Large data movement
  - Predictable strides
  - Minimal data reuse

E-business applications include attributes from both commercial and technical workloads.

Binary compatibility with 64-bit PowerPC architecture is maintained.

3.3.5.1 Characteristics

The following are characteristics of the POWER4 processor:

- Process.
  - A 0.18 micron lithography is used.
  - Copper silicon-on-insulator.
  - 170 million transistors.
- Package.
  - The package is a Multi Chip Module (MCM) that allows for dense integration.

  - Uses a large number of I/Os at chip and MCM level (MCM package has 5,500 pins with 2,200 I/O).
- High bandwidth for fast busses.
  - Elastic I/O provides greater than 500 MHz chip to chip busses.
  - Chip bandwidth of greater than 1Tb/s.
- Each module contains two processes.
- The MCM package can hold four modules (8 processors).
- An expansion bus is included for scalability using SMP, NUMA, and SP Cluster technologies.

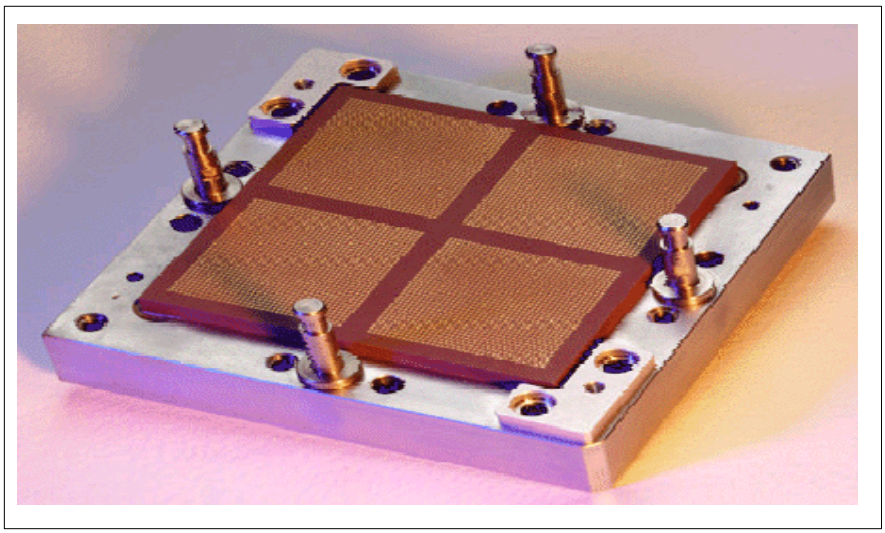


Figure 25. The POWER4 Processor

The building block of the processor is shown in Figure 26 on page 74. It has two 64-bit 1 GHz five-issue superscalar cores, a triple level cache hierarchy, a greater than 10 GBytes main memory interface and expansion bus, and a greater than 45 GBytes multiprocessor interface.

Each module has two processors (also known as *cores*) as shown in Figure 26 on page 74. Four processor chips will be offered in a single MCM package.

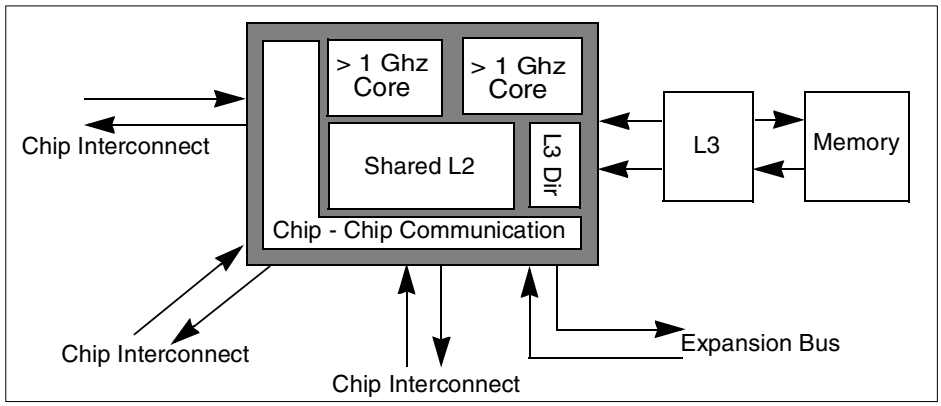


Figure 26. Server Building Block of POWER4 Processor

All processors will be able to communicate with each other as shown in Figure 27.

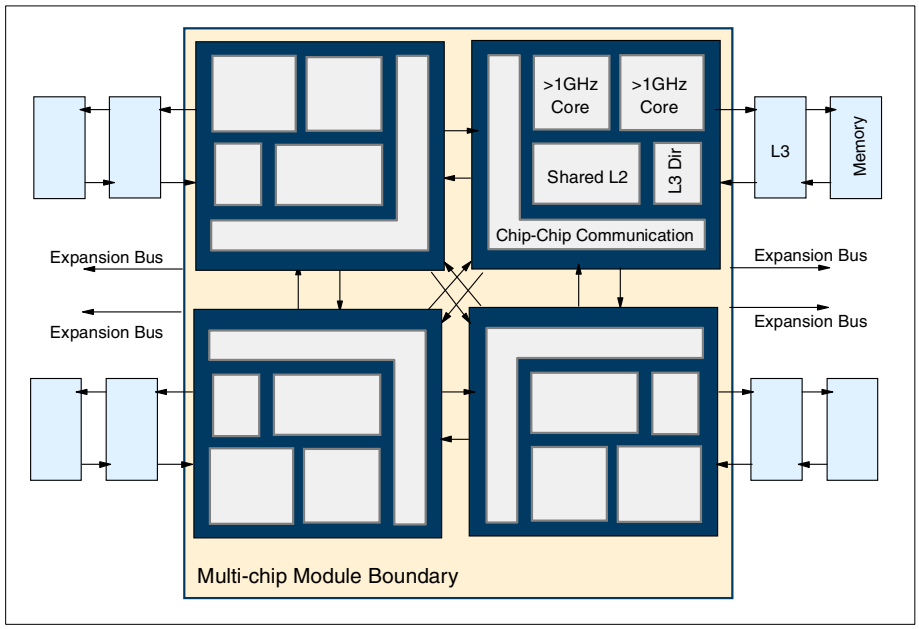


Figure 27. Server Multi-Chip Module

3.3.5.2 Reference

Additional information may be obtained from the following web site:

`http://www.chips.ibm.com/news/1999/microprocessor99.pdf`

Chapter 4. IBM RS/6000 and IBM pSeries products

This chapter provides information on IBM Symmetrical Multi-Processor (SMP) and IBM Scalable Processor (SP) products and their performance-related features.

The chapter discusses the concepts and design considerations of SMP, when SMP is the correct upgrade path for you, and the considerations when migrating from an Uni-Processor environment to a Multi-Processor environment. An SMP versus SP selection guide is also provided.

The SP section discusses the parallel architecture of the IBM SP system, the SP Switch performance, and Parallel System Support Programs (PSSP), as well as giving guidelines on how to size and configure an SP System and the associated Control Workstation.

4.1 Symmetrical Multiprocessor (SMP)

Symmetrical Multi Processing (SMP) is technique that employs more than one CPU to increase CPU throughput.

SMP RS/6000 servers are designed to spread a workload across multiple processors within a system, improving the overall performance. The RS/6000 SMP was initially based on PowerPC technology employing the 601 microprocessor. SMP provides an upgrade path for performance improvements for the installed base as well as giving scalability.

An SMP system uses a different kernel than uniprocessor systems. The multi-processor kernel is contained in the fileset *bos.mp*, and the uni-processor kernel is contained in the fileset *bos.up*

4.1.1 Migrating to SMP

Many installed systems are uni-processor, so migrating to SMP to increase CPU power is an option to enhance performance.

Performance analysis of the system to be upgraded must be completed to make sure that the addition of extra CPUs will solve any performance issues, and the software you intend running on the system will support multiple threads so it will run successfully in the SMP environment.

If improved speed is the objective of the migration from uniprocessor to multiprocessor, the following should all be true:

- The existing processor has reached CPU saturation. This means the CPU is unable to provide any more throughput and the system is constrained by lack of CPU.
- The workload contains multiple processor-intensive elements, such as transactions or complex calculations that can be performed simultaneously and independently.
- The existing processor cannot be upgraded or replaced with another uniprocessor with adequate power.
- One or more considerations, such as a centralized database, preclude dividing the workload among multiple uniprocessor systems.
- The proposed software is multi threaded.
- Your existing hardware will support additional CPUs.

It is important to observe these recommendations. If your applications do not support an SMP environment, then the performance of the system could be worse than on a uniprocessor system.

Changing single thread applications to an SMP environment can have unpredictable results on performance. Migration to a multiprocessor environment can improve throughput of a system and the execution time of complex, multi thread applications, but single threaded applications will seldom see any benefits from a multiprocessor environment.

4.1.2 Symmetrical Multiprocessor (SMP) concepts and architecture

As with any change that increases the complexity of the system, the use of multiple processors generates design considerations that must be addressed for satisfactory operation and performance. The additional complexity gives more scope for hardware and/or software trade-offs, and requires closer hardware and/or software design coordination than in uniprocessor systems. The different combinations of design responses and trade-offs produces a wide variety of multiprocessor system architectures.

This section describes the main design considerations of multiprocessor systems, and the responses of AIX and the RS/6000 to those considerations.

Perhaps the most fundamental decision in designing a multiprocessor system is whether the system will be symmetrical or asymmetrical. AIX only supports symmetrical systems.

The major design considerations in an AIX SMP environment are:

- Data serialization
- Lock granularity
- Locking overhead
- Cache coherency
- Processor affinity
- Memory and Bus contention

In a symmetrical multiprocessor system, all of the processors are essentially identical and perform identical functions:

- All of the processors work with the same virtual and real address spaces.
- Any processor is capable of running any thread in the system.
- Any processor can handle any external interrupt (each processor handles the internal interrupts generated by the instruction stream it is executing).
- Any processor can initiate an I/O operation.

This interchangeability means that all of the processors are potentially available to handle whatever needs to be done next. The cost of this flexibility is primarily borne by the hardware and software designers, although symmetry also makes the limits on the multi process ability of the workload more noticeable, as we shall see.

4.1.2.1 Funneling

A single processor is initially in control during the boot process. This first processor to be started is designated as the *master processor*. To ensure that user-written software continues to run correctly during the transition from uniprocessor to multiprocessor, device drivers and kernel extensions that do not explicitly describe themselves as able to run safely on multiple processors are forced to run only on the master processor. This constraint is called *funneling*.

4.1.2.2 Data serialization

Any storage element that can be read or written by more than one thread may change while the program is running. This is generally true of multiprogramming environments as well as multiprocessing environments, but the advent of multiprocessors adds to the scope and importance of this consideration in two ways:

- Multiprocessors and thread support make it attractive and easier to write applications that share data among threads.

- The kernel can no longer solve the serialization problem simply by disabling interrupts.

To avoid data integrity issues, programs that share data must arrange to access that data serially, rather than in parallel. Before a program touches a shared data item, it must ensure that no other program (including another copy of itself running on another thread) will change the item.

The primary mechanism that is used to keep programs from interfering with one another is the *lock*. A lock is an abstraction that represents permission to access one or more data items. Lock and unlock requests are *atomic*; that is, they are implemented in such a way that neither interrupts nor multiprocessor access affect the outcome. All programs that access a shared data item must obtain the lock that corresponds to that data item before manipulating it. If the lock is already held by another program (or another thread running the same program), the requesting program must defer its access until the lock becomes available.

Besides the time spent waiting for the lock, serialization adds to the number of times a thread becomes nondispatchable. While the thread is nondispatchable, other threads are probably causing the nondispatchable thread's cache lines to be replaced, which will result in increased memory-latency costs when the thread finally gets the lock and is dispatched.

The AIX kernel contains many shared data items, so it must perform serialization internally. This means that serialization delays can occur even in an application program that does not share data with other programs, because the kernel services used by the program have to serialize on shared kernel data.

4.1.2.3 Lock granularity

A programmer working in a multiprocessor environment must decide how many separate locks should be created for shared data. If there is a single lock to serialize the entire set of shared data items, lock contention is comparatively likely. If each distinct data item has its own lock, the probability of two threads contending for that lock is comparatively low. Each additional lock and unlock call costs processor time, and the existence of multiple locks makes a deadlock possible. At its simplest, deadlock is a situation where, for example, Thread 1 owns Lock A and is waiting for Lock B, while Thread 2 owns Lock B and is waiting for Lock A. Neither program will ever reach the unlock call that would break the deadlock. The usual preventive for deadlock is to establish a protocol by which all of the programs that use a given set of locks must always acquire them in exactly the same sequence.

4.1.2.4 Locking overhead

Requesting locks, waiting for locks, and releasing locks add processing overhead in several ways:

- A program that supports multiprocessing always does the same lock and unlock processing, even though it is running in a uniprocessor or is the only user in a multiprocessor system of the locks in question.
- When one thread requests a lock held by another thread, the requesting thread may spin for a while or be put to sleep and, if possible, another thread may be dispatched. This consumes processor time.
- The existence of widely used locks places an upper bound on the throughput of the system. For example, if a given program spends 20 percent of its execution time holding a mutual-exclusion lock, at most five instances of that program can run simultaneously, regardless of the number of processors in the system. In fact, even five instances would probably never be so nicely synchronized as to avoid waiting on one another.

4.1.2.5 Cache coherency

In designing a multiprocessor, engineers give considerable attention to ensuring cache coherency. They succeed but their success is not free. To understand why cache coherency has a performance cost, we need to understand the problem:

If each processor has a cache, which reflects the state of various parts of memory, it is possible that two or more caches may have copies of the same line. It is also possible that a given line may contain more than one lockable data item. If two threads make appropriately serialized changes to those data items, the result could be that both caches end up with different, incorrect versions of the line of memory; that is, the systems state is no longer coherent because it contains two different versions of what is supposed to be the content of a specific area of memory.

The solutions to the cache coherency problem usually include invalidating all but one of the duplicate lines. Although the invalidation is done by the hardware, any processor whose cache line has been invalidated will have a cache miss, with its attendant delay, the next time that line is addressed.

Figure 28 on page 82 shows an example of the problem. Suppose process p1 is running on processor 1 and process p2 is running on processor 2. Suppose also that both processes p1 and p2 are working together on a problem sharing some memory. Consider the following sequence of events:

Process p1 loads address *123*, which contains character *A*.

Process p2 stores the character *B* into address 1*23*.

Process p1 loads address *123* again.

The value seen by process p1 at step 3 is very important. With a naive implementation, p1 sees *A* because it has a copy of address *123* in its cache, the store request of p2 never goes out to memory, and p1 does not see the new value *B* that process p2 placed there.

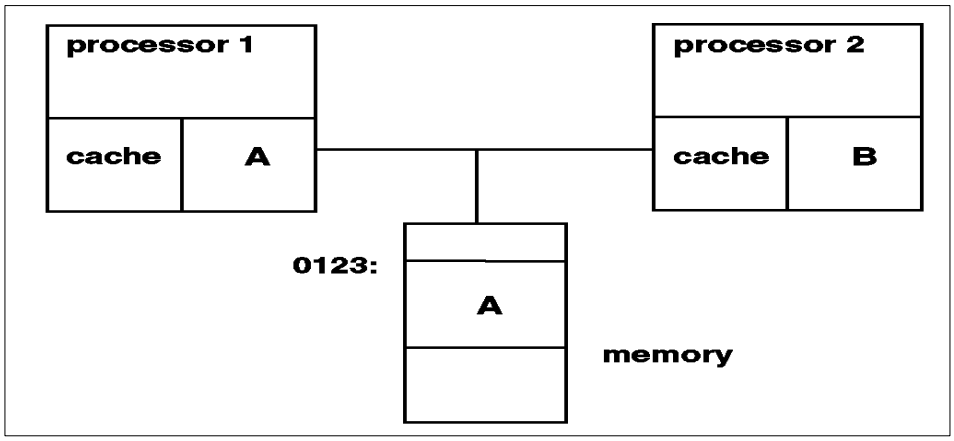


Figure 28. Cache Coherency Problem

Snooping

One solution to the cache coherency problem is snooping. Snooping is a hardware logic at each processor/bus interface that broadcasts a message over the bus each time a word in its cache is changed. The logic also snoops on the bus for such messages from other processors. Whenever it detects that another processor has changed the value at an address that is copied in its own cache, the snooping logic invalidates that entry in its cache. This *cross invalidate* reminds the processor that the value in that location in the cache is invalid. If the processor needs to access this data, it will have to look for the value in another cache or in the main memory. Cache-to-cache data transfers are called *intervention*.

In the example, when process p1 reads address 123 the second time, it gets a cache miss and must look elsewhere for the value. The extra snooping logic determines where process p1 should look to get the proper value for address 123. If the new value has not yet been written to memory, process p1 will obtain the value from the cache of processor 2.

Because cross invalidates increase cache misses and the snooping protocol adds traffic to the bus, solving the cache consistency problem reduces the performance and scalability of all SMPs. Therefore, IBM SMPs employ a specific technology, called the data crossbar switch, to alleviate the impact of cache consistency on performance.

4.1.2.6 Processor affinity

If a thread is interrupted and later redispatched to the same processor, there may still be lines in that processor's cache that belong to the thread. If the thread is dispatched to a different processor, it will probably experience a series of cache misses until its cache working set has been retrieved from main memory. On the other hand, if a dispatchable thread has to wait until the processor it was previously running on is available, the thread may experience an even longer delay.

Processor affinity is the dispatching of a thread to the processor that was previously executing it. The degree of emphasis on processor affinity should vary directly with the size of the thread's cache working set and inversely with the length of time since it was last dispatched.

Provided your application is multi-threaded, AIX automatically tries to encourage processor affinity, but in AIX Version 4 processor affinity can be forced by binding a thread to a processor with the *bindprocessor* command. A thread that is bound to a processor can run only on that processor, regardless of the status of the other processors in the system. Care must be taken when binding threads to a processor, as you may be denying affinity to other threads and therefore causing a potential degradation of performance.

4.1.2.7 Memory and Bus contention

In a uniprocessor, contention for some internal resources, such as banks of memory and I/O or memory buses, is usually a minor component of processing time. In a multiprocessor, these effects can become more significant, particularly if cache-coherency algorithms add to the number of accesses to main memory.

False sharing

The unit of access in the cache is called a *line*. A typical cache line on the RS/6000 machines is either 32 bytes or 128 bytes. It is possible for two processes to reference two different portions of data that fall in the same cache line because they lie close to each other in memory. For example, if a process on processor 1 changes the value of d1, the cache consistency logic will invalidate processor 2’s cache line, causing a cache miss when d2 is accessed, even though the two processes were not sharing any data. This is

called *false sharing*. False sharing increases cache misses and bus traffic, further reducing SMP throughput and scaling.

Memory subsystem performance
One of the techniques used to improve memory latency is to interleave the memory. This technique is not specific to the IBM SMP but is generally used by the industry. However, the IBM SMP implements a very high level of 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 *0x0FFFFFFF*; the second would store *0x1* 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, which 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. In the example of a 32 bytes cache sector, the data for physical addresses *0x0* through *0x1F* would be stored in the first module, addresses *0x20* to *0x3F* in the second, addresses *0x40* to *0x5F* in the third and addresses *0x60* to *0x7F* in the fourth. Then, the addresses would wrap around to the first module again for addresses *0x80* to *0x9F*, and so on. The exact way in which the physical address space is interleaved among the modules is invisible to the software.

When interleaving is done with four modules, we say it is a 4-way interleaving. IBM SMP systems have a very high level of interleaving. The level of interleaving depends on the machine type and memory configuration.

In summary, memory interleaving is a technique developed to allow simultaneous access to adjacent areas of memory. Interleaving also provides the ability to minimize *bank busy* effects in highly contended memories.

4.1.2.8 Memory switch
A high number of cache misses is characteristic of commercial applications. Snooping activity, high intervention rates, and transfers between memory and the I/O subsystem cause a high level of activity on the bus. If an SMP

machine uses processors that are tied together using a shared bus, the memory bandwidth can become a potential bottleneck.

Large IBM SMP systems are therefore designed in a different way. There is still a mechanism for the snooping activity and the addressing, but another component has been added for data transfers. That component is a switch. 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.

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 29 illustrates the use of the switch for data transfers.

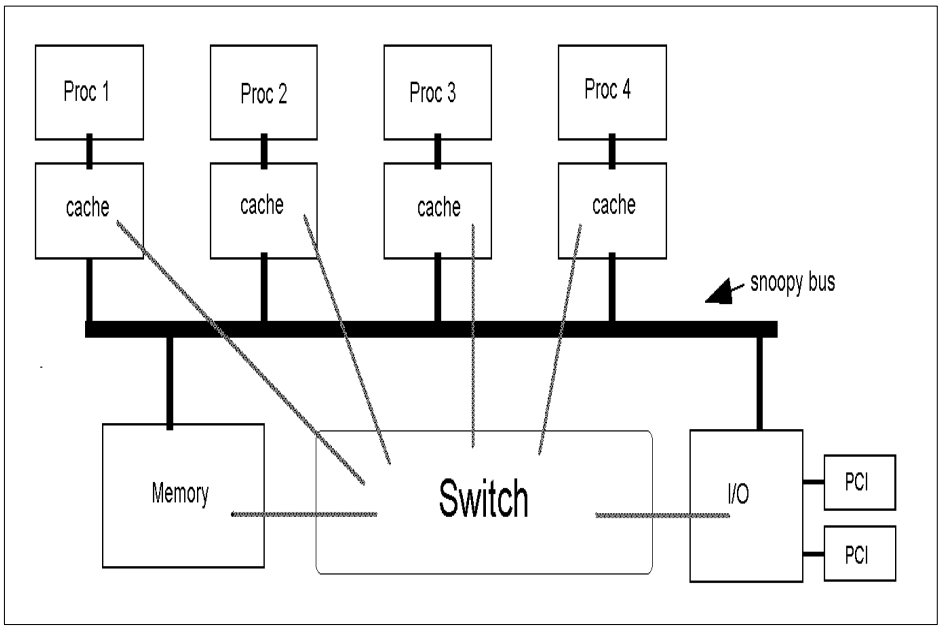


Figure 29. Using a switch for data transfers

Memory switch performance
Memory switches do not eliminate data buses; they duplicate them to deliver more bandwidth. Every port on the switch is constrained to be point-to-point so that data transfers can operate at the bus speed.

In the IBM RS/6000 S80 and the IBM @server pSeries 680, the backplane uses a switch based memory controller complex. The complex contains 10 chips and additional high-function address and data buffers. There are actually two sets of switch chips. Each independent set of chips work on odd or even cache line accesses.

In each set of chips, one chip acts as the data flow control chip for four data switch chips. There is an additional system bus arbiter chip. Cross-port traffic is queued at the switch if needed. Each of the processor card dual system buses is directly connected to a port on each of the sets of modules in the switch-based memory controller complex. Each data path is 128 bits wide. Addressing is via a separate 64-bit data path.

There are additional high-function address and data buffer chips that break these logical buses into smaller physical buses. This allows the frequency of the buses and data rates to be increased to the 150 MHz level. These high-function buffers allow each of the system buses to support up to 2.4 GB/s of throughput.

4.1.3 Software

Because 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. To handle this activity correctly, changes must be made to the UP operating systems to be used with the SMPs.

4.1.3.1 Threads and locks

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. Because a classical UNIX process is considered to be a single-threaded process, threads will be used in this section to illustrate some concepts.

SMP synchronization issue
There is a potential synchronization problem when two processors try to update the same piece of data at the same time, and incorrect results can be generated.

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

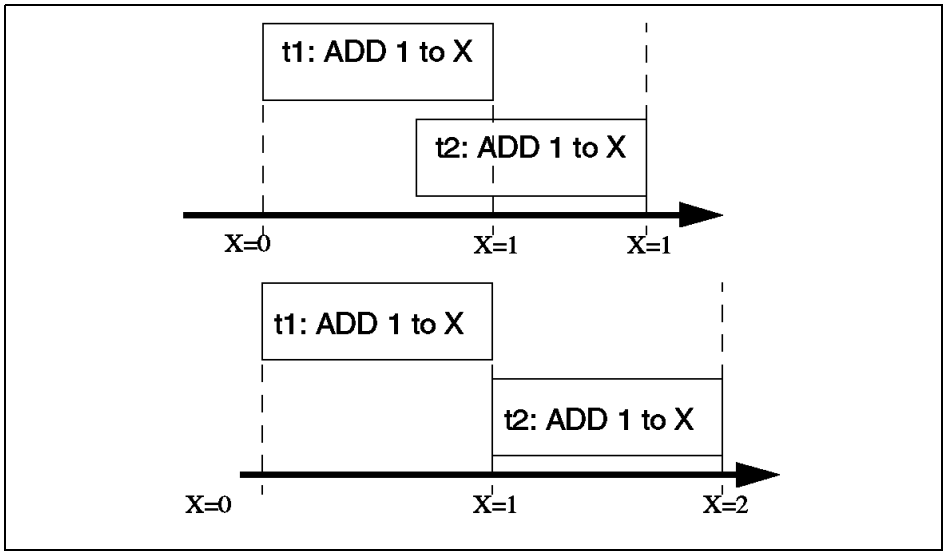


Figure 30. Synchronization Issue

In the top half of the Figure 30, threads t1 and t2 are both adding one to the same shared variable, X, whose value is n. The final value must be n+2, but t2 is incrementing the variable before t1 has finished. Therefore, the final value will be n+1. In order to avoid this, both threads have to be serialized, as shown in the bottom half of the diagram.

A critical section is a section of code that modifies shared data. Therefore, it must not be executed by more than one thread at a time. The other thread(s) must wait. 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.

Conceptually, a lock is just a bit in memory that threads use to regulate their entry into critical sections. But locks are not that simple to implement because two or more threads could test the same lock simultaneously, determine that the lock is available, and enter the critical section. Because establishing a lock requires several operations (read, test, and set the lock bit), this operation is itself a critical section. Thus, multiprocessor hardware must provide a way to perform this test and set operation atomically with respect to the other processors. This means that if more than one processor

is trying to obtain the same lock simultaneously, exactly one of them will succeed.

The two major types of locks are:

- **Mutually exclusive (simple) lock**

  It allows one process or thread at a time in a critical section.

- **Read/write (complex) lock**

  It allows multiple readers into the critical section at once but guarantees mutual exclusions for writers.

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. There are two ways of waiting; spinning and 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 is then 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, will slow down performance.

Lock penalty

Suppose that we know from tprof that when running a certain application, the system spends 10 percent of its time in a kernel component. Let us assume that the component is complex and touches a lot of data. 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 4-way SMP, this mutex lock will be busy 4 x 10 percent = 40 percent of the time.

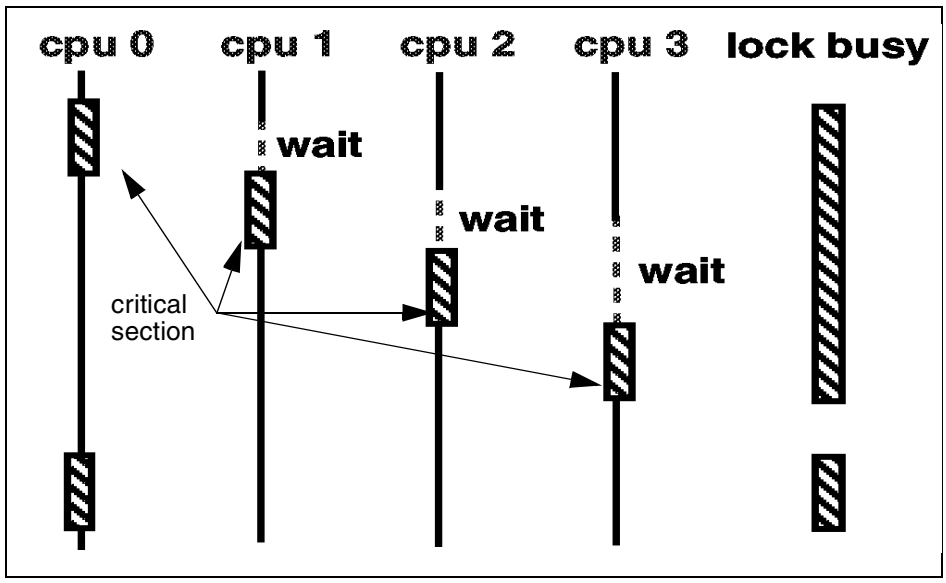


Figure 31. Lock Penalty

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

The thundering herd problem occurs when several threads are queued waiting for a resource, the resource is freed, and then several waiting threads are awakened at the same time. For simple locks, this problem is avoided by selectively waking only the highest priority sleeping thread. For complex R/W locks, either the highest priority writer is awakened, or all the readers are awakened if no writer is waiting.

Lock granularity

The amount of time a lock is busy is a function of how often it is requested and how long it is held once acquired. 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 32 on page 90 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.

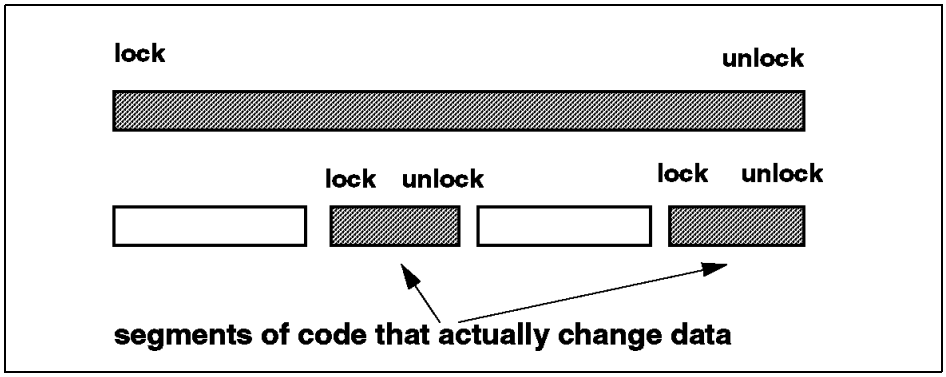


Figure 32. Lock Granularity

But, with granularity too fine, the frequency of lock requests and lock releases will increase. This adds additional instructions. Tests show that a lock/unlock pair costs approximately 20 to 125 instructions (added to the path length of the program). Therefore, if lock granularity is too fine, too many instructions are used to request and release the locks. If the lock granularity is too coarse, too much time is spent waiting for the locks. A balance must be found between a too-fine and a too-coarse granularity. This is, again, the developer's responsibility.

Lock performance considerations

Here are a few performance tips for locks:

- Never perform synchronous I/O or any blocking activity while holding a lock.
- Move all the unnecessary instructions (those not directly related to reading or modifying the protected data) outside the critical section.
- If more than one access is needed to the same data in a given component, try to move the accesses together so they can be covered by one lock/unlock pair (provided there are not too many instructions).
- If more than one lock are to be held simultaneously, request the busiest one last, if possible.
- If the protected data is mostly read, consider using a complex lock instead of a simple one.
- 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.

4.1.3.2 Processor affinity

In AIX V4, the schedulable entity is the thread, and the thread with the highest priority is the one that gets dispatched. This means that a thread is bounced from one processor to another over its lifetime. 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 where 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 has been changed to enforce affinity with the processors.

As shown in Figure 33 on page 92, run queues are ordered according to their priority, with 127 being the lowest and 0 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.

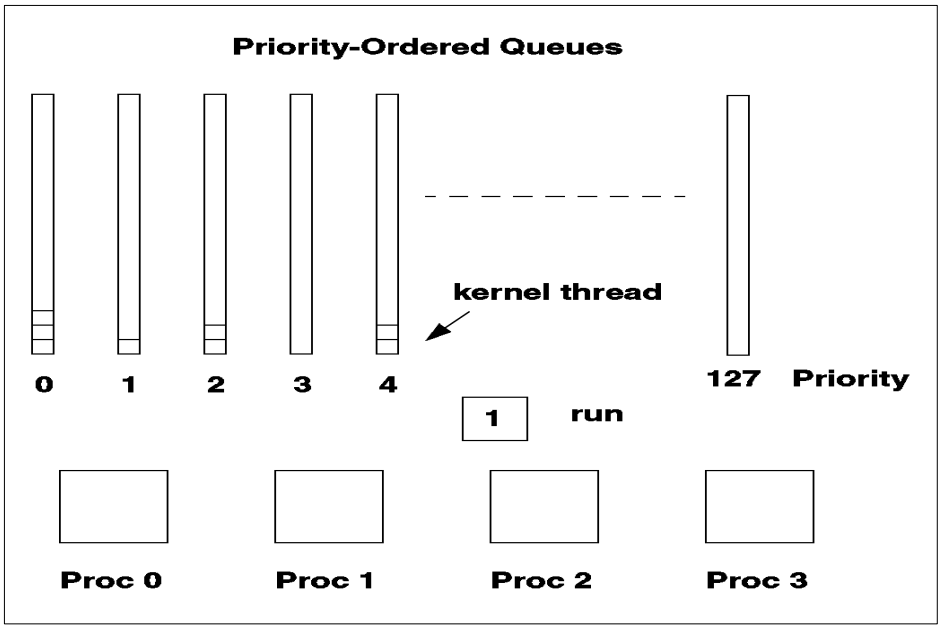


Figure 33. Threads Dispatching

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 by scanning the queues until it finds one. This scanning is not done indefinitely; the limits are:

- 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.
- 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 4-way SMP).
- Scanning is also stopped when the dispatcher encounters a boosted thread and the parameter affinity_skipboosted is FALSE.

- If a thread with a low priority holds a lock and a higher priority thread is waiting for the same lock, the low priority thread gets the priority of the higher-priority thread (it is boosted so that the higher priority thread doesn't have to wait too long for the lock). This priority inversion is always done by the system. If the affinity_skipboosted parameter is set to TRUE, the boosted thread is skipped and the dispatcher goes to find a thread that has affinity with the processor. To avoid running a lower-priority thread instead of a boosted thread, the default value of the parameter is FALSE.

4.1.3.3 Binding

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

The `bindprocessor` command allows a user to bind all threads of 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 another processor. Binding might be useful for a process that seldom blocks for long periods and whose response time is important. However, binding a process may cause some performance problems by letting some of the processors remain idle. Binding is appropriate only in special circumstances, such as on a system that is dedicated to a single application.

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

4.1.4 Scaling

One of the most important metrics of MP performance is scaling. When a processor is added to the system, how much additional performance is obtained on a given workload? Scaling is workload dependent. Some workloads will scale better than others. In addition, workloads will scale differently on different types of MPs. For example, a workload that shares a lot of data is likely to scale better on an SMP than on a shared nothing MP. This is because all processes on an SMP have a consistent view of the data, and processes on a shared nothing cluster must do message passing to share data when using a high degree of parallelism.

4.1.4.1 Scaling myth

In a perfect world, one would expect SMP performance to increase linearly as processors are added. But, as we have seen, this does not happen due to the overhead required to maintain a consistent view of the memory and other shared resources for each of the processors.

In an SMP configuration, programs share the operating system, memory subsystem, and disk I/O. Sharing means conflict, and that limits the number of processors that can be effective in a system.

The notion that SMP performance scales linearly is wishful thinking. Even the applications and benchmarks that show near linear performance must share the operating system and resources. Only those benchmarks or applications that spend very little time in the operating system, are cache resident, perform little I/O, have little main memory activity, and are CPU intensive may exhibit near linear scaling. Real programs use main memory, disk I/O and operating system services that would cause resource conflicts in an SMP system.

4.1.4.2 Scaling limitation factors

As more processors are added, each additional processor increases performance slightly less than the previously added processor. In fact, adding more processors ceases to boost performance after some critical number, as Figure 34 on page 94 shows. In the worst case, a 16-way symmetrical multiprocessor machine may provide less performance than a 12-way.

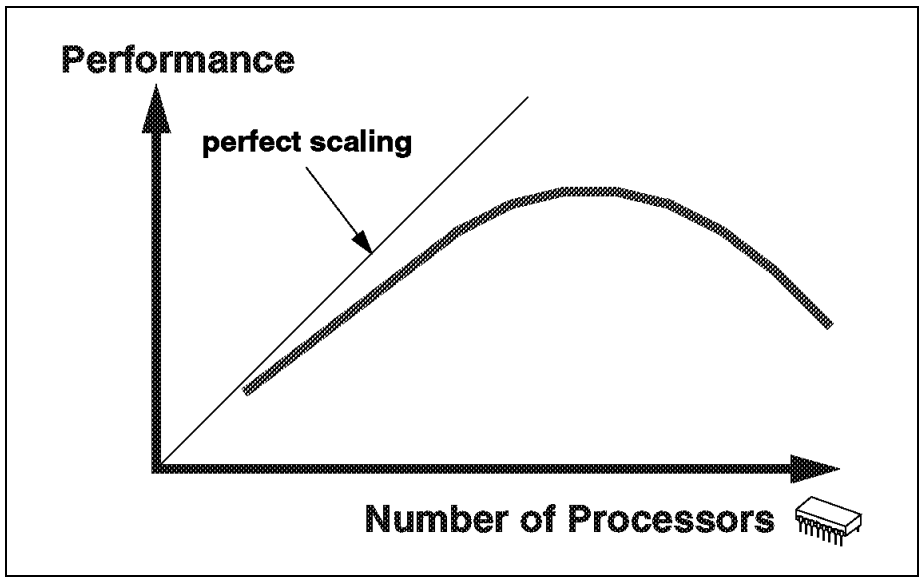


Figure 34. Scaling

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

- Increased bus/switch contention when the number of processors increases.
- Increased memory contention because all the memory is shared by all the processors. Memory conflict might occur when a processor needs to access a bank that is busy or some other memory component is locked. Obviously, a system that has many memory banks will scale better than a system that has one.
- Increased cache misses because of larger operating system and application data structures.
- Cache cross invalidates and lateral reads to maintain cache coherency. For example, if one processor requests a read of a cache line that happens to be modified and resident in another processor, the holder of the cache line will force the requestor to retry. The retry duration is not bound. Several retries may be needed to push a modified line down from a processor's L1 cache to the memory controller before being sourced to the requesting processor.
- Increased cache misses because of higher dispatching rates.
- Increased cost of synchronization instructions.
- Increased operating system and application path lengths for lock/unlock.
- Increased operating system and application path lengths waiting for locks.

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

4.1.4.3 Commercial vs. 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 processor speed is the key. Code is often made with short loops of instructions and may fit in the L1 cache.

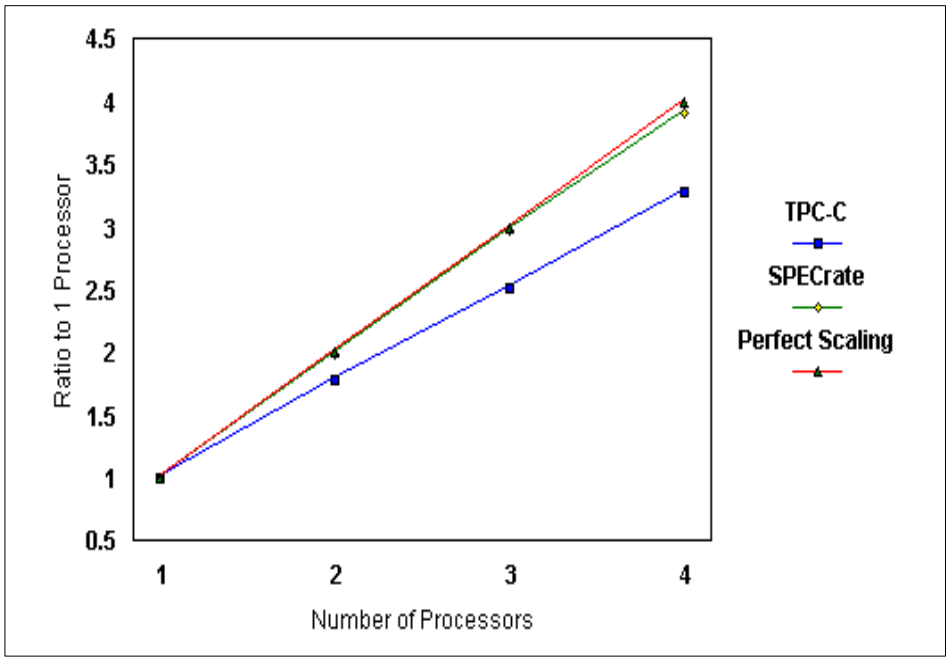


Figure 35. Scaling is workload dependent (F50)

4.1.4.4 Scaling metric

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 happens because of the MP overhead that is inherent in the kernel of the MP operating system. As a result, most vendors will show scaling starting from two processors.

Table 6 on page 97 shows how scaling can be represented. Having a ratio of 3.43 for eight processors shows that the OLTP benchmark scaling is good on the RS/6000 M80 machine.

Number of Processors	Relative OLTP Performance Value	Ratio to 2 Processors	Ratio to Number of Processors
2	61.3	N/A	N/A
4	108.7	1.77	0.44
6	160.0	2.61	0.43
8	210.0	3.43	0.43

Table 6. SMP OLTP scaling metrics for M80

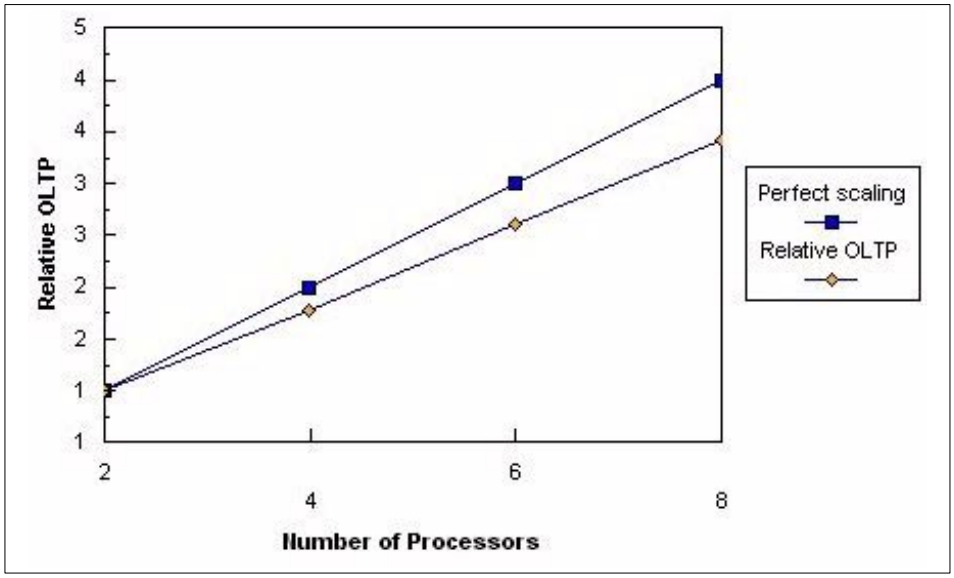


Figure 36. Graphical representation of the SMP OLTP scaling metrics for M80

4.1.4.5 Two-Dimensional scaling

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

The IBM RS/6000 SMP servers allow two-dimensional scaling by being able to utilize higher-performance processors as well as by increasing the number of processors that can be added. The memory subsystem has been designed to cater for growth. Here is an example of faster processor scaling for the M80.

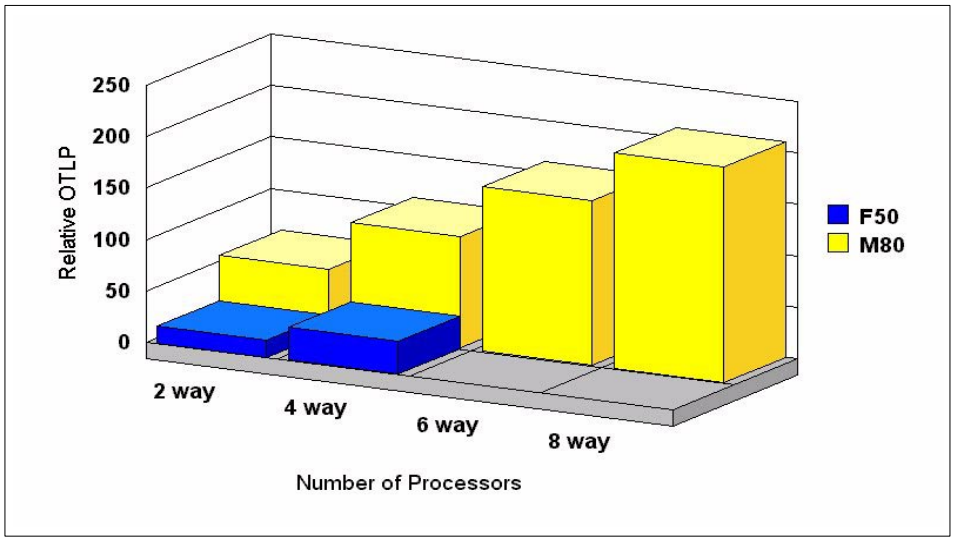


Figure 37. Two dimensional scaling between F50 and M80

4.1.5 References

Additional information can be located in:

- AIX Versions 3.2 and 4 Performance Tuning Guide

and on the web:

- `http://www.rs6000.ibm.com/resource/technology`

4.2 Scalable POWERparallel (SP)

In this section the IBM SP (Scalable POWERparallel) system is introduced, including the architecture, the SP Switch, and the shared disk components of PSSP.

Guidelines are given for sizing and configuring the Control Workstation as well as the SP System.

4.2.1 Parallel architecture

The speed of conventional computers has increased tremendously over the years, but they are still considered not fast enough to reach the level of performance required to solve some complex computations or run highly computation-intensive environments. See Figure 38 on page 99.

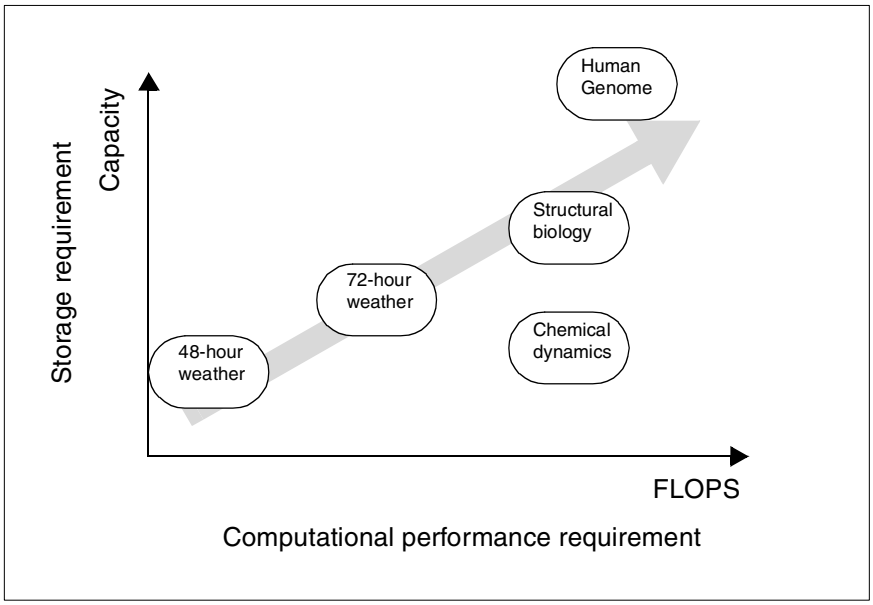


Figure 38. High computational performance requirement

As we have seen, there are several ways to increase the speed of computers. Increasing the speed further would be difficult and very expensive because of the limitations inherent in the architecture upon which conventional computers are built.

To overcome the limitation described above, the parallel architecture was introduced and implemented in several ways. You can categorize the architecture of parallel computing in terms of two aspects; whether the memory is physically centralized or distributed, and whether or not the address space is shared (see Table 7).

Table 7. Categorization of Parallel Architectures

	Shared Address Space	**Individual Address Space**
Centralized Memory	SMP	N/A
Distributed Memory	NUMA	MPP

- SMP: Symmetrical Multiprocessor
- NUMA: Non-Uniform Memory Access
- MPP: Massively Parallel Processors

4.2.2 IBM SP (Scalable POWERparallel) system

The IBM SP (Scalable POWERparallel) system belongs to the parallel architectures that are mentioned above; it is one of the MPP architectures. The MPP architecture consists of nodes that are connected by the network, which usually has a wide bandwidth. Each node has its own processor, cache, memory, and I/O subsystem. An instance of the operating system is running on each node.

There are several ways to implement such a high speed network within the SP. Either you can use well-known networks like FDDI, Gigabit Ethernet, and Ethernet, or you will use a SP unique feature - the SP Switch. The SP Switch, a state-of-the-art IBM innovation, provides a high-bandwidth, low-latency internode communication.

While the SP is designed to provide a parallel environment, it is also effective for serial workloads and for both batch and interactive applications. The system has an architecture with significant growth capabilities and is based upon proven RS/6000 technology and AIX software. It also features innovative, topology-independent switches for high-speed, inter processor communication for parallel computing, and a sophisticated set of IBM-developed software tools for system management, job management, and parallel application development and execution. Because the SP is also an open system, popular parallel interfaces from other sources are supported.

The basic SP building block is the processor node. It consists of a POWER3 or PowerPC Symmetrical multiprocessor (SMP), memory, disk, and PCI expansion slots for I/O, connectivity, and the SP Switch adapter. Node types may be mixed in a system, and are housed in short or tall system frames. Depending on the type of node selected, an SP frame can contain up to 16 nodes. These frames can be interconnected to form a system with up to 128 nodes (512 by special bid). Currently, a maximum of 16 SMP high nodes can be installed per system.

A frame is vertically divided into drawers that span the internal width of the frame, as Figure 39 on page 101 depicts. The bottom-most drawer is smaller than the remainder and will accept a switch unit. The remaining drawers, if occupied, either contain one wide node or two thin nodes. A thin node is half the width of a wide node. A high node occupies four thin-node drawers or two wide-node drawers. All node types can coexist in a single SP system with the exception that the high node is not currently available for the short frame. The number of nodes of any type can be increased incrementally to meet computing power requirements for interactive, batch, serial, and parallel jobs,

which can all be run simultaneously. Refer to the RS/6000 Configurator Program for the node restrictions.

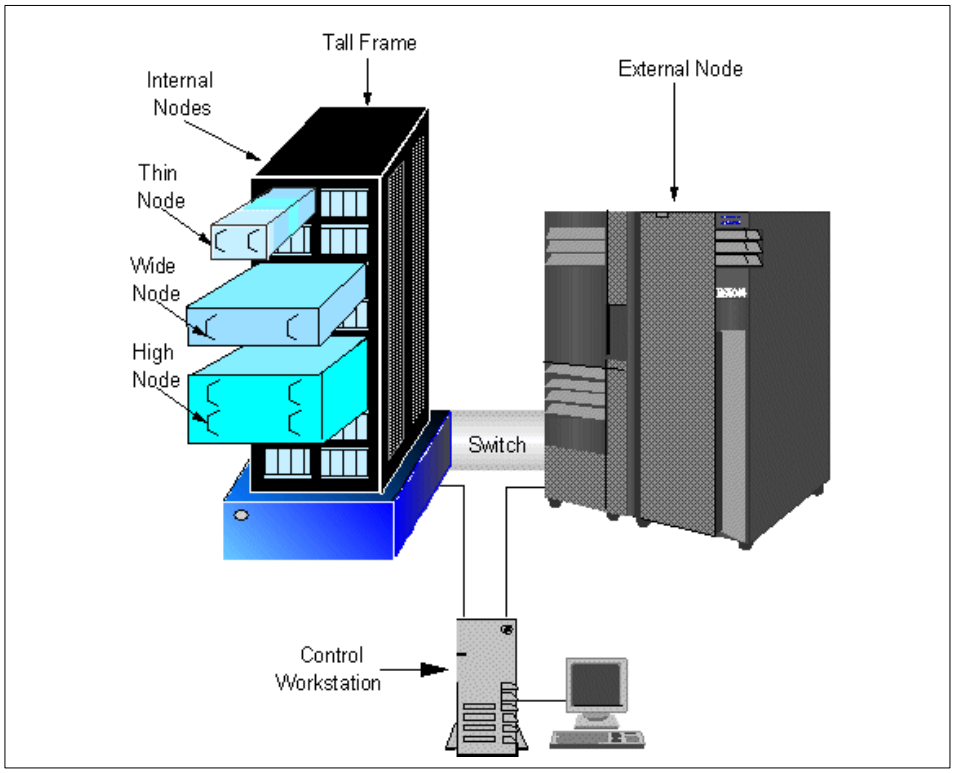


Figure 39. RS/6000 SP system sample configuration

All SP node types are equipped with 10 or 10/100 Mbps Ethernet adapters. Higher network data transfer rates may be achieved using other standard network types, such as token ring, gigabit ethernet, asynchronous transfer mode (ATM), or high performance parallel interface (HIPPI), which require adapters that occupy one or more PCI slots. Substantially higher internode communication performance, a prerequisite for true parallelism in most applications, is obtained using the IBM SP Switch. Each internal node gains access to this high-speed network through an adapter that does not occupy a PCI slot (it occupies an exclusive slot for the SP Switch MX2 adapter). External nodes like S7A or S80 use PCI based Switch adapters for the connection.

The availability of the SP Switch unit has been enhanced by improving circuit reliability, increasing redundancy, and reducing the global impact of error

detection. The SP Switch supports many communication protocols, which are IP, TCP, UDP, Standard AIX Socket, and User Space communication passing. The SP Switch is the buffered multistage packet switch. The SP Switch provides the message passing network through which SP processor nodes communicate with a minimum of four disjoint paths between any pair of nodes, and any-to-any internode connection. An important characteristic of the SP Switch is that its bi-directional bandwidth is designed to scale linearly up to 256 node connections, including intermediate switch frames, with an essentially constant latency per connection. See Figure 40 on page 103. Hence, the necessary balance between inter processor communication speed and the total system computation power is retained as the number of processor nodes is increased. This is why the SP is a truly scalable parallel system.

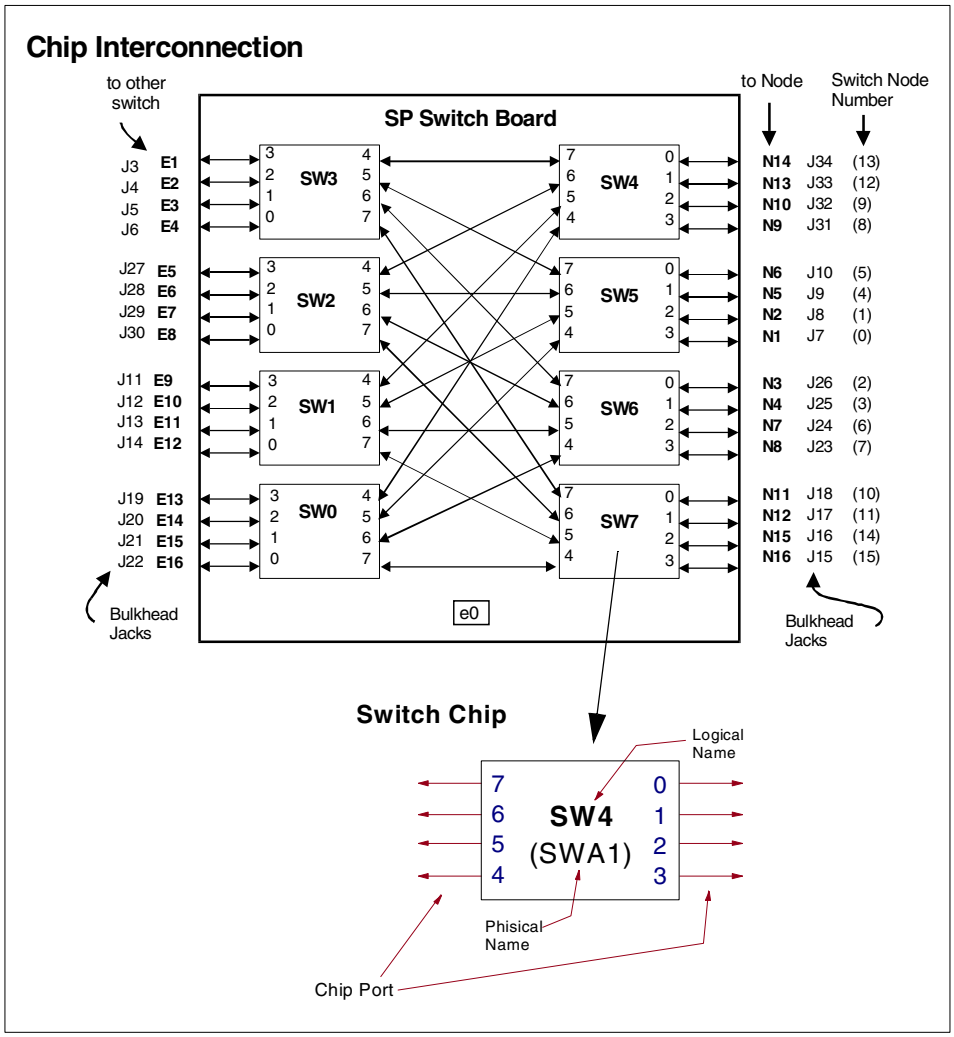


Figure 40. SP Switch chip

As you can see in Figure 41 on page 104 an RS/6000 workstation (so called control workstation = CWS) is needed to mange, control, and monitor an SP system. Not all RS/6000 workstations are supported as control workstations. Central control is provided by the IBM Parallel System Support Programs (PSSP) executing on the CWS.

Each frame is connected to the CWS through a RS-232 link. SP-attached servers require two RS-232 connections to the CWS. So, if four frames and an SP-attached server are used, the control workstation should be configured

with at least six tty ports. An Ethernet connection from the CWS to each node is also mandatory.

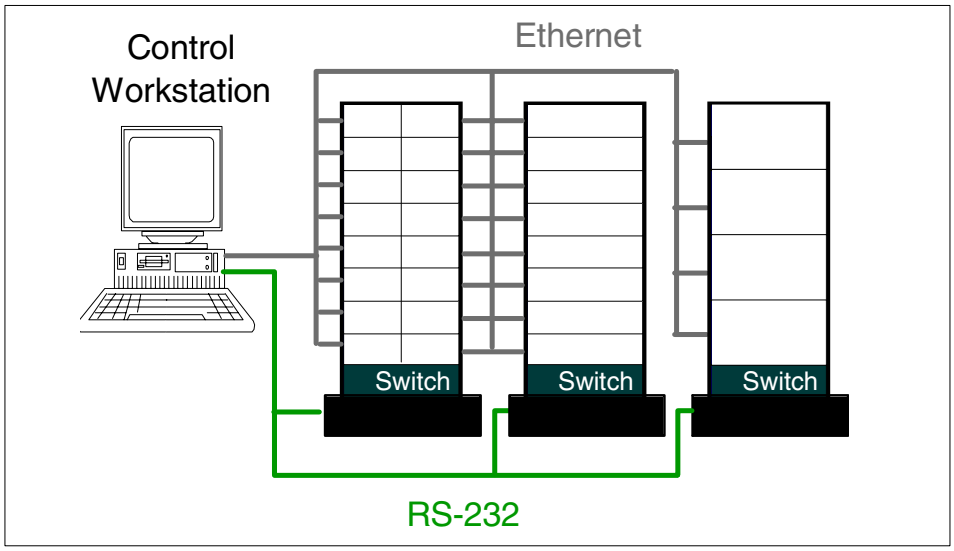


Figure 41. Control workstation interface

A wide range of storage devices can be attached to the SP system through the SP nodes. The storage devices that can be attached are the 7133 serial storage architecture (SSA), the 2105 Enterprise Storage Server (ESS), the 2102 Fibre Channel RAID Storage Server (FCRSS), the 2104 Expandable Storage Plus (EXP Plus), the 3995 optical library data server, the 3494 Magstar Virtual Tape Server (VTS), and others.

4.2.3 SP switch performance

The SP Switch provides the base communications performance capability. It provides a high-performance interconnection between the processors of the SP. This is essential if acceptable parallel application performance is to be obtained. In parallel systems, it is usual to discuss the performance of an inter processor communication fabric in terms of latency and bandwidth. Latency is the overhead associated with sending data between two processors, typically measured in microseconds (10^{-6} s). Bandwidth is the rate at which data can be transmitted between two processors, typically measured in MegaBytes per second (MB/s).

Latency and bandwidth over the SP Switch will vary depending on how it is measured. At the lowest level, we can consider the latency and bandwidth of

the switch hardware. At higher levels, which include the SP software, users may access the SP Switch via several different software communication paths. The time for processing the communication protocol stack is determined by the software path length and the performance characteristics of the processor.

In this section, we will describe the communication performance of the SP Switch for the key communication protocols available in the SP. These include the fastest protocol available, called user space, and TCP/IP performance. The high-performance user space protocol is most commonly used for technical and scientific applications via a message-passing interface (MPI). TCP/IP is an industry-standard protocol and forms the basis of many applications, such as NFS and FTP, which are used to support file systems and file transfers. TCP/IP is also widely used in parallel commercial database applications.

Hardware: peak communication performance

The SP Switch is a multistage network that is designed to allow communication bandwidths to scale linearly to many processors. Scalability is achieved by increasing the number of switch stages as the number of processors increases. For each additional switch stage, there is an additional latency. This is not detectable at the application level. As shown in Table 8, the hardware unidirectional bandwidth is constant at 150 MB/s; the bi-directional bandwidth is 300 MB/s. This can be considered peak performance, and it usually cannot be obtained by a user application.

Table 8. SP Switch peak hardware performance

Number of SP Nodes	Maximum Hardware Latency (microseconds)	Hardware Bandwidth (MB/s)	
		Uni-directional	Bi-directional
up to 80 nodes	1.2	150	300
from 81 to 512 nodes	2.0		

- This peak performance cannot be achieved by applications.
- The SP Switch MX2 adapter (MX2) is available on the 200 MHz POWER3 SMP nodes. This adapter has an on-board 125 MHz 603e processor attached to a 480 MB/sec Mezzanine bus.
- The SP Switch MX adapter (MX) is available on the 332 MHz PowerPC 604e SMP nodes. This adapter has an on-board 100 MHz 603e processor.

- The SP System Attachment Adapter (PCI Switch Adapter), available on the Enterprise Server Models S7A and S80, attaches on one end to a PCI slot in the server I/O drawer, and on the other to the SP Switch cable.

Processor performance

- 200 MHz POWER3 Processor
  The 200 MHz POWER3 Processor operates at a clock frequency of 200 MHz and delivers high sustained floating-point performance. The POWER3 SMP Processor chip has a 128 bit wide memory bus. This 16 byte bus width and a bus frequency of 100 MHz delivers a peak memory bandwidth of 1.6 GB/s (gigabyte is defined as 10^9 bytes, in this section).
- 332 MHz SMP Processor
  The 332 MHz PowerPC 604e SMP Processor operates at a clock frequency of 332 MHz and delivers high integer performance. The SP 604e processors are part of the RS/6000 PowerPC family of processors. The 332 MHz PowerPC 604e SMP processor chip has a 128 bit wide memory bus. This memory bus width and a bus frequency of 83 MHz delivers a peak memory bandwidth of 1.33 GB/s.

Several factors contribute to the communication performance that can be obtained by a user application. First, the switch is not the only piece of hardware to be considered. Communication performance depends on the processor, the memory subsystem, and the switch adapter. Therefore, when considering communication performance measurements, it is extremely important to understand the exact configuration of the system to which the data applies. In addition to hardware considerations, the system software contributes to the overhead involved in sending data between processors.

MPI/User space

On a distributed memory system like the SP, parallel scientific applications perform inter-processor communication via some form of message passing. Several different message-passing protocols exist. AIX PVMe, IBM's implementation of Parallel Virtual Machine (PVM) and Message Passing Library (MPL), are two examples of message-passing interfaces that can access the SP Switch through user space. Message Passing Interface (MPI) has been adopted as the industry standard for message passing, and it is expected that, over time, applications using MPI will predominate. IBM fully supports MPI on the SP via the Parallel Environment for AIX software product, so we will discuss the performance of the SP Switch for scientific applications in terms of what can be measured using MPI. Note that inter processor communication performance measured using MPL is very close to the performance measured using MPI.

Table 9 shows inter processor communication performance measured from a Fortran program with MPI calls, using user space. Latency is measured by sending a zero byte message between two processors using mpi_send and mpi_recv from the MPI library. It is calculated as half the time for a round trip between the processors for that zero byte message. Latency represents the time taken to set up a single message for transfer at the level of an application; it can be regarded as the overhead involved in transferring information between processors.

SP nodes and attached servers

The MX2 adapter latency is the lowest achieved on the current SP system.

Table 9. MPI user space performance with SMP nodes with a single MPI tasks per node

SMP Processor Type	RS/6000 Model Equivalent	SP Switch Adapter	Latency (microsecond)	Bandwidth (MB/s)	
				Uni-directional	Bi-directional
200 MHz POWER3 SMP	43P-260	MX2	21.7	139	170
332MHz SMP	H50	MX	23.5	83	86
262 MHz RS64II	S70	PCI	37.3	70	87

The 200 MHz POWER3 node with the 125 MHz MX2 adapter has a slightly lower latency than a 332 MHz SMP node with the 100 MHz MX adapter. This is due to the increased speed of the processor in the adapter. The MX2 adapter connects directly to the system Mezzanine bus with a throughput of 480 MB/sec. POWER3 has better MPI bandwidth than the 332 MHz SMP node because the POWER3 CPU can perform memory-to-memory copies faster. The results above were obtained using the non-threaded MPI library.

The 332 MHz SMP node with the MX adapter has lower latency and higher bandwidth than the PCI Switch-Adapter used in SP-attached servers. The superior memory bandwidth of the 332 MHz SMP node contributes to the increased MPI bandwidth. The MX adapter also connects directly to the Mezzanine bus with a throughput of 400 MB/sec. The superior design of the MX leads to improved MPI performance relative to PCI Switch adapter.

Measurements for the PCI Switch adapter are included for completeness, even though applications that use SP-attached servers will generally use IP rather than MPI (attached servers will generally be used for commercial

computing applications, while MPI is generally used by scientific and technical computing applications).

The performance data shown in Table 9 on page 107 was generated using the following hardware configurations. Because the measurements were memory-to memory, the processor memory and node internal disk storage configurations did not affect the results.

Multiple MPI tasks

PSSP 3.1 (Parallel Support System Programs) allows multiple user space processors per adapter (MUSPPA). Table 10 shows uni-directional and exchange bandwidth for multiple MPI tasks per POWER3 and 332 MHz SMP nodes. As the number of tasks per node increases, the aggregate memory to memory copy rate increases. The bandwidth through the MX/MX2 adapter also increases up to four MPI tasks, which is the throughput limits of the adapters.

The MX2 adapter is the limiting factor on POWER3, so the bandwidth with two MPI tasks per node is not much better than with one MPI task per node.

Table 10. MPI user space performance on SMP nodes with multiple MPI tasks per node

SMP Processor Type	Number of MPI tasks	Bandwidth (MB/s)	
		Uni-directional	Bi-directional
332 MHz SMP	1	83	86
	2	127	149
	4	128	162
200 MHz Power3 SMP	1	139	170
	2	140	185

TCP/IP performance

TCP/IP is a more common industry standard communication protocol used to transfer information between any two systems running IP. It is a robust protocol that supports multiple users and reliable transport of data. However, it supports networking functions not currently used by MPI such as multiplexing, so it requires higher processor overhead compared to the user space protocol using MPI.

The performance of the TCP/IP socket protocol on various nodes was measured using Netperf, a public-domain benchmark, and the results are listed in Table 11 on page 109. All Netperf measurements were memory-to-memory to keep slower devices, such as disks, from impacting

the performance. Note that in these tables, megabyte is defined as 2^20 bytes due to the way Netperf calculates its results. As with the user space measurements, TCP/IP bandwidths are largely determined by the speed with which the TCP and IP protocol stacks are processed. The processor memory copy rate also affects the maximum throughput rate.

It must be emphasized that the performance of the IP protocols family is a robust and complex function of the characteristics of the network, the processor, the processor's memory bandwidth, as well as a lengthy list of IP stack tuning parameters termed *network options*.

Table 11. TCP/IP performance with SMP nodes

Number of processors	Bandwidth (MB/s), uni-/bi-directional					
	200 MHz Power3 SMP node		332 MHz SMP node		262 MHz SP-attached Server	
	Uni	Bi	Uni	Bi	Uni	Bi
1	114.3	156.2	63.9	101.0	73.5	88.5
2	134.8	174.0	114.5	156.0	73.7	89.9
4	N/A	N/A	128.6	156.5	73.9	89.9

The POWER3 SMP node, compared to the 332 MHz PowerPC 604e SMP node, delivers between 12 percent and 78 percent better bandwidth. These improvements are primarily attributable to the difference in memory bandwidth of the nodes. The 262 MHz SP-attached server delivers excellent single process throughput. However, it is limited by the throughput of the PCI bus to which the adapter is connected. The PCI adapter uses a 132 MB/s PCI bus and a single adapter in that bus can only get 90 MB/s under a real application.

Bandwidth is the maximum obtainable both uni-directional and bi-directional over TCP between two identical applications running on two identical SMP nodes. All Netperf measurements were memory-to-memory to keep slower devices such as disks from impacting the performance.

The SMP nodes can take advantage of multiple processors if there are multiple IP connections running at the same time. The results in Table 11 show that as you increase the number of processors or TCP streams, the aggregate throughput increases for all but the SP-attached server. If only one TCP/IP socket is used, the maximum throughput will be similar to the single-processor throughput no matter how many processors are configured

in the node. A single TCP/IP socket currently cannot take advantage of multiple processors due to the single-threaded nature of memory-to-memory copies and the TCP/IP stack.

SP switch router

The SP can send switch traffic to outside networks through an SP Switch Router. Sold exclusively by IBM, this is a combination of the Lucent Technologies (formerly Ascend Communications) GRF router and the IBM SP Switch Router Adapter that connects to the SP Switch fabric. The SP Switch Router node only supports IP traffic. Its performance was measured using the Netperf benchmark described earlier. Table 12 shows the peak aggregate throughput of the SP Router node.

Table 12. TCP/IP performance through the SP Router nodes

Adapter type	Bandwidth (MB/s)	
	Uni-directional	Bi-directional
SP Switch Router Adapter	100	200

The node internal disk storage configuration did not affect the results. Not all nodes in the test configuration were needed to sustain the peak throughput of the SP Router node.

The SP Switch vs. the gigabit ethernet

The gigabit ethernet performance is close to the SP Switch performance in theory, but the gigabit ethernet performance falls far short in a real environment. The 332 MHz SMP 4way nodes, using the current SP Switch adapter, deliver data over TCP/IP to applications at 156.5 MB/s (see Table 10 on page 108), more than three times the peak traditional ethernet traffic rates over gigabit ethernet (about 42.1 MB/s per adapter using 2way TCP stream 2 sessions with standard 1500 byte MTU on RS/6000-F50 332 MHz 4way). At this throughput rate, a little more than one CPU is consumed sending SP Switch traffic, and 65 percent of a CPU is consumed sending gigabit ethernet traffic.

4.2.4 Shared disk components of Parallel System Support Programs

The components of Parallel System Support Programs (PSSP) that help you create and manage virtual shared disks are IBM Virtual Shared Disk (VSD), Hashed Shared Disk (HSD), and IBM Recoverable Virtual Shared Disk (RVSD). This section is an introduction to these subsystems.

4.2.4.1 IBM virtual shared disk (VSD)

IBM virtual shared disk is a software layer between applications that use the virtual shared disks and raw logical volumes that are managed by the AIX Logical Volume Manager (LVM). VSD allows requests for data blocks to be resolved from locally attached disks or from disks attached to other processing nodes (see Figure 42). When the data block requested exists on another processing node, VSD issues remote I/O requests across the SP Switch. Currently, the primary application using VSD is Oracle8 Parallel Server. The performance impact of VSD depends on the application and block size of the databases.

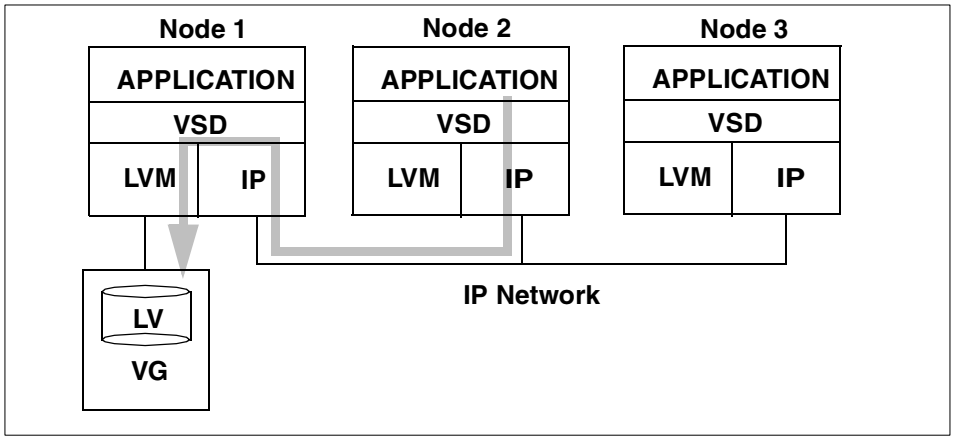


Figure 42. VSD implementation

4.2.4.2 Hashed Shared Disk (HSD)

The Hashed Shared Disk is the subsystem that works with the VSD subsystem. The HSD has a data striping device driver that distributes data across multiple nodes and multiple virtual shared disks. In this way the HSD reduces I/O bottlenecks (see Figure 43 on page 112). Refer to *PSSP: Managing Shared Disks*, SA22-7349, for more information.

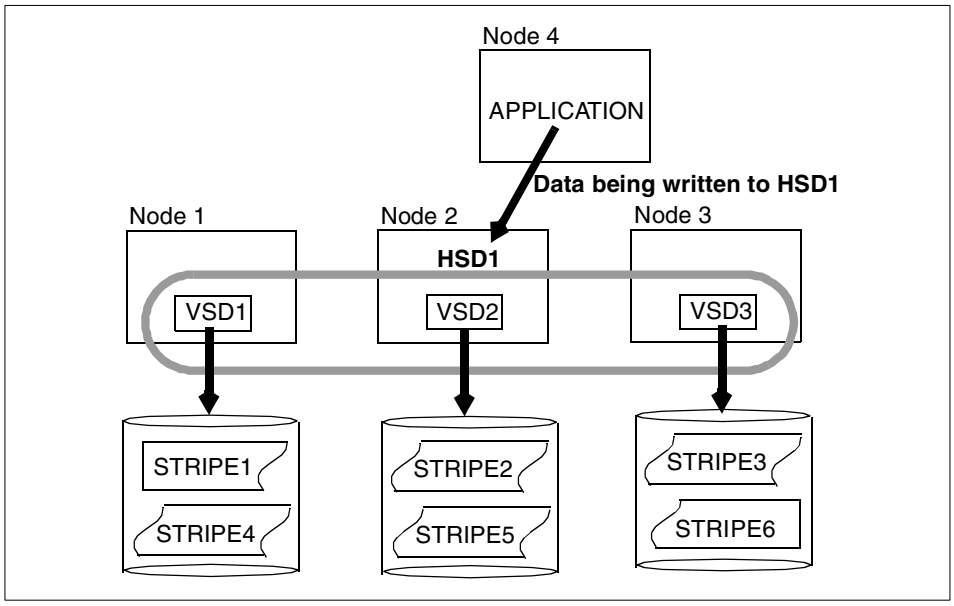


Figure 43. HSD implementation

4.2.4.3 IBM Recoverable Virtual Shared Disk (RVSD)

IBM Recoverable Virtual Shared Disk (RVSD) is the subsystem that provides recoverability of your virtual shared disks if a node, adapter, or disk failure occurs. If you use the VSD and the server node fails, access to the disk is lost until the server node is rebooted. By using the RVSD component and twin-tailed disks or disk arrays, you can allow a secondary node to take over the server function from the primary node when certain types of failure occur. The RVSD provides transparent failover to a secondary server node if the primary server node for a set of virtual shared disks fails (see Figure 44 on page 113).

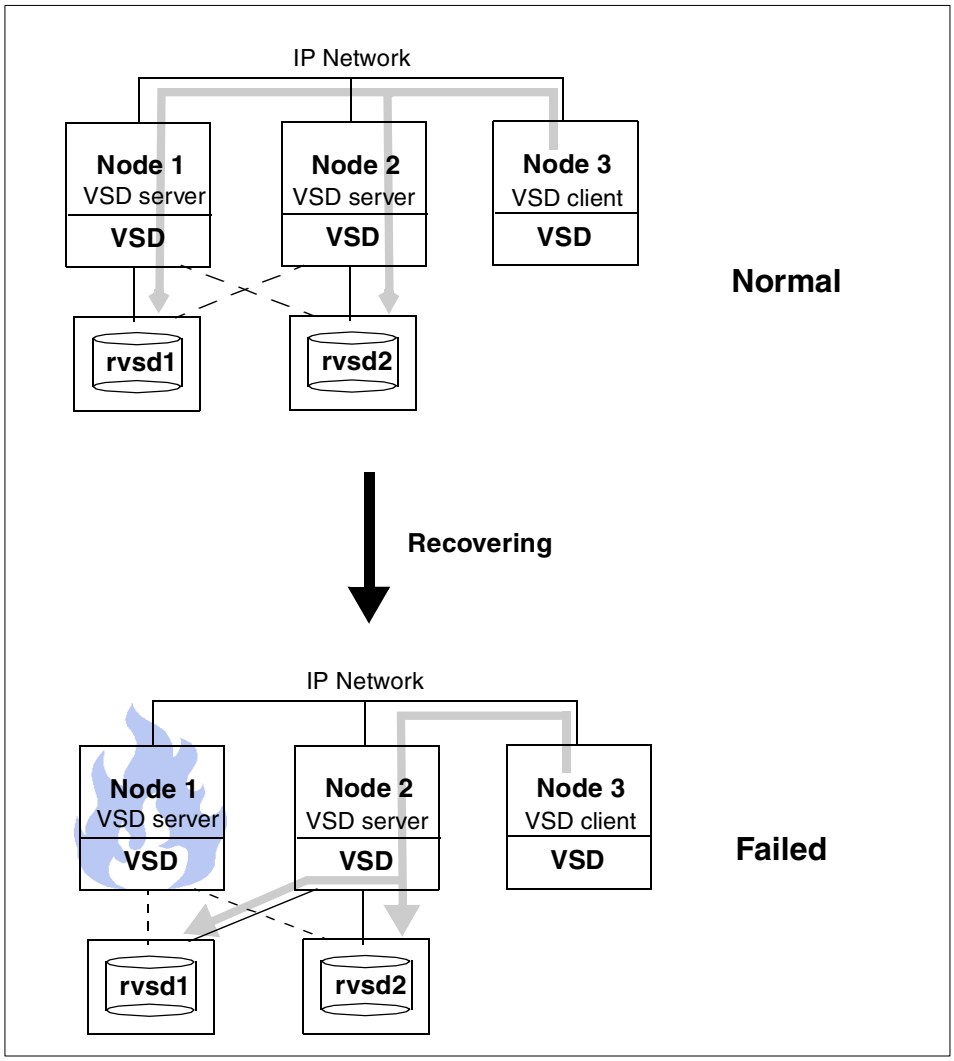


Figure 44. RVSD implementation

4.2.5 Sizing and configuring a control workstation

The control workstation links to the supervisory subsystem of the SP system through the RS-232 asynchronous link. This link has to be provided for each SP frame to be controlled by the control workstation. Thus, if four frames are planned, the control workstation should be configured with at least four tty ports. This link uses a custom protocol for supervisory and tty data communication (point-to-point). The RS-232 operates at 19200 baud. The

byte-oriented interfaces of the RS-232 will inherently put a heavy load on the CPU of the control workstation, especially at faster baud rates.

4.2.5.1 Selecting the control workstation

The control workstation is configured separately from the SP configuration. It must be an RS/6000 machine as mentioned before; not all workstations are supported as a CWS. The currently supported RS/6000 models as a control workstation are:

- 7026 H10 and H50
- 7025 F3
- 7025 F40 and F50
- 7043 140 and 240

Usually a LAN connects the control workstation to the nodes through the Ethernet. With the external LAN adapter, the workstation can also act as a gateway to external networks when no direct connections (FDDI, token ring) exist from the nodes. The SP Ethernet is the network that connects all nodes to each other and to the control workstation in the SP system. We recommend that the SP Ethernet is only used to control the SP system.

If the control workstation also acts as a file server of some kind, enough storage space must be available. This depends on the type of server it is. Usually, the control workstation is the Boot/Install server, so you have to reserve enough storage space for mksysb image files of your SP nodes.

4.2.5.2 Using the High Availability Control Workstation (HACWS)

A single control workstation is an Single Point of Failure (SPOF). If you need high availability, configure an HACWS. An HACWS provides the most reliability for the control workstation and the SP system. All hardware and software components are redundant, which allows recovery from any single failure. For more information about the HACWS, see the *Planning for a High Availability Control Workstation* chapter of *IBM RS/6000 SP: Planning Volume 2,* GA22-7281.

4.2.5.3 Sizing the disk space

The PSSP specific data is kept in a dedicated subdirectory, */spdata*. It contains, among other items, the mksysb and installp file sets. We suggest you create a separate volume group for the /spdata file system. These file sets require a minimum of 2 GB of disk space. You will require additional disk space if you need to support multiple AIX and PSSP release levels and multiple mksysb images. If you have not done so already, use *IBM RS/6000 SP: Planning Volume 2,* GA22-7281.

Following is an approximation of free space required in various filesystems and volume groups:

- The recommendation for rootvg is 2 GB, providing that /spdata is installed in a separate volume group.
- /var requires at least 20 MB of free space. Most of the PSSP logs are stored here. The actual disk space used by these logs depends entirely on the types of problems that occur on your system and their frequency. The /var filesystem should be monitored frequently to ensure that there is always sufficient free disk space for new logs.
- /tmp requires at least 16 MB of free space.
- We suggest that you create a separate file system for /tftpboot. Each lppsource level requires a minimum of 25 MB. You should also consider future AIX installations. We suggest 25 MB x (number of AIX versions+1). For example, if you are installing AIX 4.2, 4.2.1, and 4.3, then you will need 25 MB x (3+1) = 100 MB.
- Downloading all of the AIX filesets to the lppsource directory requires approximately 1.5 GB of disk space. Downloading only the minimal filesets requires approximately 500 MB. Each mksysb image may vary between 100 MB and 700 MB. A simple example for AIX 4.3.2 is:

  lppsource + mksysb_images + pssp_lpp_images + SPOT = total disk space

  lppssource = 500 MB

  mksysb_images = 300 MB

  pssp_lpp_images = 350 MB

  SPOT = 200 MB

  500 MB + 300 MB + 350 MB + 200 MB = 1.35 GB

  You should have at least 1.35 GB of free disk space for the /spdata file system. To be on the safe side, reserve 2 GB of disk space for /spdata. For multiple AIX and PSSP version coexistence, this figure will increase considerably due to the disk space taken up by different versions of mksysb images, SPOT, AIX, and PSSP filesets.

4.2.6 Sizing and configuring an SP system

Since the initial introduction of the IBM RS/6000 SP systems, IBM has continued to announce new node types with faster performance and better price/performance ratios. Today, all the types of SP nodes are SMP nodes.

4.2.6.1 Node selection

When selecting nodes for an SP system, we can use the methodology shown in Figure 45 as a general rule.

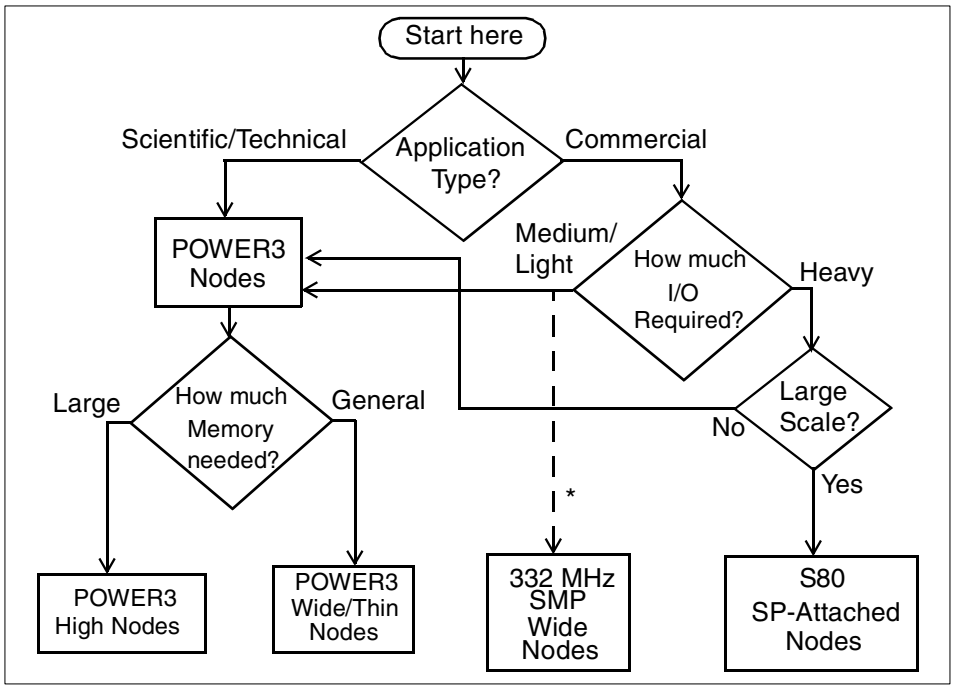


Figure 45. Node type selection

* The 332 MHz PowerPC node is supported but a new POWER3 node is recommended.

1. Scientific/Technical Application
   A scientific/technical application generally needs high floating-point processor performance. The POWER3 chip has higher floating-point processor performance than PowerPC 604e. The SPECfp_base_rate95 of the POWER3 SMP 8way High Node is 1760, and that of the 332 MHz SMP 4way Wide/Thin Node is 364.

   Also, if your application requires large memory, the high node is recommended. Because the maximum memory size of the POWER3 SMP High Node is 16 GB and that of the POWER3 Wide and Thin Node is 4 GB.

2. Commercial Application
   If you use a commercial application, consider how much I/O is required. The S80/S7A SP-Attached Node is recommended when very heavy I/O

and very large scalability are required. If the I/O is heavy but not large scale, you can chose POWER3 SMP Nodes. If the I/O is medium or light, 332 MHz SMP Wide Node is recommended. Commercial applications generally need many PCI slots.

4.2.6.2 Parallel sizing factors

Sizing a scalable system for commercial applications is different from sizing a non-scalable system. It depends on the type of commercial applications, and the following factors should be considered. Usually there are two types of commercial applications predominately used on a scalable system; Decision Support System (DSS) and On-line Transaction Processing (OLTP).

The performance metrics for the system designed for OLTP applications are:

- Throughput
- Response time
- N-node scale-up

The performance metrics for the system designed for DSS applications are:

- Throughput
- Response time
- N-node scale-up
- N-node speed-up
- Database scale-up

Scale-up and speed-up

In the case of scale-up, twice as much hardware capacity typically should perform twice the work in the same elapsed time. Scaling up a scalable system can be done in two ways, as shown in Figure 46 on page 118. The first case occurs when increasing the data and resources equally. In this case, perfect linear scaling will produce the same elapsed time. In the second case, the amount of data increases while the resources stay constant. In the latter case, elapsed time should increase proportionally to the workload increase.

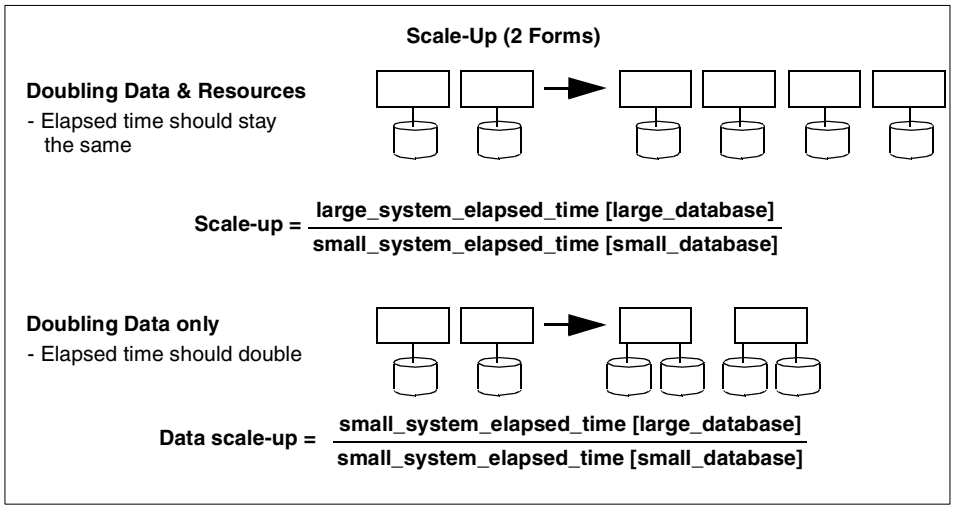


Figure 46. Scale-Up

In the first case, the ratio of resources to workload remains constant as more data and more CPUs are added. This is the primary reason for the existence of distributed memory parallel machines. Once the ratio of *CPU speed to disk* is ascertained, the parallel machine can grow to any possible database size, assuming perfectly linear scale-up (100 percent scale-up). Simply stated, twice the hardware and twice the data execute in the same elapsed time if scale-up is linear.

In the second case, the workload is increased while the resources remain constant. Increasing the workload within a single processing unit will scale favorably until a threshold is crossed in one or more of the resources. At that point, the elapsed time degrades significantly. When only the data is increased, computers rarely scale up linearly. Typically, a system will handle the workload up to a point, and then the phenomena of *thrashing* occurs. Once thrashing begins, efficiency decreases and more resources are required to achieve acceptable performance.

Most commonly, scale-ups are required in operational OLTP, interactive client/server, and batch applications. Thus, if we know that one CPU can support 200 end users, we would like to be able to combine four CPUs to handle 800 end users. If a system cannot scale up, the programmers may have to rewrite portions of the application or split it across two systems to support a larger number of end users. The objective of scale-up is to increase workload capacity primarily through replication of resources.

Speed-up is different. In this case, our goal is to use twice as much hardware to perform the same task in half the time. Typically, applications that require speed-up are strategic analytical applications such as data warehousing and data mining. Allowing end users to submit twice as many problems to be solved per day allows them to refine the accuracy of the result and meet deadlines. Inefficient speed-up results in fewer transactions per time period, slowing down the overall pace of the business productivity. The objective of speed-up is to improve response time to end users primarily through partitioning of workloads across resources. Speed-up is therefore more of a software concept that relies on the underlying hardware to scale up. Thus, the industry speaks most often about scalability when comparing computer systems of any size.

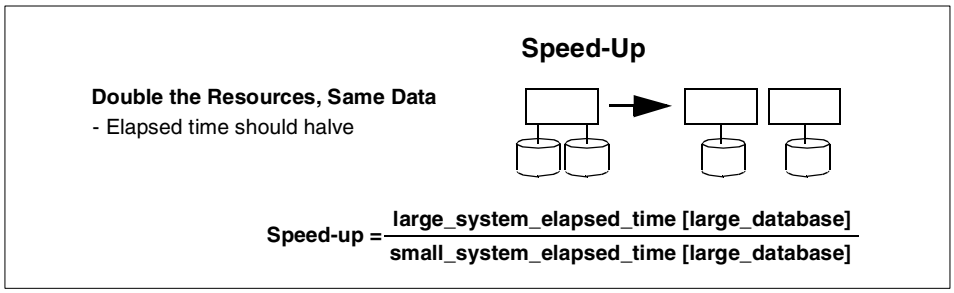


Figure 47. Speed-Up

According to Gartner Group, a system must achieve better than 80 percent linear scaling to be considered scalable and useful. This means achieving better than 80 percent scale-up or speed-up across a wide variety of production workloads. Because it is not possible that every conceivable workload will scale in this range, the best balance is achieved when most common production workloads run at better than 80 percent. While some may run at efficiencies of 60 percent or even as low as 40 percent, occasional low efficiency is tolerable. Consistently low efficiency is not.

4.2.6.3 Amdahl's Law

The speed-up factor on a parallel machine depends on what portion of jobs or sub tasks can be parallelized. It can be shown in Figure 48 on page 120.

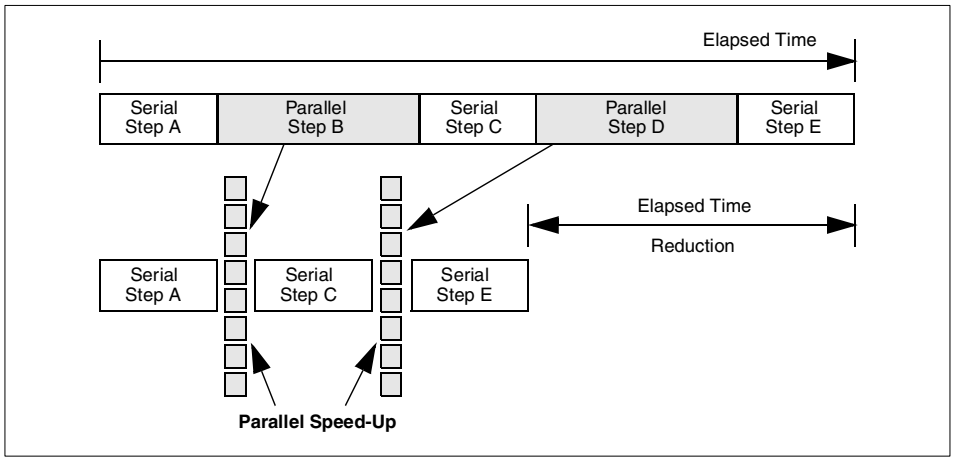


Figure 48. Amdahl's Law

In the above diagram, we see that some tasks can be parallelized. This will speed up the whole process. However, the performance is always limited by the slowest component because we cannot parallelize everything in the parallel machine.

In distributed memory parallel machines, the most common reason for not achieving 100 percent scale-up or speed-up is a large serial processing component in the workload. A serial activity, for example, would be the emission of a million data rows to a LAN client. While retrieving the rows from the database can be done in parallel, the emission to the Ethernet or LAN cannot. Therefore, a major portion of the elapsed time is controlled by a low-speed, serial activity lowering overall efficiency.

4.2.6.4 Commercial applications

The SP system is used in many ways because the SP nodes have many characteristics such as high performance, high scalability, easy administration, and so on. In this section, we introduce some commercial applications of the SP system.

Large web server

The web server is responsible for serving up everything from static pages to invoking various applications through interfaces such as Common Gateway Interface (CGI), FastCGI, web server APIs, and servlets. Because of this varied workload careful planning is needed to ensure that the web server is processing the request in the most efficient manner. Very large web servers

like the Nagano Olympic web site or the Wimbledon Tennis Championship web site require:

- High throughput
- Short response times
- Superior scalability
- Optimized to withstand the load
- High reliability

Web servers provide two types of pages; static and dynamic. Static pages are served from the file system. By contrast, dynamic pages are created on-the-fly by server programs that execute at the time a request is made. Dynamic pages are essential for situations like the Olympic Games where Web pages are constantly changing. However, dynamic Web pages are often expensive to serve. A static page typically requires 2 to 10 milliseconds of CPU time to generate. By contrast, a dynamic page can consume several orders of magnitude more CPU time to generate.

In the Nagano Olympic, the base requirement for the server were 100 million hits/day, response time of 30 seconds, and 100 percent availability.

In addition, there were some implied requirements. Hits were never evenly distributed throughout the day; there were peaks and troughs as various countries woke up, traveled to or from work, and had lunch breaks.

To maintain the target performance at all times, the servers must be sized to handle the largest of these peaks. In addition, the target was to retain availability and performance in the event of a failure, up to and including a complete site failure.

Node Sizing

There is no major web server that is parallelized. Using and sizing an SP node as an web server is the same as other RS/6000 models. Sizing of an web server system is discussed in Section 7.5, “Web server sizing” on page 306.

Server consolidation (serial applications)

In a server consolidation environment, we will not be running parallel applications, so the most important consideration is whether the particular applications to be implemented support SMP processors and will take advantage of such nodes.

In general terms, server consolidation applications are likely to be among the best ones for running on high nodes within the SP. Adopting high nodes will allow us to minimize the number of nodes required, but assuming that we wish to isolate applications, we can still have enough nodes within the SP to allow for some level of potential redundancy from an availability point of view.

One of the issues when running server consolidation applications is the performance available on any one SP node. By definition, server consolidation applications will not be able to utilize more than one node, and therefore, as the application grows in performance terms, the maximum performance available on any individual node can be a deciding factor.

The introduction of the high nodes is a major advantage now, in that the customers can grow their applications from the smallest of 1 way SMP thin nodes right through to 8 way high nodes with greatly increased performance. The binary compatibility of AIX running on each node means that the same application can run unchanged as we move from one node to another.

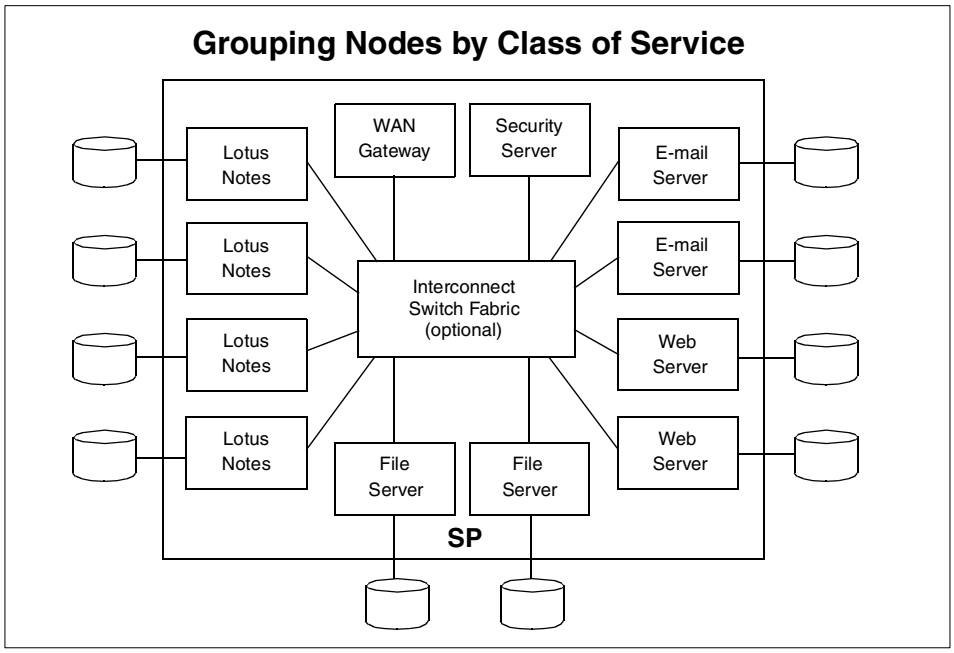


Figure 49. Server Consolidation

Parallel databases

There are two types of RDBMS supported in the SP system, shared disk systems (I/O shipping) and shared nothing systems (function shipping). Each

of these types has a different approach and implementation to take the benefit of the parallel function on the SP system. We will look at both of these systems in order to understand sizing requirements and node types that can take advantage of the parallel database architecture.

- **Shared disk systems (I/O shipping)**

  Oracle Parallel Server (OPS) is a parallel RDBMS using the shared disk system. For serial applications (not parallel), the Oracle RDBMS is already running on the RS/6000 SMP systems and has been for some time. In other words, it exploits an SMP architecture. Therefore, for any serial applications, if a high node is selected for performance or other reasons as described in this chapter, we are likely to find the high node a good solution.

  As we consider parallel applications, we can see that a parallel database consisting of a number of high nodes also running Oracle is likely to be a good solution, if high nodes are required.

  The option of running a parallel database with four eight-way high nodes, rather than 16 SMP wide nodes, for example, is likely to be the preferred option, as we will have far fewer nodes to maintain, manage, and administer. The situation is not quite so clear if we were to mix nodes, for example wide and high nodes, within an Oracle parallel database solution.

  How easy would it be to make sure that each node took its fair share of the workload according to its potential performance? Oracle8 is known to have an intelligent optimizer. It notes that certain nodes get through their work more quickly than others, and adjusts subsequent workloads accordingly. In this scenario, we would expect the high nodes over a period of time to accept more of the workload when compared to other, less-powerful nodes. Further experience with high nodes in real life - and seeing how well they may work with such an optimizer that builds up a history of node performance - will give you more detailed information in this area.

  When running an Oracle parallel database on nodes that include high nodes, we will not need to run multiple Oracle instances on each node. We will run one instance of the Oracle software that will take into account the multiple processors and manage the resources.

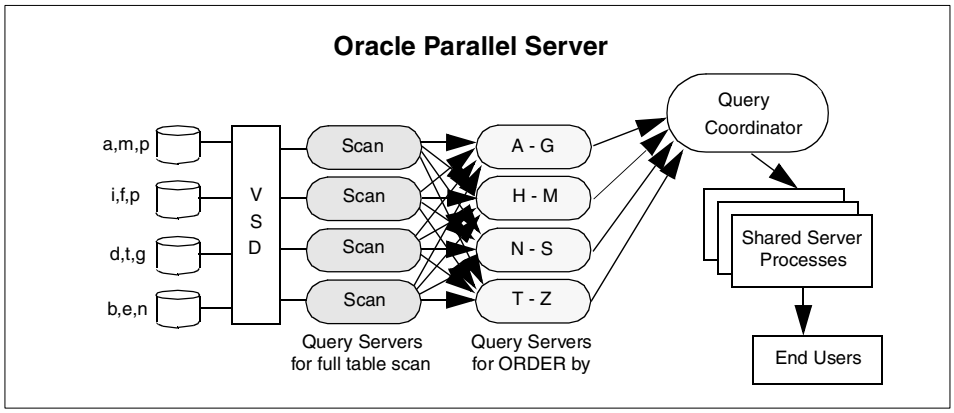


Figure 50. Typical Oracle Parallel Server Implementation

- **Shared nothing systems (Function shipping)**

  There are three RDBMSs operating on the SP with a shared nothing architecture; DB2 UDB EEE, Informix XPS, and Sybase. While such databases are often referred to as *parallel*, it is more accurate to label them *partitioned databases* because in each case, the non-partitioned, or single partition, version of the database exhibits parallel characteristics such as SMP enablement.

  If the database(s) are relatively small, such as those within a server consolidation environment, then single partition databases can be run on single (usually SMP) nodes. However, for large databases single nodes are not sufficient and the clustered approach of a multi-partition database is required.

  The partitioned IBM RDBMS, DB2 UDB EEE, uses a shared nothing architecture and therefore operates in a different way than the shared disk architecture of the Oracle Parallel Server.

  DB2 UDB EEE partitions or *splits* the data using a hashing algorithm, spreading it evenly across the nodes within the parallel database. Each database partition *owns* and has sole access to a section of the data and operates in many ways like a small independent database with its own bufferpool and logs while still presenting a single database image to the application. When a query is presented to the database, it is broken into sub-queries that are shipped to each partition for resolution (hence the phrase *function shipping*) and the result sets from each partition are merged together by a coordinator process that passes the final results back to the application.

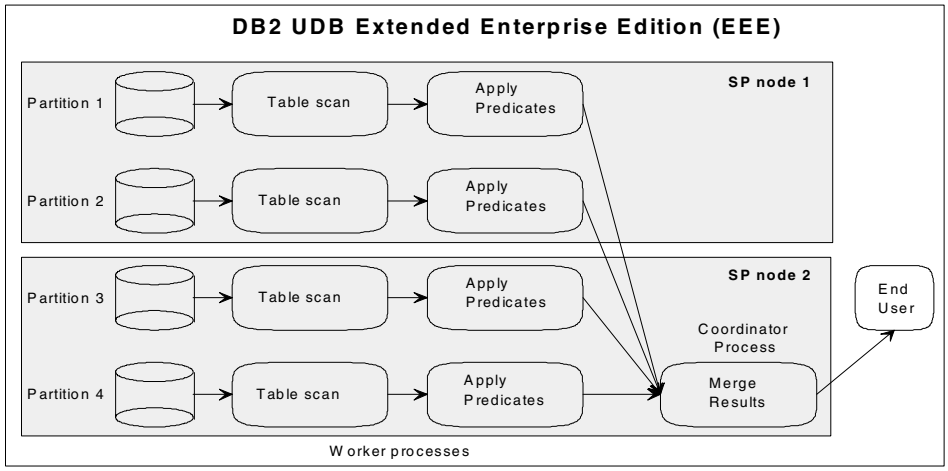


Figure 51. DB2 UDB EEE implementation with 2 database partitions per node

DB2 UDB EEE partitions cannot span multiple nodes, but it is possible to have multiple partitions on a single node. While each partition operates as a database in its own right including SMP enablement, it can benefit performance to have more, smaller partitions and hence less data to scan during queries, more log devices, and so on. From an administration perspective, a system with fewer partitions is easier to manage.

Again, as with all parallel databases, it is best to use equal power and similarly configured nodes across the entire environment. The use of multiple partitions on some nodes or the alteration of the data distribution allows non-uniform systems to remain balanced, however.

Parallel database workloads

There are two distinctly different workloads that can run against a parallel or partitioned database; Decision Support and on-line Transaction Processing.

- **Decision Support Systems (DSS)**

  DSS or data warehouse applications are characterized by very complex query transactions against large databases. Typically there are few users and the queries are either summarizing (aggregating) the detailed data for further analysis by other, more numerous users or they are ad-hoc operations, varying greatly in size, frequency, and complexity.

  Applications that belong to this category are ad-hoc analysis, profitability, credit card use, target market analysis, cross selling, and inventory management. The response time for this kind of application can be very long (minutes and hours). The industry-standard benchmarks for the DSS

workload are TPC-H and TPC-R. For detailed information about TPC-H, see Section 6.3.3, “TPC-H” on page 234. For detailed information about TPC-R, see Section 6.3.4, “TPR-R” on page 238.

DSS users require the twin strengths of scale-up and speed-up.

Speed is needed both for queries and for data maintenance. Fast response times, even for complex queries, permit the user to issue multiple refinements to a query. If unusual results occur, fast response time is needed to further explore the anomalies or surprises in order to guarantee the results are accurate. More importantly, speed is required to ensure data can be loaded and summaries built as rapidly as possible, and for maintenance such as backups to be performed with minimal impact.

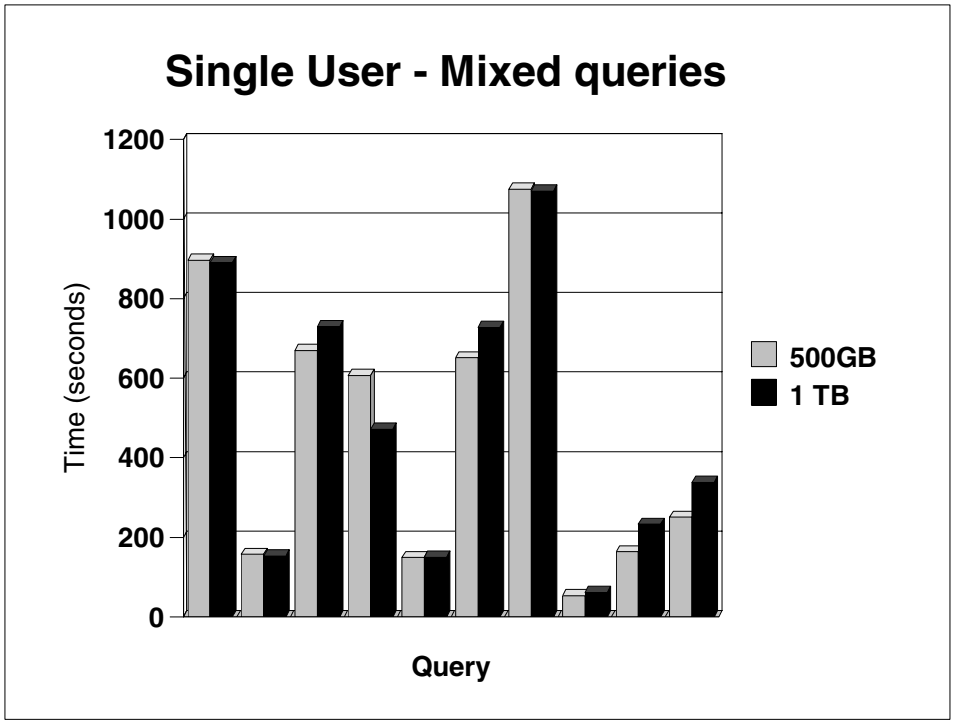


Figure 52. DB2 UDB EEE scalability test (1)

Scalability is also critically important to DSS environments. As more and more data is accumulated in the database, it is desirable to maintain the same response times for commonly used queries. If the database software is scalable, once the proper ratio of data to computation power is known

for a given data warehouse or DSS, scalability ensures that more nodes and data can be added in that ratio while performance is maintained.

DB2 UDB EEE, for example, scales almost linearly for most utilities and queries from 2 partitions up to many tens or even hundreds of partitions as shown in the following example. Scaling from a single partition to multiple partitions does incur a multi-partition overhead, so care needs to be taken when predicting parallel performance from a single partition database.

Figure 52 on page 126 shows how DB2 UDB EEE scales for DSS queries. The scenario was to double the amount of data from 500GB to 1TB and double the system resources (CPU, memory and disk); by doubling the number of partitions from 24 to 48, each partition remained the same size and, as can be seen from the results, a high degree of scalability was achieved, averaging 96.8 percent.

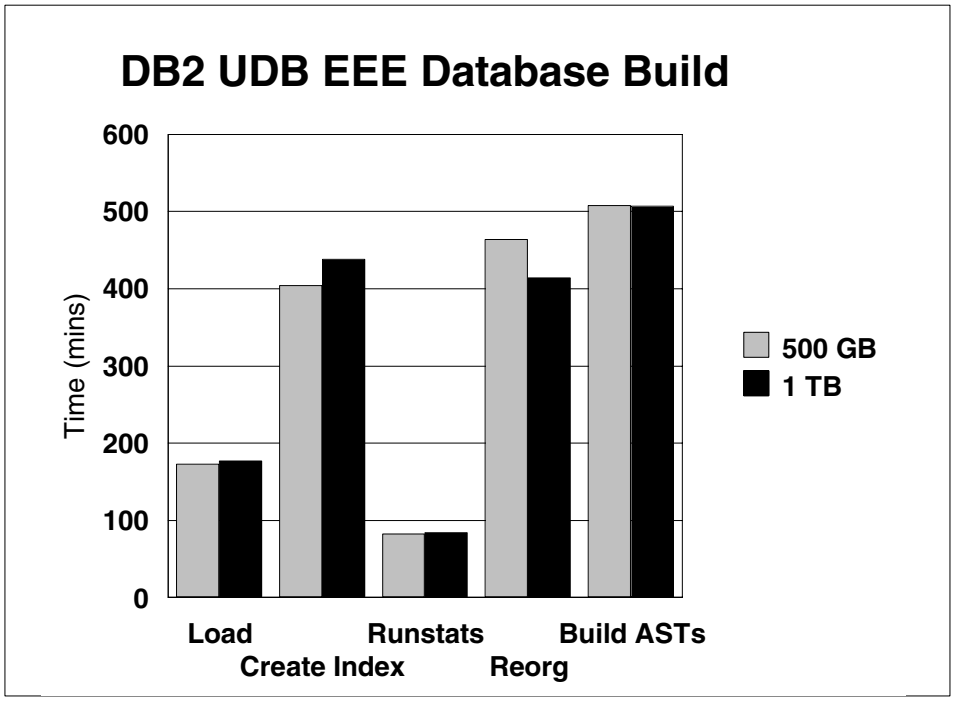


Figure 53. DB2 UDB EEE Scalability test (2)

Figure 53 shows how the database utilities scale in the same test. With data warehouse systems, it is frequently the data maintenance operations that drive the sizing requirements, rather than response times.

As stated earlier, on an SP the database will either run on a single node or in parallel (partitioned) across multiple nodes if the most powerful SMP node is not powerful enough.

Data warehouses typically hold many hundreds of gigabytes of data, which may be manageable on a large S80 SMP machine but not on a 4-way SP node, so large databases tend to span many SP nodes. In the DSS environment, there tends to be a premium on I/O performance because:

- The data is too large to be cached in memory
- Data loading is usually on the critical path
- Data maintenance causes large volumes of I/O

So avoiding I/O contention with multiple paths to disks is important; together with a common requirement for high availability, the typical DSS node is wide and well configured for memory.

With regard to the SMP processors, DSS performance and scaling is expected to be good because a relatively small amount of time is spent performing kernel activity compared with OLTP. As processors are combined within an SMP system, it is usual to get a diminishing return on the investment as additional processors are added. However, with a shared-nothing architecture adding more partitions and keeping the partition size will minimize this effect, and eliminate it for certain queries and utilities. Hence moving from a single partition on a 2-way node to two partitions on a 4-way node should scale very well, depending of course on the dynamics of the specific workload.

- **On-Line Transaction Processing (OLTP)**

  OLTP applications usually demand rapid interactive processing for a large number of simple transactions. Normally, an OLTP transaction has no floating-point calculations and has poor locality of code and data. OLTP applications require a fast, predictable, and consistent response time to satisfy user workloads. The industry benchmark for OLTP is TPC-C (see Section 6.3.1, “TPC-C” on page 231).

  In the parallel machine, data must be partitioned to enable scale-up, and the programming model must change accordingly. Usually, an OLTP machine has three major functions; application, transaction, and database manager. Sometimes in a small OLTP environment there is no need for a transaction monitor (transaction router) to simplify the configuration. In a parallel machine, each of these parts runs on one or several nodes and communicates to each other through high speed network or LAN. Figure

54 shows the simplified diagram of OLTP implementation on a parallel machine.

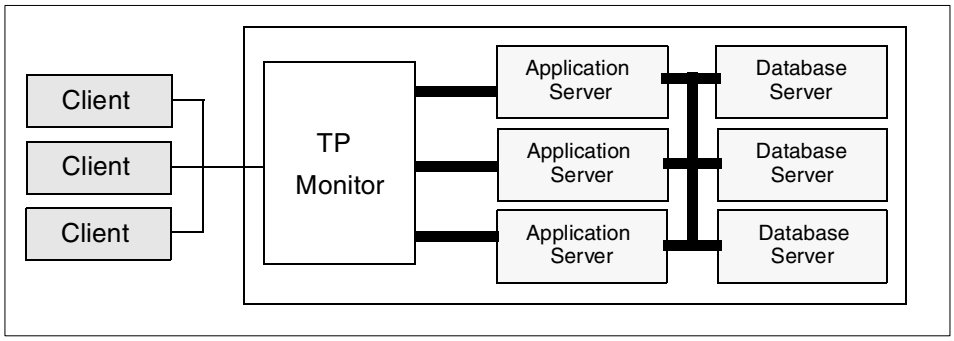


Figure 54. Typical OLTP Implementation Diagram

A transaction processing (TP) monitor such as Tuxedo or CICS/6000 allows the application to invoke a transaction on any node in the network based on control information from the client station. Thus, a transaction invoked by the client may contain data that the TP monitor uses to route the *run transaction* invocation to a node considered optimal by the application designer. This added sophistication is needed only when the OLTP transaction rate is extremely high and the response times must adhere to strict, sub-second performance. This is because the *routed transaction* gains less than 100 milliseconds of response time in most cases.

In the above diagram, the bar between the application server and database server is the interconnection fabric. In the SP machine this bar represents a message-passing event, running between 50 - 200 microseconds, depending on the vendor(s). While this seems like a dramatic difference, it only changes the response time for the end user by less than 1/5 of a second, if that much.

The TP monitor logic that does the routing can be placed in a node of the parallel machine or the client station. The partitioned database servers may be separate database instances, or may appear as a single database to the application. To achieve a single image database, a parallel database manager is required.

Node Sizing

In most situations, OLTP is likely to be reasonably well-suited to run on SMP nodes. We would usually expect a lot of users who are each running relatively light transactions (when compared with complex transactions in a DSS, for example). This is more suited to an SMP environment, where we can spread the transactions across the CPUs more easily. In effect, OLTP workloads have an element of parallelism built into them as a result of the fact that we typically have a large number of relatively light transactions. In each application case, it is important to investigate whether the application in question will support SMP processors.

One of the factors that will significantly effect scaling of performance across nodes in a parallel database on the SP will be the amount of I/O that is performed on disks attached locally (on the actual node that requests the I/O) as opposed to I/O to disks that are remote from this node. Clearly, the more disk activity that is local, the better the scaling. TPC-C transactions, for example, tend to perform less than 15 percent of their I/O remotely. In contrast an application performing 50 percent of its I/O to remote disks would not scale so well. Such factors need to be considered as we size our SP system for specific applications.

4.2.6.5 Scientific applications

Scientific applications depend on single processor performance (integer, floating-point, and memory subsystem) of each node to achieve a good scientific performance on the SP system. Another important factor is the communication subsystem between nodes. It must have a low Message Passing Interface (MPI) latency, high bandwidth, and good TCP/IP performance. A balanced system can produce a good application-level performance.

Node Sizing

When selecting node types for a typical scientific and technical application, we will usually have less of a problem in terms of which nodes to utilize within the SP system. These applications will need the higher floating-point performance of the POWER3 nodes. If large amount of memory are required, the POWER3 SMP high nodes are recommended. The individual PowerPC 604e processors do not have high floating-point performance and should not be used for scientific and technical applications. Therefore, combinations of POWER3 SMP nodes are likely to be the best solution. However, in a scientific and technical environment, there is often a requirement for commercial type workloads, such as file servers, print servers, and other such servers for which an SMP node can be used.

4.2.7 Resources

- *RS/6000 SP: Practical MPI Programming*, SG24-5380
- *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment*, GA22-7281
- *PSSP: Managing Shared Disks*, SA22-7349
- *Understanding and Using the SP Switch*, SG24-5161
- *RS/6000 SP Switch Performance* white paper, Version 3
- *Nagano Olympics Web site runs on RS/6000 SP* white paper
- `http://www.rs6000.ibm.com/resource/technology/gig_ether.html`
- *RS/6000 Systems Handbook*, SG24-5120

Chapter 5. Hardware

This chapter provides relevant information on processor and memory features as well as actual hardware products such as disk storage, asynchronous communication adapters, LAN/WAN adapters, and graphics adapters. Section 5.3, “Storage” on page 148 also explains the characteristics of various RAID levels as well as the performance impact of different AIX configuration options such as Mirrored Write Consistency (MWC).

5.1 Processors

IBM has developed industry-leading microprocessor fabrication technologies. These technologies are copper circuitry and Silicon-On-Insulator (SOI) on Complimentary Metal Oxide Semiconductor (CMOS) chips. The net effect of using copper circuitry is increased clock speeds, smaller die sizes, smaller channel lengths, and lower voltages. SOI protects the millions of transistors on a chip with a thin layer of silicon oxide, reducing harmful electrical effects that consume energy and hinder performance. These technologies, which contribute to higher performance and reduced power requirements, are the basis for enhancements to IBM’s current POWER3 processors and POWER4 Gigahertz processor.

Efficient pipelining of instructions and data allows RS/6000 to provide exceptional performance. However, this performance is heavily influenced by the type of application being measured and the actual design of the code being executed. Applications that run primarily in cache such as the LINPACK benchmarks will yield results comparable to those of the synthetic benchmarks. Today’s processor works at very high speed and spends a large percentage of time just waiting for information. The faster the processor, the more it will have to wait for data from the main memory. As an example, some processors running in commercial environments can spend 10 to 50 percent of their time stalled, waiting for instructions or data. This idle time is not reported by the system (`vmstat`, `sar`) as the system thinks the processor is busy. This shows that the memory subsystem (caches, buses, bandwidth, and latency) design is a key point for computer performance.

Cycles per instruction (CPI) and instructions per cycle (IPC=1/CPI) are common measures of processor efficiency. The infinite cache CPI is a gauge that gives the relative efficiency of a processor on a specific workload. Its value depends on the workload as well as on the processor itself. The main advantage of using 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 the power of the system for that workload.

5.2 Memory

When considering the performance of a system, the processor and the memory should not be considered separate devices, but as elements that closely interact.

Here are some hardware system parameters that can influence performance:

- Frequency ratio between the processor speed and the L2 cache (for instance 2:1 for a processor speed of 100 MHz and an L2 bus speed of 50 MHz)
- L1 cache size, line size, associativity
- L2 cache size, line size, latency, intervention
- Cache replacement algorithms
- Bus speed, width and protocol
- Frequency ratio between the processor speed and the memory bus speed
- Memory subsystem latency

5.2.1 Cache memory

After a program is loaded into main memory, the RS/6000 main processor requests the very first instruction in the program. In compliance with the request, the first instruction, along with the next several instructions, is retrieved from the main memory and loaded into the Instruction Cache Unit (ICU), which is used as a temporary holding area for programming instructions that are likely to be next in line to be executed. When the main processor requests the next instruction, it will first look in the ICU. Most of the time, the next instruction needed will already have been loaded into the ICU, eliminating the delay associated with getting the instruction from slower main memory. The primary functions of the ICU are:

- Fetch all instructions
- Execute branch and logic on Condition Registers instructions
- Dispatch instructions to the Fixed-Point Unit (FXU) and Floating-Point Unit (FPU)
- Process interrupts
- Maintain the architected Condition, Count, and Link registers

- Maintain interrupt control registers
- Provide engineering support processor (ESP) functions

The Data Cache Unit (DCU) is used to efficiently move the data on which the programming instructions are to operate between the RS/6000 main processor and main memory. The DCU operates much like the ICU, only the DCU provides temporary holding area for data needed during program execution rather than programming instructions. When a program instruction requires data on which to operate, the DCU is first checked to see if the needed data has already been loaded. If the data is not found (which is known as a cache miss) in the DCU, the needed data is automatically loaded from the slower main memory to the DCU.

In general, the DCU provides an instruction reload buffer for transferring instruction cache lines to the ICU, as well as store-back buffers for data cache operations. The DCU also provides an I/O cache for DMA operations.

The Level 2 (L2) cache is a combined instruction and data cache. The L2 cache, typically much larger than the instruction and data cache, reduces the effect of instruction or data cache misses by holding the majority of code that was initially loaded in the L2 cache from slower main memory. When instructions or data is not to be found in the ICU or DCU, the processor first checks the L2 cache. If the data is not found in the L2 cache, it is loaded from the slower main memory into the L2 cache.

The different types of cache provide several temporary holding areas for program data. The main processor fetches all instructions and data from the ICU and DCU, which together are sometimes referred to as the Level 1 (L1) cache. The ICU and DCU fetch data from L2 cache (when present, or from main memory when L2 cache is not available), and the L2 cache fetches data from main memory. L1 cache is faster than L2 but smaller in size. Main memory is much slower than the L2 cache.

5.2.2 Addressing considerations

Efficient use of caches is a major factor in achieving high processor performance, so software developers should use appropriate coding technique in order to achieve higher performance. This requires some knowledge of RS/6000 cache architectures.

5.2.2.1 Addressing

When the program requests that a register be loaded with the contents of a portion of memory, the memory location is specified by a 32-bit virtual address. The high-order 4 bits of this address are used to index into the bank

of 16 segment registers. The segment registers are maintained by the operating system, and at any given time contain the 24-bit segment IDs that have been assigned to the currently executing process. Those segment IDs are unique unless the process is sharing a segment with one or more other processes. The 24-bit segment ID from the selected segment register is combined with the 28 low-order bits of the data address to form the 52-bit virtual address of the data item to be loaded. Because the offset within the segment is 28 bits, each segment is 256 MB long. Figure 55 shows the successive transformation of a memory address.

.

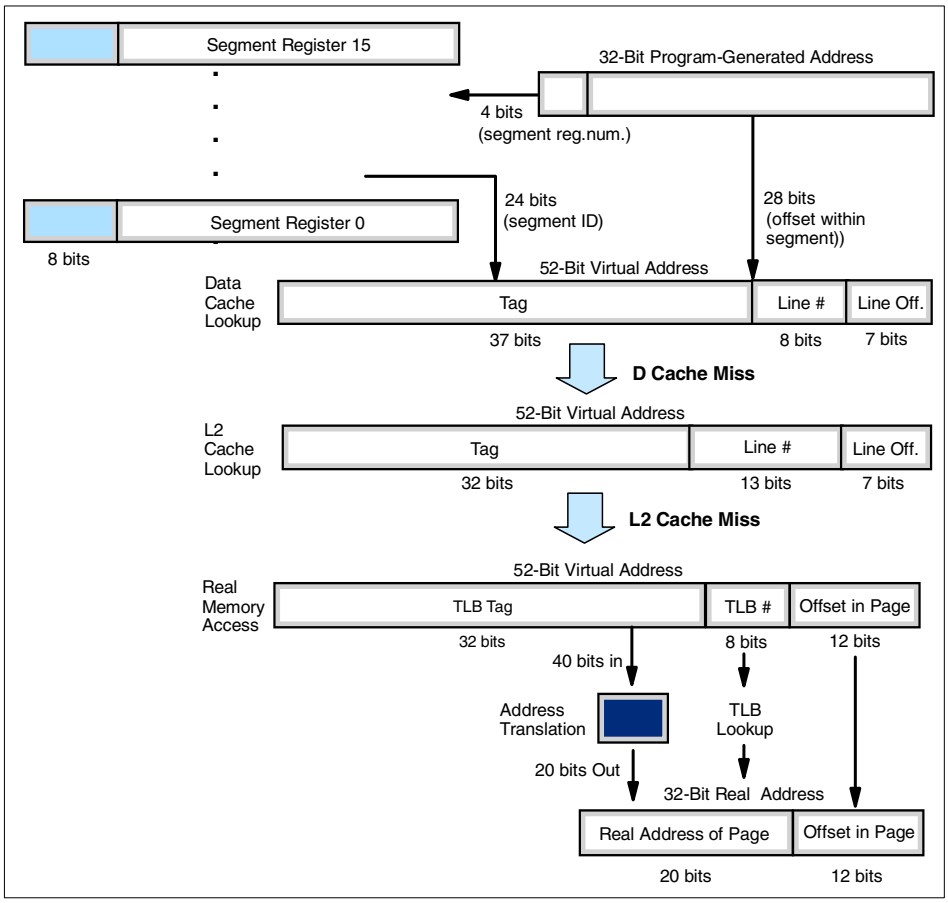


Figure 55. Successive transformations of a memory address

5.2.2.2 Cache lookup

The 52-bit virtual address is used for the data cache lookup, as shown in Figure 56 on page 137. Since the lines in the cache are 128 bytes long, the

low-order 7 bits of the address represent the offset within the cache line. The data cache contains 128 KB of space, and is four-way set associative. Thus each bank of the cache contains 256 128-byte lines (128KB/(128*4) = 256), and so the next higher-order 8 bits represent the line number (0-255). Each bank of the cache has a line with that number, and the four lines with the same number form the congruence class, that is, the four possible locations for the data being sought. This is a four-way set-associative cache. If the congruence class had two members, we would speak of the cache as two-way set-associative. If there were exactly one cache line corresponding to a given address, the cache would be direct-mapped.

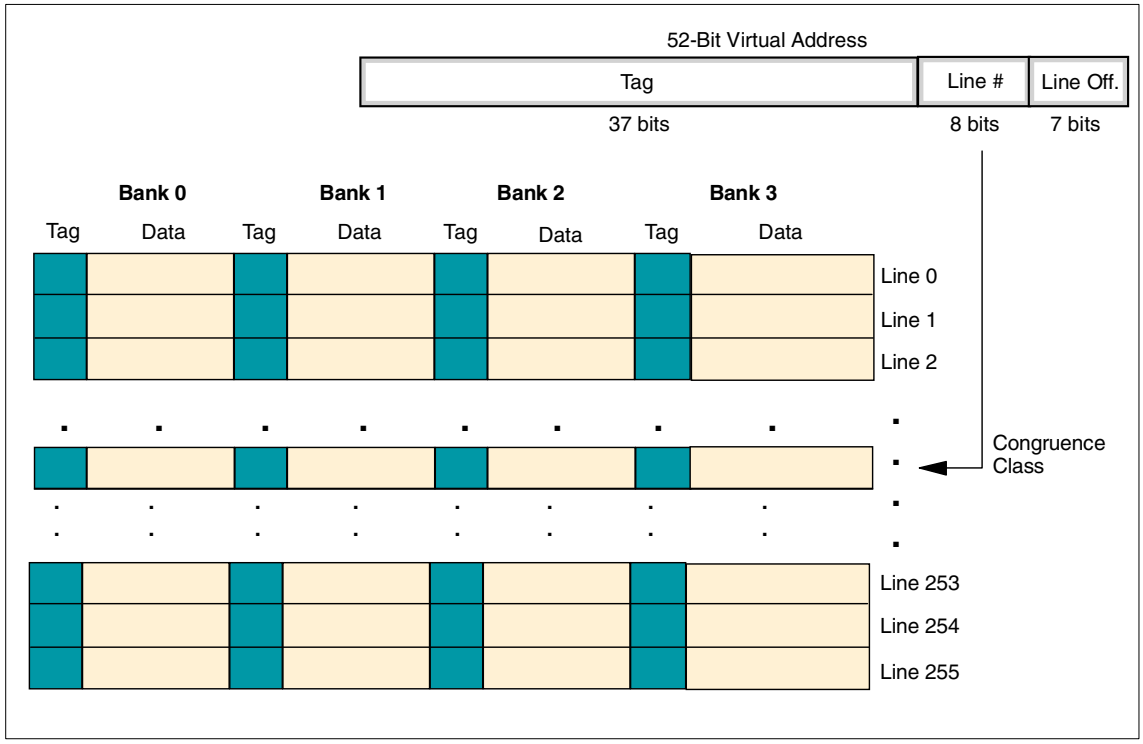


Figure 56. Cache lookup

Associated with each line of the cache is a 37-bit tag, which is the high-order part of the 52-bit address from which the cache line was originally loaded. If one of the tags of the four lines in the congruence set matches the high-order 37 bits of the 52-bit virtual address just generated, we have a cache hit. The data from the cache line is loaded into the register, and no access to the RAM (and so no real address) is required.

If none of the four tags in the congruence set matches the tag of the data to be loaded, there is a data cache miss. In this machine there is an L2 cache, so a cache lookup similar to the one in the data cache is performed. The primary difference between the data cache lookup and the L2 cache lookup is that the L2 is direct mapped. The lines are 128 bytes long, and the cache can hold 1MB. There are therefore 8192 lines. The low-order 7 bits of the 52-bit address are still the offset within the line. The next 13 bits constitute the cache line number. Each line is associated with a single 32-bit tag. If that tag matches the high-order 32 bits of the 52-bit address, there is an L2 cache hit. If not, the real address of the data must be determined and the data obtained from RAM.

Different implementations of the POWER architectures have different sizes and geometries of caches; some have no L2 cache, some have combined instruction and data caches, some have different line lengths. The precise size and position of the fields in the 52-bit address may differ, but the principles of cache lookup are the same.

5.2.2.3 TLB lookup

The data Translation Lookaside Buffer (TLB) is a cache of addresses. The TLB tag is the high-order 32 bits of the 52-bit virtual address. The next 8 bits of the 52-bit virtual address are the line number in the TLB, which has 512 entries and is two-way set-associative (so each bank has 256 entries). The low-order 12 bits of the 52-bit address are the offset within the 4096-byte page. The data portion of each TLB line is the 20 high-order bits of the 32-bit real address of the page (see the Figure 57 on page 139). If there is a TLB hit, the 20 high-order bits from the TLB entry are combined with the low-order 12 bits of offset within the page to form the 32-bit real address of the data.

If there is a TLB miss, the hardware determines the real address of the data using the page tables via an algorithm that is beyond the scope of this book. Obtaining the real address from the page tables takes several dozen processor cycles. When the 32-bit real address has been calculated, its 20-bit page-address portion is cached in the appropriate TLB entry, and the tag for that entry is updated appropriately.

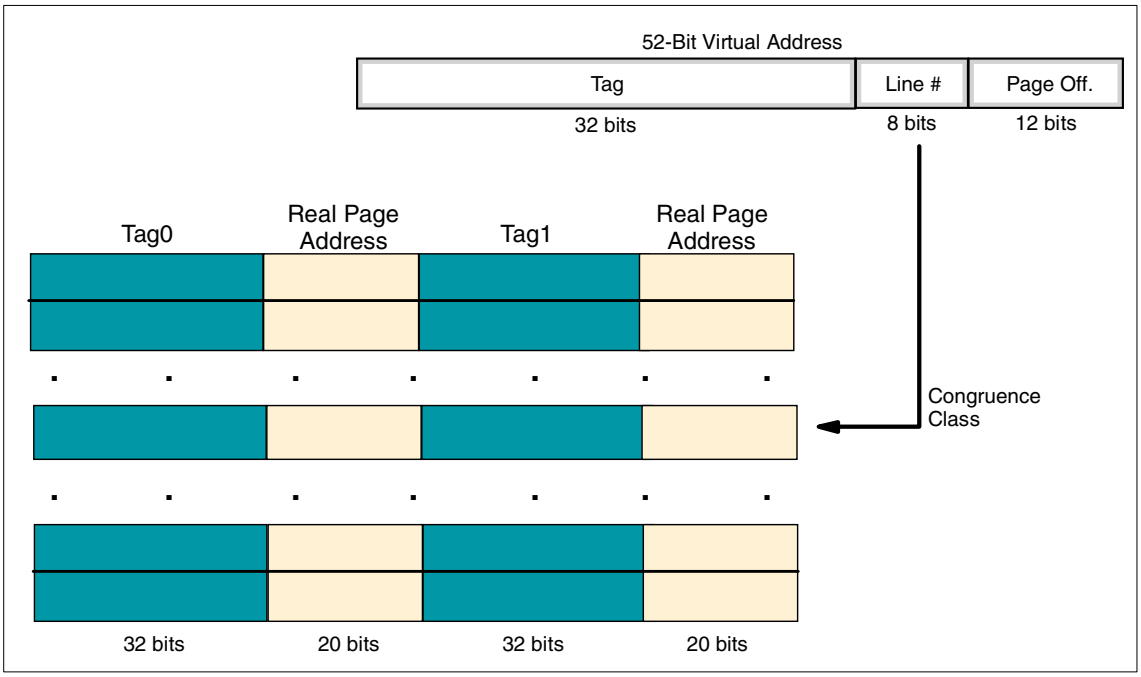


Figure 57. Data TLB lookup

5.2.2.4 RAM access

However derived, the 32-bit real address of the data is used to issue a request to RAM. Normally, there is a latency of at least eight processor cycles between the issuing of the RAM request and the return of the first 16-byte (128 bits, the width of the memory bus) section of data, which includes the data being loaded. At this point the processor can resume operation. The RAM access continues for a further seven processor cycles to load the appropriate data cache line with its full 128 bytes, 16 bytes at a time. Thus, a cache miss entails at least 16 processor cycles from beginning to end. The tag of the cache line is updated with the high-order 37 bits of the data address. The previous content of the cache line is lost.

5.2.2.5 Implications

Several kinds of pathological addressing patterns can cause incessant cache or TLB misses, greatly slowing the effective rate of execution. For example, if the program accesses an array larger than the cache with a stride of exactly 128 bytes, it will incur a cache miss for each access. If the program presents the processor with a series of requests for the same cache line number but in different pages, a series of congruence-set collisions will occur, resulting in

numerous cache misses even though the full capacity of the cache is not being used. The fact that the cache is four-way set-associative makes it unlikely that this will happen by chance, but a particularly unfortunate choice of offsets for data items could make a specific program particularly slow.

5.2.3 Memory cycles

In order to understand the importance of memory hierarchy's performance, here are the number of memory cycles required for a typical processor to access data from the different memory components.

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 about 100 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 seven to 10 cycles to access data from L2 in case of a cache miss in L1 and a cache hit in L2. It will take between 20 to 50 cycles to get data from memory in case of a cache miss in L2. And finally, if you need to access data from disk, it will take between 750000 to 1.5 million cycles. To highlight this point, if we assume that one cycle is one second, it will take 17 days, 8 hours and 40 minutes to access data from disk, whereas accessing data in the L1 cache would take one second.

Note

Detailed values are hardware dependent. These numbers should only be used as guidelines

Tuning small user programs often involves cache/TLB management. Tuning large application programs often involves page and I/O management. Figure 58 on page 141 shows the relative access times of different memory levels.

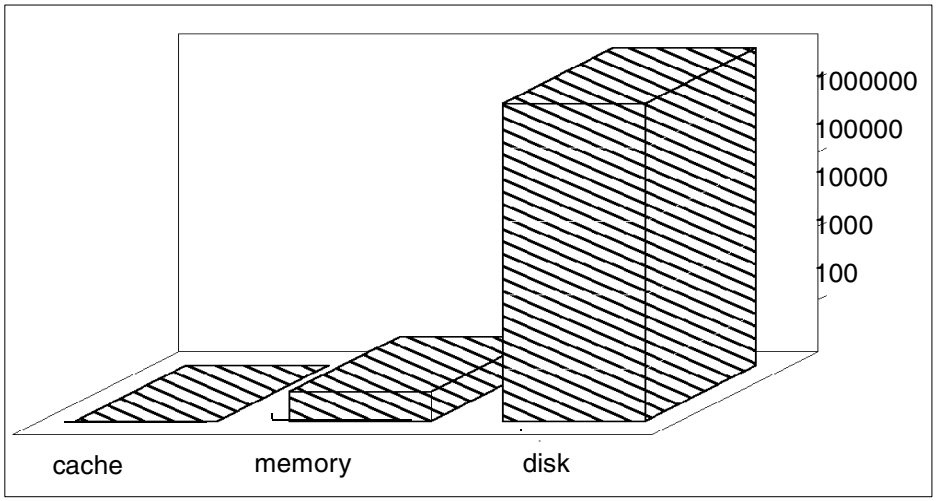


Figure 58. CPU Cycles per data access

Clearly, in terms of performance, accessing data from disk must be avoided at all costs. Thus, a commercial system should be designed to avoid misses to disks as much as possible. Because the memory latency is much better than the disk's latency, the memory itself should be used as a huge cache. Therefore, there is a need for high memory capacity.

5.2.4 Uniprocessor vs. symmetric multiprocessor memory cycles

The number of memory cycles needed to access data depends on whether the machine is a uniprocessor or a symmetric multiprocessor.

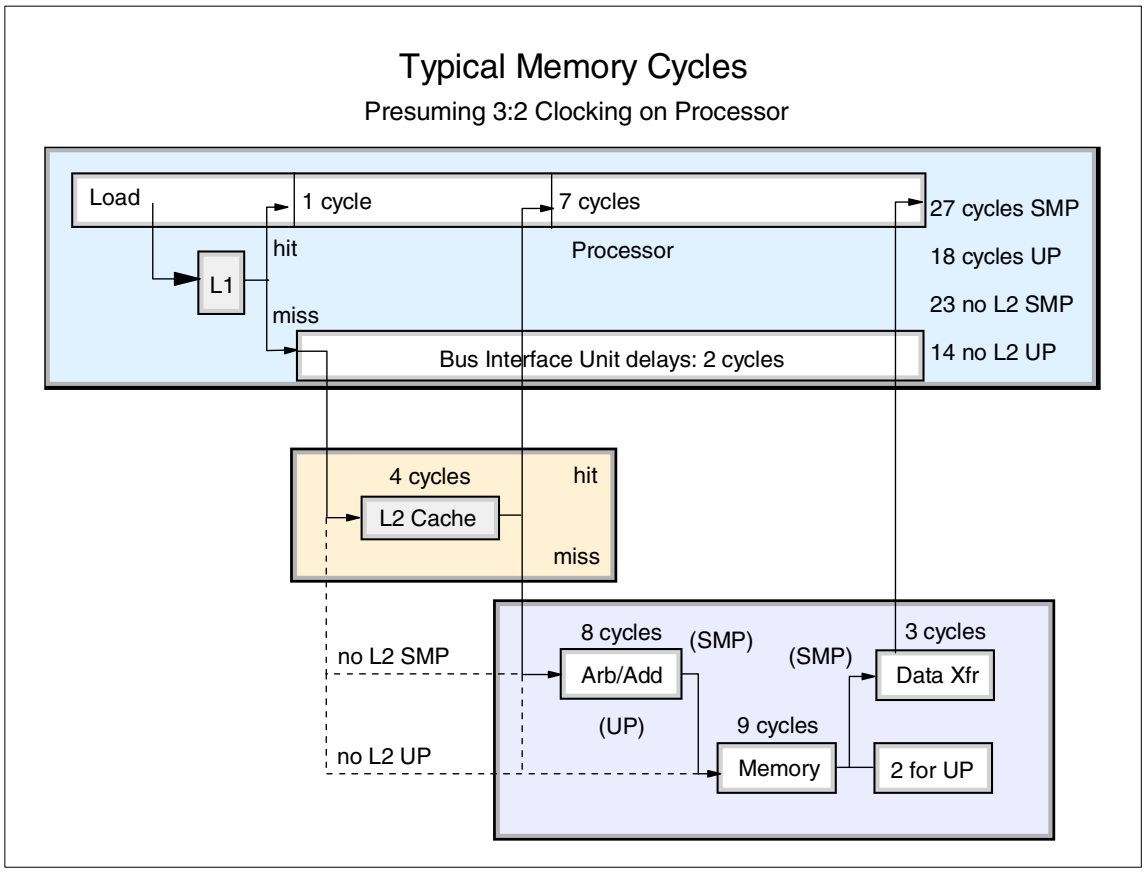


Figure 59. Typical memory cycles

In Figure 59 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 59 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. Phase locked loop (PLL) technology is used to match the bus and processor operating frequencies.

5.2.5 Miss rate penalty

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

Let us take a 100 MHz processor without an L2 cache, and also assume that the measured infinite cache CPI is 1.3 on that processor.

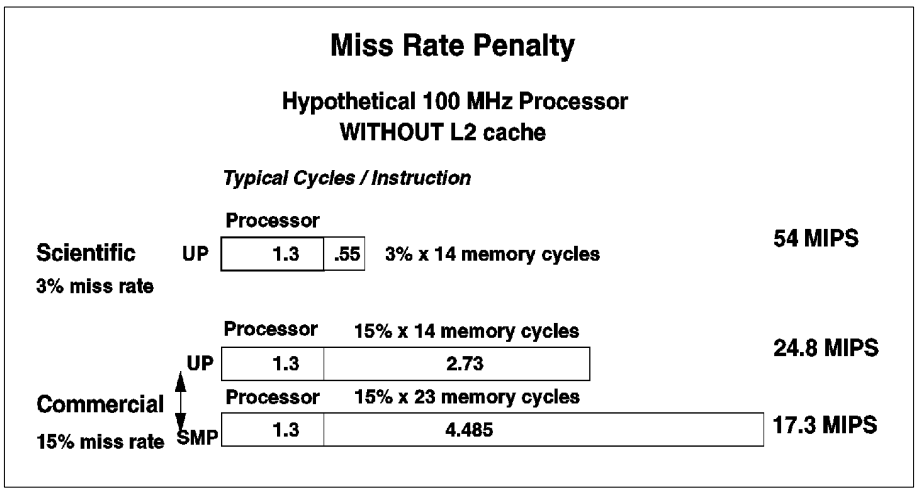


Figure 60. Miss rate penalty

In an engineering and scientific environment, the L1 miss rate is around 3 percent. This means that 3 percent of the time, you will need 14 cycles on a UP to access your data from memory. In this case, the total number of CPI will be:

```
1.3 + (0.03 x 14 x 1.3) = 1.3 + 0.55 = 1.85 CPI.
The "cost" of accessing the real memory is 0.55 CPI.
```

The first 1.3 value is the infinite cache CPI, which can be measured. The second 1.3 value in this calculation 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 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 100/1.85 = 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 CPI will be:

```
1.3 + (0.15 x 14 x 1.3) = 1.3 + 2.73 = 4.03 CPI.
This corresponds to 100/4.03 = 24.8 MIPS.
```

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 (24.8 MIPS vs. 54 MIPS).

For an SMP, the number of CPI will be:

```
1.3 + (0.15 x 23 x 1.3) = 1.3 + 4.485 = 5.785 CPI.
This corresponds to 100/5.78= 17.3 MIPS.
```

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 to the higher number of cycles required to access data from memory. Figure 60 on page 143 shows the miss rate penalty for UP and SMP systems that do not have an L2 cache.

5.2.6 Effect of L2 cache

Adding an L2 cache to a UP or to an SMP system can lower the time spent accessing memory data. This has a different effect depending on commercial or scientific environments. Typically, 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 cache.

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 as follows:

```
0.03 x 0.8 x 7 x 1.3 + 0.03 x 0.2 x 18 x 1.3 = 0.22 + 0.14 = 0.36 CPI
versus 0.55(accessing the memory without L2 cache).
The total CPI number is 1.3 + 0.36 = 1.66 CPI
```

Adding an L2 cache to a UP system saves only 0.19 CPI (0.55 CPI - 0.36 CPI) in a scientific environment. An L2 cache does not significantly improve the performance of the system in a scientific environment (100 / 1.66 = 60.2 MIPS instead of 54.0 MIPS). For example, the 3BT (POWER2) without L2 cache is rated at 3.14 SPECint95. With L2 cache of 1MB, it is rated at 3.21 SPECint95.

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 is:

```
0.15 x 0.8 x 7 x 1.3 + 0.15 x 0.2 x 27 x 1.3 = 1.09 + 1.05 = 2.14 CPI
versus 4.485 (accessing the memory without L2 cache).
The total CPI number is 1.3 + 2.14 = 3.44.
This corresponds to 100/3.44 = 29 MIPS.
```

In this case, the L2 cache has a great effect and can increase the performance of the system up to 67 percent (17.3 MIPS vs. 29 MIPS). Figure 61 on page 146 shows the L2 cache effect for a UP in a scientific environment and the L2 cache effect of an SMP in a commercial environment.

Conclusion
As the L1 miss rate increases, adding an L2 cache can greatly improve the performance of the system.

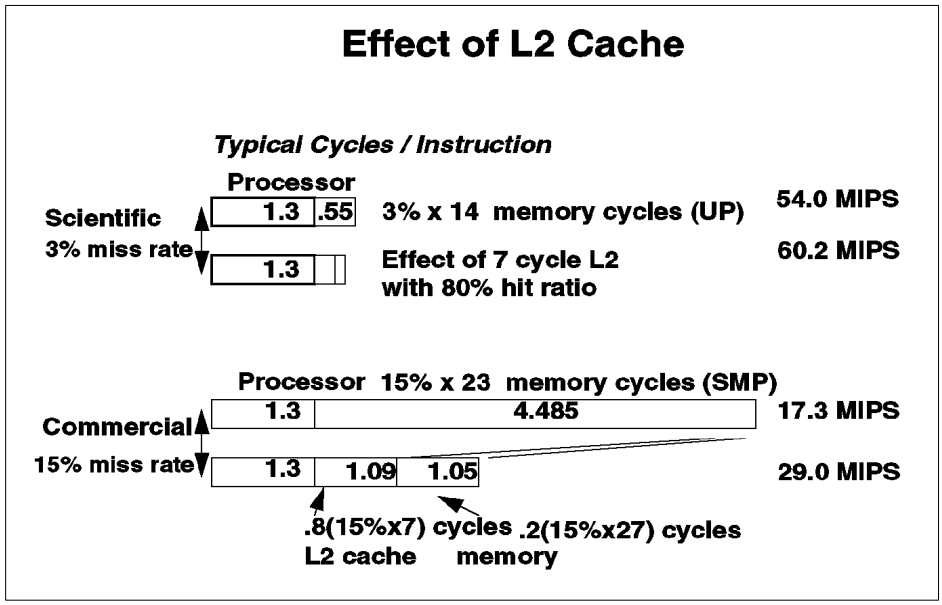


Figure 61. Effect of L2 Cache

5.2.7 Effect of processor speed

If the processor speed is doubled, the effect of L1 cache on CPI will be the same as calculated above (this is true if L1 is embedded in the processor chip itself). The L2 cache must run at a speed close to the processor speed to maintain good performance. This can be achieved because the technology used to build such L2 caches comes from the technology used to make processor memory. Of course, using such an L2 cache will increase the overall price of the system. In this case, typical L2 latency time should be between two and five cycles, depending on the processor bus frequency. For very high processor frequency, a 1:1 ratio between processor and L2 cache is hard to achieve. The typical ratio is 3:2 or 5:4.

Let us now assume that for a bus running at 66 MHz, it takes eight cycles to access memory. For a processor running at 100 MHz and working with a bus running at 66 MHz, the ratio will be 3:2 (1.5 * 66 = 100 MHz). Thus, accessing the memory will require 1.5 * 8 = 12 CPU cycles.

Scientific environment

For a processor running at about 200 MHz, this will give a 3:1 ratio with the memory bus (a bus running at 66 MHz). In that case, the main memory latency will be about 24 (3 * 8) processor cycles. Accessing the L2 cache

should still require about 5 processor cycles. In that case, for scientific environments, we will have:

```
0.03 x 0.8 x 5 x 1.3 + 0.03 x 0.2 x 24 x 1.3 = 0.156 + 0.187 = 0.343 CPI
The total CPI number is 1.3 + 0.343 = 1.643
This corresponds to 200/1.643 = 122 MIPS.
```

In Table 13 we summarize performance gains for increasing processor frequency based on the previous formula.

Table 13. Processor speed effects for the scientific environment

Processor speed	Memory Latency	MIPS	Delta Improvement	Ratio to 100 MHz
100 MHz	12 cycles	65 MIPS	N/A	N/A
200 MHz	24 cycles	122 MIPS	88%	1.9
300 MHz	36 cycles	173 MIPS	42%	2.7
400 MHz	48 cycles	219 MIPS	27%	3.4
500 MHz	60 cycles	260 MIPS	19%	4

For scientific environments using high-speed processors, even if the cache miss ratio is low, the high latency time of the main memory will greatly reduce the overall performance of the machine. The speed of the bus (66 MHz) will also reduce the performance of processors with high frequencies.

Commercial environment

For commercial environments with a processor running at 200 MHz, we will have:

```
0.15 x 0.8 x 5 x 1.3 + 0.15 x 0.2 x 24 x 1.3 = 0.78 + 0.936 = 1.716 CPI
The total CPI number is 1.3 + 1.716 = 3.016.
This corresponds to 200/3.016 = 66 MIPS.
```

In Table 14 we summarize performance gains for increasing processor frequency based on the previous formula.

Table 14. Processor speed effect for the commercial environment

Processor speed	Memory Latency	MIPS	Delta Improvement	Ratio to 100 MHz
100 MHz	12 cycles	39 MIPS	N/A	N/A
200 MHz	24 cycles	66 MIPS	69%	1.7
300 MHz	36 cycles	86 MIPS	30%	2.2
400 MHz	48 cycles	101 MIPS	17%	2.59
500 MHz	60 cycles	113 MIPS	12%	2.9

For commercial environments using high-speed processors and with a high cache miss rate, the high latency time of the main memory will greatly reduce the overall performance of the machine. The speed of the bus (66 MHz) will also reduce the performance with high processor frequency.

Conclusion

The higher the miss rate, the smaller the benefit from increasing the processor speed. When reaching high processor frequency, continuing to increase the processor speed will lead to little performance gain.

5.3 Storage

The evolution of the RS/6000 system architecture has included many changes in power management, processors, and memory. Also, the I/O requirements have increased, and faster response times are required. In order to reach the response times and the throughput levels required today, some changes have been made to adapter and disk technologies.

The traditional SCSI and SCSI-2 interfaces, used in all the old RS/6000 system models, have begun to reach their limits. The disks themselves have realized large-scale performance improvements, placing greater demands on the disk interface bus. Disk seek times have been reduced to less than 10 milliseconds (ms). The rotational speed has gone up to 10000 rpm.

Better response time is not the only need in a system. The demand for large storage solutions is growing faster than before. Applications today demand larger data objects, so complex databases and historical information issues require total disk space in Terabytes. Historical information maintenance, client/server computing, and mission-critical applications also demand higher

levels of availability, fault tolerance, higher levels of performance, and greater connectivity options.

5.3.1 Performance view

Fast processors running applications that cause large numbers of I/O disk drive accesses can become I/O bound and degrade system throughput. To sustain performance, computer systems have been using ever-larger memories. To use large amounts of memory for more file system buffering works well for systems that have locality of reference. But applications dominated by a high rate of random requests for short records or by a smaller number of requests for massive records must still face the underlying disk performance problem.

Amdahl's Law can be used to explain how the performance of an application can be affected by increasing only the processor speed or both processor and disk speeds as seen in Figure 62.

```
EAS = 1 / ((1 - FFM) + FFM / SFM)
```

Figure 62. Amdahl's Law

EAS = Effective application speed-up

FFM = Fraction of the work in the faster mode (using the fastest components such as processor and memory)

SFM = Speed-up of the faster mode

The law predicts that, if we have an application running 80 percent of its time on the processor (FFM=0.8), and 20 percent in I/O operations, then increasing the processor speed by a factor of 2 (SFM=2) results in an increment of 1.67 in the application speed (EAS = [1/((1-0.8)+(0.8/2)]). When the processor is 16 times faster, the same application will be only 4 times faster.

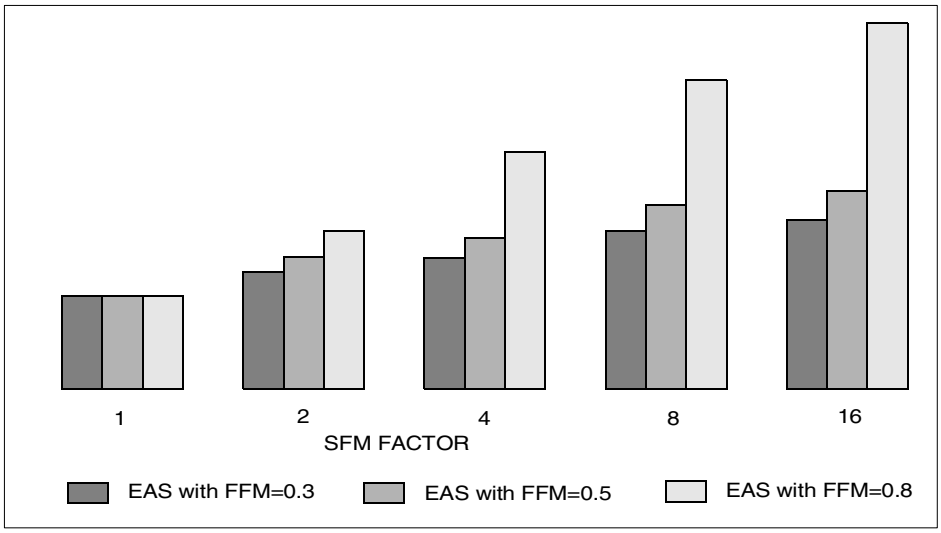


Figure 63. Speed-Up comparison

Note that for applications with high I/O, the most effective speed-up tends to be 1/(1-FFM). For this example, we theoretically could increase the processor speed by a factor of 1000, but the acceleration of the application will be only about 5. Figure 63 shows the speed-up comparison.

5.3.2 Levels of storage

When we consider storage as the slower mode that an application can work in, several components of the system architecture are involved; I/O subsystem, bus controller (PCI, ISA), disk subsystem adapters (SCSI, SSA, Serial Link), disk array controllers, disk drive electronics and, finally, disk drive mechanics. The lower the level of the component, the slower speed it has. Figure 64 on page 151 shows the levels of storage

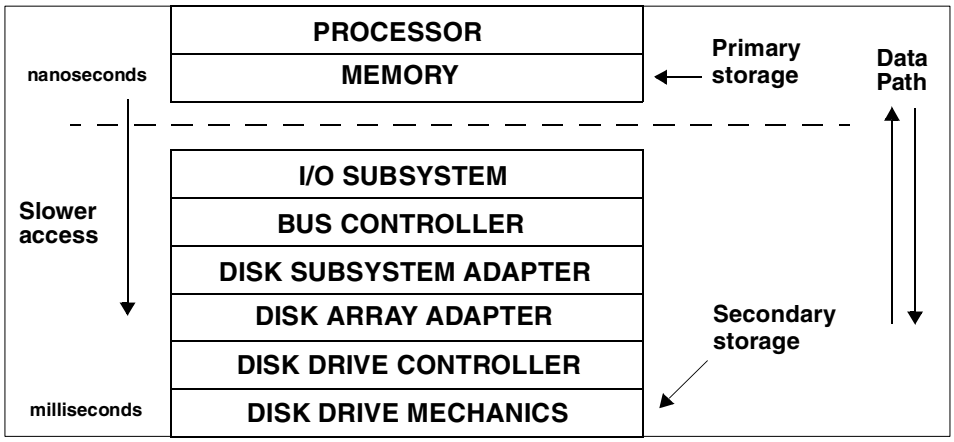


Figure 64. Levels of storage

5.3.3 How an I/O request is processed

When a process needs to reach data from a data file, it uses a system call to the operating system. The operating system puts the request of the process for I/O in a queue managed by a specific driver for the device. All requests are organized in the queue using a seek optimization algorithm, so the disk heads can easily find all the data requested. The device driver translates the request for data into commands for the disk drive. These commands will manage the disk mechanics to find the data across the disk surface.

Once the request is being processed by the disk drive and the head is moved to the correct track, the data is transferred to the drive controller. The device driver takes the data from the controller when a hardware interrupt notifies the operating system of the result of the operation. The data is finally transferred to process memory (via DMA), and the process is awakened and placed on the run queue to wait for the processor.

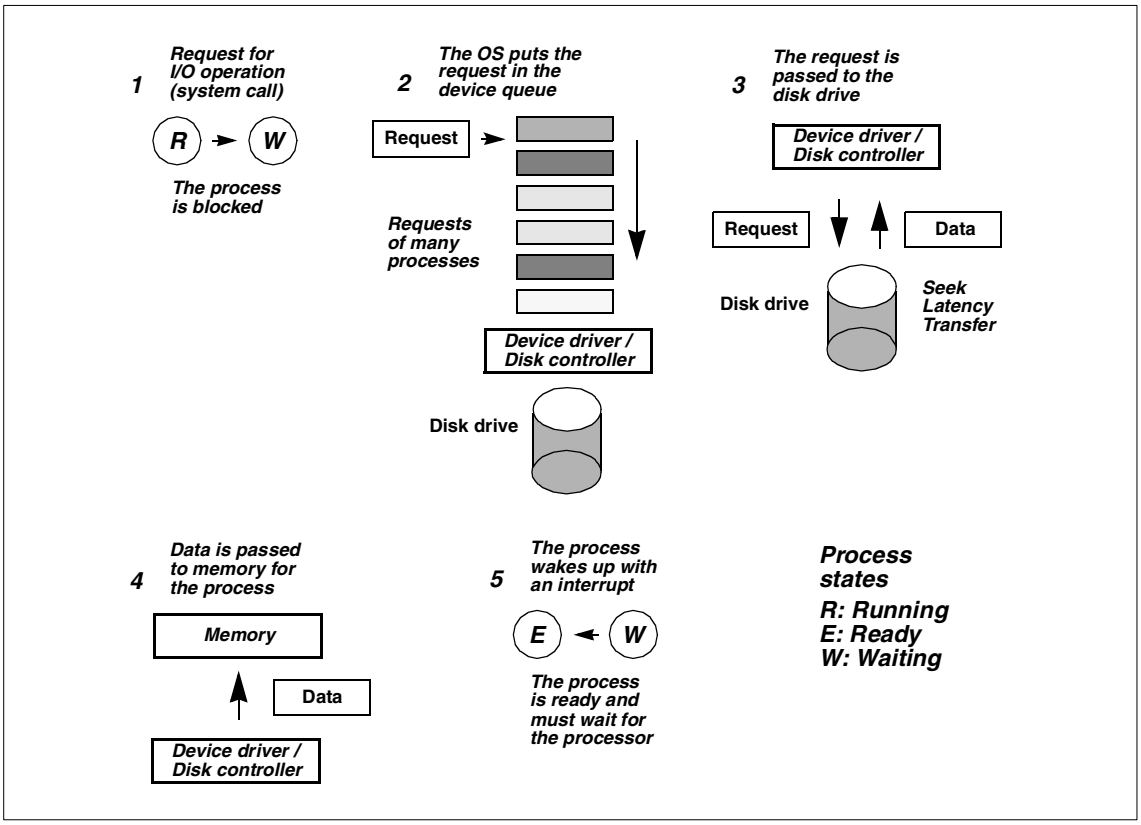


Figure 65. The I/O request path

The workload for a single process is not meaningful when analyzing an entire system except when the process is the only active one. Secondary storage of a system is accessed many times by different applications, and some requests can be concurrent. The read or write commands to be processed by a disk drive could be either a few or many thousand bytes each. There is no specific pattern for the order in the queue of the device. Figure 65 shows the I/O request path.

Random access: The workload is composed of many accesses to many files simultaneously in a system, and most of the accesses are only a few bytes. If the data is spread over the disk in a random pattern, then most of the accesses would be over different tracks of the disk. This implies that the disk is spending most of its time in setting up (seeking) the next transfer.

Sequential access: A workload is considered to have sequential access if most of the requests to the disk drive are for large and continuous byte

streams for a few files, and if the data is allocated sequentially in continuous portions of the disk. Most of the reads or writes would be over continuous tracks, and the disk would not spend much time setting up for the next transfer.

Because the I/O requests of the processes are translated in system calls, many I/O accesses will demand significant processor time. The more commands executed per time interval, the greater the processor utilization. It must be noted that intensive I/O does affect computation-intensive applications when both I/O and application processes are running in parallel. For a heavily loaded system with a high number of physical I/Os per second, this might result in 15 to 25 percent of the processor time needed for servicing the I/O requests alone.

In many environments, the overall performance of an application is bound by the speed at which data can be accessed from secondary storage. A good predictor of the overall performance of a fixed disk is the rate at which it can read or write these disk blocks.

For most applications following a random-access pattern, the access time of a disk is the best way to measure its performance, especially when the access is for small data blocks.

For applications following a sequential access pattern, a good indicator of disk performance is the throughput of the disk adapter at peak rates.

The degree to which a disk may be utilized depends on the way it is used. If the disk drive is dedicated to a single process, even a 100 percent device busy state is generally no reason for concern. In fact, it might be desirable. This situation is quite different in the case where multiple users are working with the disk at the same time. Here, a 40 percent device busy state should be investigated.

5.3.4 How a disk works

Disk drives only understand special commands given by their controllers. So a SCSI disk drive only works with SCSI commands and an SSA disk with SSA commands. The manner in which the hardware of the disk drives works is the same for all of them. Each request for reads or writes has a virtual address of the data on the disk (sector address), an action (read or write), the size of the string to be processed, and the string to be written (if any).

All the actions for reading or writing bytes to the disk drive are coordinated by the kernel and are transparent to the processes. They only attempt to

interpret the result and data of the `read()`, `write()`, `seek()`, `open()` or other functions when the kernel returns them.

System calls for I/O generate four main tasks:

- To evaluate where the bytes are or will be located on the disks (disk drive, platter, head, track, sector)
- To create a command for the disk drive hardware
- To activate the disk arm to position the correct head over the track to be read or written
- To read/write and transfer the bytes to/from memory

The last two tasks are handled by the disk drive (except the transfer part).

The access time of a disk consists of three components:

- **Seek time**: A seek is the physical movement of the head at the end of the disk arm from one track to another. The time for a seek is the time for the disk arm to accelerate, travel over the tracks to be skipped, decelerate, and finally settle down and wait for the vibrations to stop while hovering over the target track. The total time the seeks take is variable. The average seek time is used to measure the general disk capabilities, and it is generally lower than 15 ms.
- **Latency**: The rotational latency is the time that the disk arm has to wait while the disk is rotating underneath until the target sector approaches. Rotational latency is, for all practical purposes except sequential reading, a random function with values uniformly between zero and the time required for a full revolution of the disk (less than 10 ms). The average rotational latency is taken as the time of a half revolution, and it is generally lower than 5 ms. For the Ultra160 SCSI disk drives used today, this time is 4.17 ms.
- **Transfer**:The data transfer time is determined by the time it takes for the requested data block to move through the read/write arm. It is linear with respect to the block size. For a 4 KB page transfer, this time is typically near 1 ms.

The average disk access time is the sum of the averages for seek time and rotational latency plus the data transfer time (normally given for a 512 bytes block). It is 6.8 ms for Ultra160 SCSI disk drives. The average disk access time generally overestimates the time necessary to access a disk. For random access, seeks tend to be shorter than the average. Typical disk access time is 70 percent of the average. Figure 66 on page 155 shows the disk times.

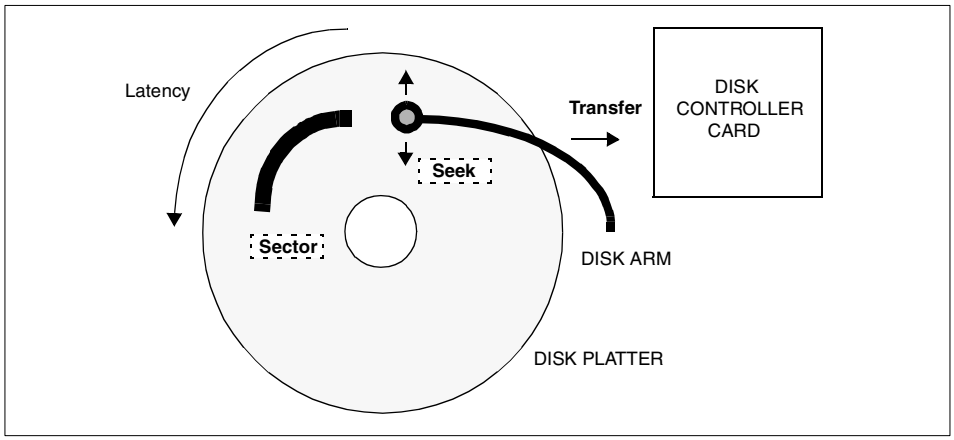


Figure 66. Disk times

5.3.5 SCSI technology

With the dramatic increase in the amount of data that today's companies must process and store, the need for affordable, reliable, and high-performance interfaces is greater than ever. To keep up with enormous growth, companies are looking for technologies that can help them expand quickly and easily by building on their existing infrastructure. One of the best interface technologies to address these requirements is SCSI. For several years, all types of companies have been leveraging SCSI technology to handle a wide range of data transfer and storage needs. The unprecedented industry-wide acceptance of SCSI technology can be largely attributed to its impressive history of providing outstanding performance gains without sacrificing compatibility from one version to the next.

5.3.5.1 SCSI and SCSI-2

SCSI involves parallel transmission of data across a parallel set of wires. These wires carry data and clock signals, and the devices are attached to the set of wires (SCSI cable) forming a bus. The number of wires in a SCSI cable is 50 for SCSI and SCSI-2 and 68 for SCSI-2 F/W. The first SCSI, also named 8-bit SCSI, is limited to handling eight total addresses and can transmit data at 4.5 MB/s. SCSI-2 improves the clock speed over SCSI and can transmit data over the bus up to 10 MB/s. SCSI-2 F/W can handle up to 16 total addresses in a 16-bit bus and can transmit up to 20 MB/s.

Note

As a rule of thumb one has to keep in mind that the *sustained* data transfer rate for SCSI devices, and therefore the performance you might expect during normal operation, is only about 30 - 40 percent of the maximum transmission rate.

Every SCSI device (disk, CD-ROM, tape) is connected to the bus using the same wires as the other SCSI devices. Therefore, data can travel in only one direction at a time. A SCSI device can process multiple commands simultaneously. An arbitration level is required in the controller to prevent multiple devices from using the bus at a time. As the SCSI adapter needs one address to access the bus, the maximum number of the devices it can support is seven for SCSI and SCSI-2, 15 for SCSI-2 F/W in a single bus or 30 in two buses (AIX V3.2.5 allows up to 14 in two buses).

The SCSI definition uses offline seeks. This means several disks on the SCSI bus can be in their seek or rotational latency phases concurrently without blocking the SCSI controller. Although the SCSI drives can seek offline, they must transfer data through the SCSI bus, so only one SCSI device can be transferring data at a time. Thus, the real throughput is limited by the disk drive and adapter throughputs. Because data is typically prestaged in cache on the disk interface card the disk drive throughput is dependant on how fast data can be transferred in and out of this cache.

5.3.5.2 Ultra SCSI

Ultra SCSI is a clock doubled version of the current 8-bit SCSI-2 and 16-bit SCSI-2 F/W standards. The 8-bit Ultra, sometimes called Fast 20, has a maximum data transfer rate of 20 MB/s, similar to the speed of the Fast and Wide interface. The 16-bit Wide Ultra has a top speed of 40 MB/s.

Ultra SCSI is the parallel implementation of the next generation called the SCSI-3 interface family. While those in the SCSI-2 line up are all parallel, SCSI-3 will support serial standards, including SSA. A current definition of SCSI-3 in parallel mode is Ultra SCSI. SCSI-3 will be defined by the American National Standards Institute (ANSI) standards committee.

5.3.5.3 Ultra2 SCSI

Ultra2 SCSI or Low Voltage Differential (LVD) is a highly compatible computer disk drive interface that is faster and more reliable than previous SCSI standards. This upgraded SCSI interface helps meet the growing need for faster data rates, and is available in 68- and 80-pin configurations.

Servers, workstations, RAID subsystems, and Internet technologies as well as CAD/CAM, multimedia, video, digital broadcasting, and groupware applications all require a more advanced interface to handle increasing data-transfer needs. The Ultra2 SCSI (LVD) interface satisfies these needs by increasing bus bandwidths, allowing greater configuration flexibility, backward compatibility, and faster transfer rates.

Ultra2 SCSI increases the bus data rate to 80 MB/s in 16-bit SCSI mode while lowering overall power consuption.This allows the bus to accommodate more devices. Ultra SCSI can connect up to 15 peripherals without creating a bottleneck. Therefore, a user can not only connect more devices, but the speed and reliability of the information also increases. Using the rule of thumb, the sustained data transfer rate for Ultra2 SCSI is 24 - 32 MB/s.

5.3.5.4 Ultra3 SCSI (Ultra 160)

Ultra3 SCSI refers to products that incorporate any or all of the following features of the SCSI SPI-3 standards:

- Double transition clocking
- Domain validation
- Cyclic redundancy check (CRC)
- Packetization
- Quick arbitration select (QAS)

A specific feature set of Ultra3 SCSI known throughout the industry as Ultra160 SCSI is currently taking SCSI to new levels of performance. Named for its superior 160 MB/s data transfer speed, the latest generation of SCSI technology incorporates the three management features of Ultra3 SCSI that specifically affect data transfer; CRC, domain validation, and double transition clocking. These new capabilities provide a cost-effective way to dramatically boost both device performance and reliability. Because Ultra160 SCSI is compatible with Ultra2 SCSI devices, it helps protect existing investments and ensures a smoother transition. The cables, connectors, and terminators are the same for both Ultra160 SCSI and Ultra2 SCSI. In fact, Ultra160 SCSI host controllers can support Ultra2 SCSI devices, which enables the mixing of Ultra160 and Ultra2 SCSI devices on the same bus. When Ultra160 SCSI and Ultra2 SCSI devices are mixed, each device can operate at its full rated speed. Using the rule of thumb, Ultra160 SCSI can be expected to have a sustained data transfer rate of 48 - 64 MB/s, while Ultra2 SCSI can be expected to have a sustained data transfer rate of 24 - 32 MB/s.

A wide variety of benefits are gained by incorporating a specific combination of Ultra3 SCSI's optimized capabilities. Ultra160 SCSI has established an effective industry standard for performance and device compatibility. IBM is building on its previous commitment to SCSI technology by incorporating Ultra160 SCSI into its products, such as the IBM Ultrastar hard disk drive family.

Before the Ultra3 SCSI specification, data was transferred over the SCSI bus by using single transition clocking, a design that limited the maximum data transfer rate to just half of the clock speed. Double transition clocking enables Ultra160 SCSI to achieve a superior data transfer rate, a critical factor in increasing overall drive performance. Data is transferred over the SCSI bus by a double transition clock that increases the speed of the data lines. As a result, the *maximum* transfer rate is twice that of Ultra2 SCSI; 160 MB/s compared to 80 MB/s. This design results in better performance, especially in environments that use extended transfer lengths or have many devices on a single bus. As an added bonus, the greater data handling capacity can also help increase reliability.

With previous SCSI versions, the host controller determined what data transfer rate was used for each connected device. Unfortunately, there was no guarantee that the connection could actually support the negotiated data rate. With domain validation, after a transfer speed is negotiated, it is checked at the actual negotiated rate. If errors are detected, the rate is decreased until the connection is free of errors. In this way, domain validation manages the connection to help ensure drive availability, reduce installation problems, and minimize costly service calls, all of which helps lower the total cost of ownership.

To improve reliability, Ultra160 SCSI leverages CRC, a proven international standard incorporated into technologies such as Fast Ethernet, FDDI, and Fibre Channel. While previous versions of SCSI used parity checking to detect transmission errors, CRC uses an additional error detection capability that is far superior for high-speed data transfer and hot-plugging situations.

Because CRC verifies that all transferred data (instead of just a single byte) is received correctly, it significantly improves data reliability. CRC also provides a solid foundation for increased data transfer rates in future versions of SCSI.

Ultra160 SCSI provides significant advantages over its predecessor Ultra2 SCSI in terms of performance:

Table 15. IBM SCSI adapters

	SCSI	Ultra SCSI Differential	Wide Ultra SCSI	Wide Ultra SCSI Differential	Ulta2 SCSI (LVD)	Wide Ultra2 SCSI (LVD)	Ultra 160
Maximum data transfer rate (MB/s)	20	20	40	40	40	80	160
Estimated sustained data transfer rate (MB/s)	6-8	6-8	12-16	12-16	12-16	24-32	48-64
Data Bus width (bits)	8	8	16	16	8	16	16
Max cable length	1.5-3	25	1.5-3	25	12	12	12
Max device support	8-4	8	8-4	16	8	16	16

The maximum theoredical data transfer rate in MB/sec. for today's commonly used interfaces. Maximum data rates are specified in the standards that relate to specific interfaces, and represent the speed of the data without considering overhead.

Ultra160+ SCSI

On the heels of Ultra160 SCSI comes Ultra160+ SCSI, which features all of the capabilities of Ultra160 SCSI and includes the two additional Ultra3 SCSI features; packetization and Quick Arbitration Select (QAS). These features are designed to enhance operating performance even further. Packetized SCSI is designed to reduce protocol overhead and provide scalable performance improvements, especially at data transfer speeds of 160 MB/s and higher. It provides faster transfer of command and status information and the ability to transfer multiple commands and multiple threads of data per connection cycle (commands, messages, and status are all transferred at the data rate speed). Enables transfers to be streamed within a single connection. Utilizes a packet structure similar to Fibre Channel.

Normal SCSI arbitration requires a certain amount of time during which one of the SCSI devices gains control of the SCSI bus. QAS is a new feature that provides faster arbitration than current SCSI devices by using a different protocol to determine which SCSI device gains control of the SCSI bus. This design reduces disconnect and reconnect time on the SCSI bus. As a result,

devices spend less time trying to establish communications, which enables much more efficient use of SCSI devices.

IBM is currently implementing Ultra160 SCSI technology in the IBM Ultrastar family of hard disk drives. Current drives, such as the Ultrastar 36LP, 36LZX, and 72ZX already operate at 160 MB/s, with some drives offering Ultra160+ transfer rates. Table 15 on page 159 shows the data transfer speed of IBM SCSI adapters.

5.3.6 Serial Storage Architecture (SSA)

Serial Storage Architecture (SSA) is an open-storage interface used to connect I/O devices and adapters to host systems. SSA was designed to be a high-performance, low-cost alternative to traditional SCSI based storage systems. SSA also directly addresses some of the performance and manageability limitations of the SCSI architecture. SSA is part of the ANSI SCSI-3 standard.

5.3.6.1 Technology overview

SSA subsystems are comprised of *loops* of adapters and disk devices. A theoretical maximum of 127 devices can be connected in a SSA loop, although current IBM SSA adapters limit this to a maximum of 48 devices per loop. Each SSA adapter has two loops, each with two ports or *nodes*. Data transfer is bi-directional in a SSA loop, with a maximum data transfer speed of 40 MB/s in each direction, for a total transfer speed of 80 MB/s per node or 160 MBs/ per loop.

In SSA terms, a node can be either an *initiator* or a *target*. As stated previously, each adapter contains two nodes or ports, and each SSA disk device also contains one node. The SSA adapter nodes are the initiator nodes responsible for issuing commands to the target nodes on the attached SSA disk devices.

SSA provides the following performance and manageability advantages:

- Dual connection paths to attached devices - If a break occurs in a loop, the data is automatically rerouted
- Simplified cabling when compared to SCSI - Cheaper, smaller cables and connectors, no need for separate terminators
- Faster interconnect technology
- Full-duplex, frame-multiplexed serial links
- Capable of transferring data at 80 MB/s per port, 160 MB/s per loop and adapter

- Hot pluggable cables and disks
- Supports large number of devices - Up to 127 per loop, although current IBM SSA adapters limit this to 96 disks per adapter
- Auto-configuration of attached devices and online discovery
- Increased distance between devices - Up to 25 meters with copper cables and 10 kilometers with optical fibre extenders.

5.3.6.2 SSA specific performance considerations

There are various performance factors specific to SSA implementations that must be considered when designing your disk subsystem. These include:

- The number of disks per SSA loop or adapter
- The distribution of the data among disks in a loop
- The position of the device in the loop

5.3.6.3 Number of disks per SSA loop or adapter

While the SSA adapter itself is capable of supporting a peak data transfer rate of 160 MB/sec., the host interface or bus usually limits the speed to a fraction of that supported by the adapter. The maximum sustained data transfer rate is approximately 90 MB/sec., a rate that is still much better than what SCSI or FC/AL are actually able to sustain.

The SSA architecture allows for a maximum of 48 disks per loop and 96 disks per adapter. The number of disks that can be effectively placed on a SSA loop or adapter is largely dependent on the I/O characteristics of the application that will be accessing the data. The exact number of disks that will provide the most optimal performance will obviously vary depending on the workload placed on the disk subsystem by the application. With that in mind, the following general rules of thumb apply:

- If the application primarily performs long sequential I/O operations, a maximum of 8 to 16 disks should be configured per SSA adapter. This configuration would provide sufficient bandwidth to saturate the host system bus in a PCI architecture.
- If the application performs a mixture of sequential and random I/O operations, then 16 to 32 disks per adapter would be sufficient.
- If the application is characterized by many short transfers with random seeks, the chances of any one disk saturating the available bandwidth of the adapter or bus is fairly low; therefore, more disks should be added per loop/adapter. In this instance, two loops of 24 to 48 disks per loop should

provide adequate performance while still staying within the limits of the adapter or host system bus.

5.3.7 RAID levels overview and performance considerations

Redundant Array of Independent Disks (RAID) is a term used to describe the technique of improving data availability through the use of *arrays* of disks and various data striping methodologies. A disk array is a group of physical disk drives used simultaneously to achieve higher data transfer and I/O rates than those available through the use of a single drive. IBM was responsible for much of the initial research and development into the use of RAID, with the first patent being issued in 1978.

The initial focus of RAID research was to improve performance while also reducing the overall cost per unit of storage. Further research emphasized the improved data reliability and fault tolerance that characterizes modern RAID systems.

The alternative to RAID disks is a set of disks connected to the system in which logical volumes are placed, and any one logical volume is entirely on one disk. This is often called JBOD, meaning Just a Bunch of Disks.

Within the RAID architecture, there are varying degrees of data reliability and performance, known as RAID Levels. Depending on the RAID Level, data can be either mirrored or striped. Data redundancy is provided through data mirroring, which maintains two copies of the data on separate physical disks. Data striping involves distributing the data among several disks by splitting it into multiple sequential data blocks and writing them to each of the drives in the array in parallel. In addition, most of the RAID Levels create parity information that can be used to reconstruct data on a particular drive in the event of a failure. The standard RAID specification provides for Levels 0-6,although some vendor specific implementations exist, such as EMC's RAID-S.

5.3.7.1 RAID level 0

RAID 0, referred to as data striping, differs from the other RAID implementations in that it does not offer any form of data redundancy. RAID 0 splits data into chunks and then writes or *stripes* the data sequentially across all of the disks in the array. This implementation offers the following performance advantages:

- Parallel I/O streams to multiple drives allow for higher data transfer rates for sequential read/write operations

- Increased throughput of random disk accesses due to the distribution of data onto multiple disks

The primary disadvantage of a RAID 0 configuration is that, should a single disk in the array fail, all of the data in the array will become unusable because it cannot be reconstructed from the remaining drives. RAID 0 should, therefore, be used for applications that require a high level of performance but do not have very stringent data availability requirements.

5.3.7.2 RAID level 1

RAID 1 uses data mirroring to achieve a high level of redundancy. In a RAID 1 configuration, two copies of the data are kept on separate disks, each mirroring the other. In the event of a single disk failure, all read/write operations will be redirected to the mirrored copy of the data. RAID 1 configurations are the most expensive of any of the other solutions because twice as many disks are required.

Read performance of a RAID 1 configuration implemented via the AIX LVM is enhanced due to the fact that, should the primary copy be busy, read requests can be directed to the mirror copy. Write performance can be slower than in non-RAID implementations depending on the write scheduling policy selected through the LVM; parallel or sequential.

Using the parallel scheduling policy, the writes to all copies of the data are initiated at the same time or in parallel. The write operation is considered complete when the copy that takes the longest time to update is finished. This is the fastest but less reliable option, as a failure to write to one of the copies may go undetected.

Under the sequential scheduling policy, the update of the mirror is not started until the write to the primary copy has successfully completed. This is the more reliable, but slower, of the two methods.

In addition, if Mirror Write Consistency (MWC) is turned on, it can have an impact on performance because potentially four disk write operations are performed for each LVM write operation - two writes to the MWC cache records in addition to the 2 data writes. The use of a Fast Write cache can mitigate the impact of the MWC cache record writes.

5.3.7.3 RAID level 2 and level 3

RAID Level 2 and Level 3 both break data into multiple chunks or segments and evenly distribute it across several physical disks. Striping in RAID 2 and RAID 3 occurs at the bit or multi-byte level. During a read operation, multiple simultaneous requests are sent to each disk causing all of the disk actuators-

the arm that holds the read/write head for the disk - to move in parallel. This limits the number of concurrent I/O operations in the array to one.

In order to provide data redundancy, RAID 2 and 3 configurations require parity information to be written for each write operation performed. While RAID 2 can distribute the parity information across multiple drives through the use of an encoding technique known as the Hamming method, RAID 3 uses only one drive for the parity. If one drive in the array fails, the data can still be accessed and updated using the parity information. However, the system will operate in a degraded mode until the drive is fixed due to the time required to dynamically reconstruct the data located on the failed drive using the parity information.

Note

AIX does not directly support RAID Level 2 or RAID Level 3.

5.3.7.4 RAID level 4

RAID 4 is very similar to RAID 3 in that it stripes the data across multiple physical disks in the array. The primary difference is that the striping increment is a block or record instead of the bit or byte method used by RAID 3 configurations. By virtue of the larger data increment used to create the stripe, reads can be matched to the one physical disk that contains the requested data. This allows both simultaneous and independent reads to be processed.

As in RAID 3, a single parity disk is used for data redundancy. This can create a performance bottleneck for write operations, as requests to update the sole parity disk cannot be processed in parallel. Due to the performance problems associated with the single parity disk and RAID 4's similarity to RAID 5, RAID 4 is not a commonly used or recommended configuration.

Note

AIX LVM does not support RAID Level 4

5.3.7.5 RAID level 5

Instead of having a dedicated parity disk, RAID 5 interleaves both data and parity on all disks. In a 4+P RAID 5 array, five disks are used for data and parity combined. Four-fifths of the space on those disks is used for data and

one-fifth of the space is used for parity. In RAID 5, the disks can be accessed independently of one another, and it is possible to use a large stripe size, so most data transfers involve only one data disk. This enables multiple concurrent accesses, thereby, giving higher throughput for OLTP or other random workloads.

Due to the way in which parity data is typically generated for RAID 5, there is a write penalty associated with write access. Random write I/Os usually result in four actual I/O operations:

1. Read the old data

2. Read the old parity

3. Write the new data

4. Write the new parity

Some IBM RAID 5 implementations, such as the SSA RAID adapters, incorporate full or partial stripe write algorithms for sequential writes. This eliminates the need to read old data and old parity information, thereby reducing the number of I/O operations required. Also, the use of read/write cache in the adapter can mask the write penalty for many random write workloads either by getting a cache hit during the data read operation or by caching the writes. It is important to note that there is some form of write penalty associated with any redundant RAID architecture, including RAID 1. This is due to the fact that some amount of redundant information must be written in addition to the base data.

The IBM PCI SSA RAID adapters can be configured with an optional fast write cache, which dramatically reduces the impact of the write penalty associated with RAID 5 implementations.

5.3.7.6 RAID 0+1

RAID 0+1, also known as IBM RAID-1 Enhanced or RAID 10, is a combination of RAID 0 (data striping) and RAID 1 (data mirroring). RAID 0+1 provides the performance advantages of RAID 0 while maintaining the data availability of RAID 1. In RAID 0+1 configurations, both the data and its mirror are striped across all the disks in the array. The first stripe is the data stripe, and the second stripe is the mirror, with the mirror being placed on a different physical drive than the data. RAID 0+1 implementations provide excellent write performance, as they do not have to calculate or write parity data. RAID 0+1 can be implemented solely in software (AIX), solely in hardware, or in a combination of hardware and software. Which is the appropriate solution for

an implementation depends on overall requirements. RAID 0+1 has the same high cost characteristics as RAID 1.

5.3.7.7 Comparison of RAID levels

Table 16 summarizes the performance and availability characteristics of the different RAID Levels.

Table 16. RAID levels

RAID Level	Capacity	Data protection	Sequential	Random Read	Random Write	Cost
RAID 0	Very High	None	High	High	High	Low
RAID 1	Moderate	Very Good	Medium	Medium-High	Medium	High
RAID 3	High	Good	High	Low-Medium	Low-Medium	Medium
RAID 5	High	Good	High	High	Medium	Medium
RAID 0+1	High	Very Good	High	High	High	High

5.3.7.8 RAID 5 vs. AIX LVM mirroring

When deciding on a data protection strategy, most customers narrow their choices down to the two most widely implemented solutions; SSA RAID 5 or LVM mirroring. Both solutions provide a highly robust and reliable data protection mechanism with varying degrees of performance and cost.

When evaluating the performance of a RAID 5 configuration, two important factors should be considered; the number of disks in the array and the read/write ratio of the application that will be using the array. In RAID 5 configurations, transaction performance (especially for reads) is directly related to the number of disks used in the array. As the number of disks in the array increases, so do the number of I/O operations processed/second, up to the limits of the RAID adapter. This is due to the fact that read operations can be processed in parallel across the disks in the array.

The read/write ratio of the application is the other factor that should be considered when assessing the performance of a RAID 5 configuration. The write penalty associated with RAID 5 configurations that do not utilize a fast write cache can result in severe performance degradation for applications that are write intensive. If the application is characterized by a large number of read operations and relatively few writes, RAID 5 solutions without a fast write cache can provide roughly equivalent performance to their mirrored counterparts provided they use sufficiently large disk arrays.

For applications that are not particularly I/O intensive, RAID 5, without using the fast write cache, can provide reasonable performance at a significant cost savings when compared to mirrored solutions. As an example, in a RAID 5 environment, eight disks would be required for seven disks worth of storage; seven for the data and one for the distributed parity. By comparison, a mirrored environment would require 14 disks; seven for the data and seven for mirrors.

5.3.8 IBM Enterprise Storage Server (ESS)

The IBM Enterprise Storage Server attaches to most commonly used servers including RS/6000 and SP2 running AIX, many leading UNIX variants, S/390, Intel-based servers, and AS/400.The ESS provides:

- Extensive scalability with capacity configurations that range from 420 GB to 11.2 TB.
- Breakthrough performance, made possible by a powerful storage architecture and SSA disks.
- Comprehensive availability: There are no single points of failure or repair, and the ESS remote copy function improves availability by mirroring the data in a remote location, thereby providing availability should a disaster occur.
- Extensive connectivity: The ESS connects to all major channel types, including ESCON, Ultra SCSI, and fibre channel.
- Low total cost of ownership: The ESS provides the lowest total cost of ownership because of its low initial acquisition cost, field upgrade capability, and the ESS Specialist storage management software.

5.3.8.1 ESS performance

The IBM Enterprise Storage Server uses fast, intelligent storage management to keep pace with escalating demands for data. It is designed with two powerful four-way RISC SMP processors, a large cache, and serial disk attachment. ESS offers outstanding performance and high bandwidth for open multiplatform environments. Performance is enhanced by an intelligent server, which provides:

- Optimized caching algorithms. Based on statistics, the optimal caching algorithm is selected from three possibilities; record stage for highly random workloads, and end of track stage and full track stage for sequential workloads.
- A special UNIX kernel resides in each cluster, which is optimized for managing I/O in the subsystem.

The outstanding performance offered by the ESS makes it ideal for DB2, IMS, SAP, Oracle, and other enterprise applications.

5.3.9 Logical Volume Manager (LVM) concepts

Many modern UNIX operating systems implement the concept of a Logical Volume Manager (LVM) that can be used to manage the distribution of data on physical disk devices. AIX LVM is a set of operating system commands, library subroutines, and other tools used to control physical disk resources by providing a simplified logical view of the available storage space. Unlike some competitors' LVM offerings, the AIX LVM is an integral part of the base AIX operating system provided at no additional cost.

Within the LVM, each disk or Physical Volume (PV) belongs to a *Volume Group* (VG). A volume group is a collection of 1 to 32 physical volumes (1 to 128 in the case of a big volume group), which can vary in capacity and performance. A physical volume can belong to only one volume group at a time. A maximum of 255 volume groups can be defined per system.

When a volume group is created, the physical volumes within the volume group are partitioned into contiguous, equal-sized units of disk space known as *Physical Partitions* (PP). Physical partitions are the smallest unit of allocatable storage space in a volume group. The physical partition size is determined at volume group creation, and all physical volumes that are placed in the volume group inherit this size. The physical partition size can range from 1 to 1024 MB, but must be a power of 2. If not specified, the default physical partition size in AIX 4.3 is 4 MB for disks up to 4 GB, but must be larger for disks greater than 4 GB because the LVM, by default, will only track up to 1016 physical partitions per disk (unless you use the `-t` option with `mkvg`, which, however, reduces the maximum number of physical volumes in the volume group). Table 17 lists typical physical partition sizes for typical physical disks.

Table 17. Physical disk size and partition1

Physical disk size	Physical partition size
2.2 GB	4 MB
4.5 GB	8 MB
9.1 GB	16 MB
18.2 GB	32 MB
36.4 GB	64 MB

After adding a physical disk to a volume group, in order to use the storage space you must create *logical volumes* (LV). Logical volumes define disk space allocation at the physical partition level. They can reside on many different, non-contiguous physical partitions, thereby allowing them to span physical disks. At the operating system level, logical volumes appear to applications as a single, contiguous disk.

When creating logical volumes, you must specify the number of logical partitions to allocate. Each logical partition maps to one, two, or three physical partitions depending on how many copies of the data you want to maintain. This allows for mirroring of the data either on the same physical disk or different disks in the same volume group.

5.3.9.1 Physical Partition striping versus LVM fine striping

Physical Partition striping refers to the technique of spreading the physical partitions of a logical volume across two or more physical disk drives. With PP striping, the size of the data stripe is the size of the physical partition, which is typically 4, 8, or 16 MB in size. This technique works well in environments that are characterized by a large amount of primarily random I/O operations, such as OLTP applications.

LVM striping, also known as fine striping, likewise attempts to distribute the I/O load by placing data stripes on multiple physical disks. However, LVM striping differs from PP striping in its use of a more granular or fine data stripe. With LVM striping, each logical partition of a logical volume is broken up into multiple stripe units and distributed among all of the physical devices that contain part of the logical volume. The stripe unit size must be a power of two in the range 4 KB to 128 KB, and is specified when the logical volume is created.

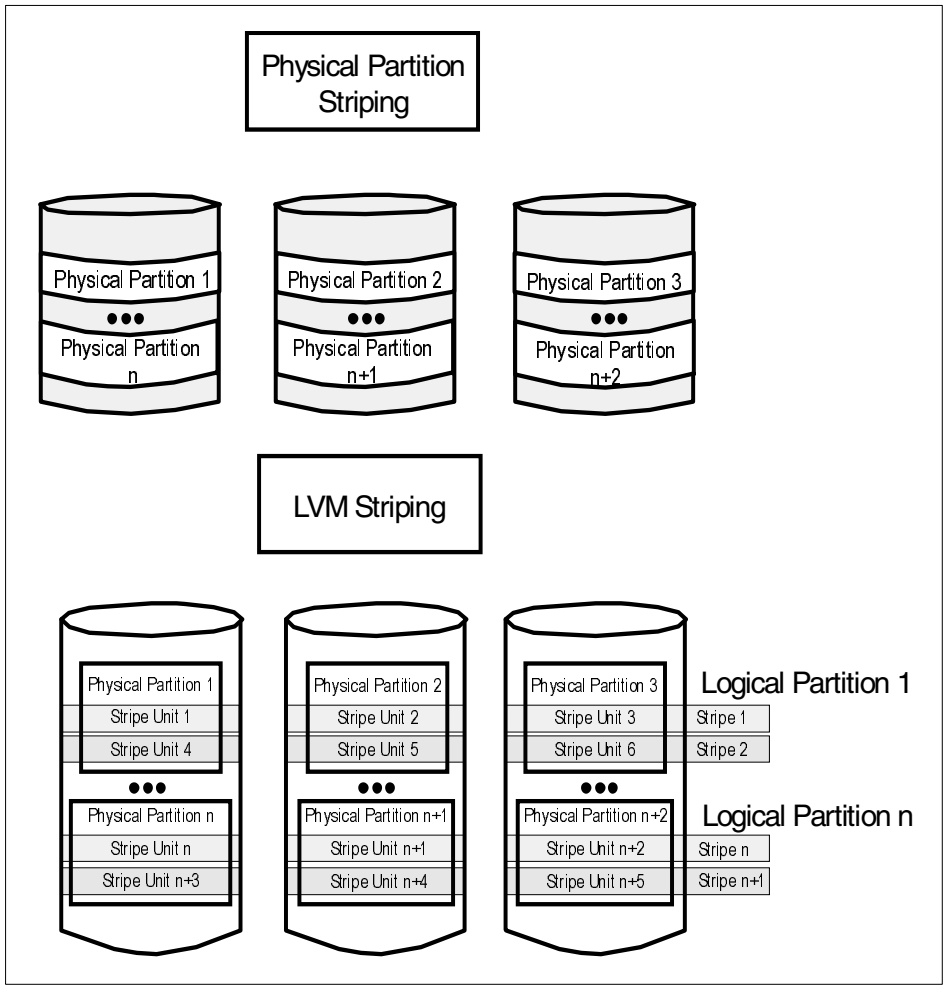


Figure 67. Physical Partition and LVM striping example

LVM striping works best in environments that perform many sequential read and write operations against large datafiles due to the performance benefits of *sequential read ahead*. Sequential read ahead occurs when either the application or the AIX Virtual Memory Manager (VMM) detects that the file is being accessed sequentially. In this case, additional disk reads are scheduled against the file in order to pre-fetch data into memory. This makes the data available to the program much faster than if it had to explicitly request the data as part of another I/O operation. Sequential read ahead is only available for files residing on JFS file systems, and has no meaning for raw devices

(raw logical volumes). Decision Support System (DSS) and batch workloads are good candidates for LVM striping.

Note

Prior to AIX version 4.3.3, logical volumes could not be mirrored and striped at the same time. Logical volume mirroring and striping combines the data availability of RAID 1 with the performance of RAID 0 entirely through software. Volume groups that contain striped and mirrored logical volumes cannot be imported into AIX Versions 4.3.2 and below.

5.3.9.2 Use of LVM policies

The AIX LVM provides a number of facilities or *policies* for managing both the performance and availability characteristics of logical volumes. The policies that have the greatest impact on performance are; *intra-disk allocation*, *inter-disk allocation*, *write scheduling,* and *write-verify* policies.

Intra-disk allocation policy

The intra-disk allocation policy determines the actual physical location of the physical partitions on disk. The disk is logically divided into the following five concentric areas: Outer edge, outer middle, center, inner middle, and inner edge.

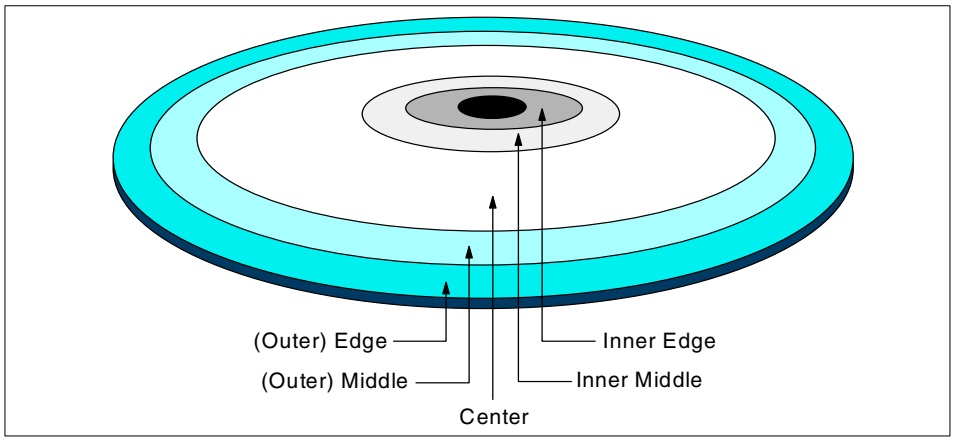


Figure 68. Physical Partition mapping

Due to the physical movement of the disk actuator, the outer and inner edges typically have the largest average seek times and are a poor choice for application data that is frequently accessed. The center region provides the fastest average seek times and is the best choice for applications that

generate a significant amount of I/O activity. The outer and inner middle regions provide better average seek times than the outer and inner edges but worse seek times than the center region.

As a general rule, when designing a logical volume strategy for performance, the most performance critical data should be placed as close to the center of the disk as possible. There are, however, two notable exceptions:

1. Applications that perform a large amount of sequential reads or writes experience higher throughput when the data is located on the outer edge of the disk due to the fact that there are more data blocks per track on the outer edge of the disk than the other disk regions.
2. Logical volumes with Mirrored Write Consistency (MWC) enabled should also be located at the outer edge of the disk, as this is where the MWC cache record is located.

Inter-disk allocation policy

The inter-disk allocation policy is used to specify the number of disks that contain the physical partitions of a logical volume. The physical partitions for a given logical volume can reside on one or several disks in the same volume group depending on the setting of the *Range* option:

- The *maximum* range setting attempts to spread the physical partitions of a logical volume across as many physical volumes as possible in order to decrease the average access time for the logical volume.
- The *minimum* range setting attempts to place all of the physical partitions of a logical volume on the same physical disk. If this cannot be done, it will attempt to place the physical partitions on as few disks as possible. The minimum setting is used for increased availability only and should not be used for frequently accessed logical volumes. If a non-mirrored logical volume is spread across more than one drive, the loss of any of the physical drives will result in data loss. In other words, a non-mirrored logical volume spread across two drives will be twice as likely to experience a loss of data as one that resides on only one drive.

The physical partitions of a given logical volume can be mirrored to increase data availability. The location of the physical partition copies is determined by the setting of the *Strict* option. When *Strict = y*, each physical partition copy is placed on a different physical volume. When *Strict = n*, the copies can be on the same physical volume or different volumes. When using striped and mirrored logical volumes in AIX 4.3.3 and above, there is an additional partition allocation policy known as *Super Strict*. When *Strict = s*, partitions of one mirror cannot share the same disk as partitions from a second or third

mirror, thus further reducing the possibility of data loss due to a single disk failure.

In order to determine the data placement strategy for a mirrored logical volume, the settings for both the Range and Strict options must be carefully considered. As an example, consider a mirrored logical volume with range setting of minimum and a strict setting of yes. The LVM would attempt to place all of the physical partitions associated with the primary copy on one physical disk, with the mirrors residing on either one or two additional disks, depending on the number of copies of the logical volume (2 or 3). If the strict setting were changed to *no*, all of the physical partitions corresponding to both the primary and mirrors would be located on the same physical disk.

Write-scheduling policy
When mirrored copies of the data are maintained by the LVM, the setting of the mirrored write-scheduling policy determines the sequence of the write operations to the logical volume. Mirrored writes can be either parallel or sequential.

The sequential write-scheduling policy writes the physical partitions for a mirrored logical volume in the sequence *primary*, *secondary,* and *tertiary,* where primary represents the first copy of the logical volume, secondary the second, and tertiary the third. A write request for a given copy must complete prior to updating the next copy in the sequence. Read requests are first directed to the primary copy. If that copy cannot be accessed due to a drive failure or physical corruption, the request is redirected to the secondary copy and so forth. While the redirected read request is being processed, the LVM automatically attempts to correct the copies on which the read failed through a process known as *bad block relocation*.

The parallel write-scheduling policy schedules the write operation to all of the copies at the same time. The write request is satisfied when the copy that takes the longest to update is finished. The parallel write-scheduling option provides the best performance, as the duration of the write request is limited only by the speed of the slowest disk and not the number of copies that must be updated. Read requests are directed to the copy that can be accessed in the shortest amount of time. Just as with the sequential-write policy, failed read requests will automatically initiate bad block relocation.

Write-verify policy
When the write-verify policy is enabled, all write operations are validated by immediately performing a follow-up read operation of the previously written data. An error message will be returned if the read operation is not

successful. The use of write-verify enhances the integrity of the data but can drastically degrade the performance of disk writes.

Recommendations for performance optimization
As with any other area of system design, when deciding on the LVM policies to be used, a decision must be made as to which is more important; performance or availability. The following LVM policy guidelines should be followed when designing a disk subsystem for performance:

- When using LVM mirroring:
  - Use a parallel write-scheduling policy.
  - Allocate each logical partition copy on a separate physical disk by using the Strict option of the inter-disk allocation policy.
- Disable write-verify.
- Allocate heavily accessed logical volumes near the center of the disk, with the exceptions noted in Section “Intra-disk allocation policy” on page 171.
- Use a intra-disk allocation policy of *maximum* in order to spread the physical partitions of the logical volume across as many physical disks as possible.

5.3.10 Raw logical volumes versus Journaled File Systems (JFS)

There has been a long standing debate surrounding the use of raw logical volumes (raw devices) versus Journaled File Systems (JFS), especially in database environments. Advocates of raw logical volumes stress the performance gains that can be realized through their use, while JFS supporters emphasize the ease of use and manageability features of file systems. As with many other aspects of system design, a decision must be made as to which is more important; performance or manageability.

In order to better understand the performance advantages associated with raw logical volumes, it is helpful to have an understanding of the impact of the JFS file system cache. Most UNIX file systems set aside an area of memory to hold recently accessed file data, thereby allowing a physical I/O request to be satisfied from memory instead of from disk. In AIX, this area of memory is known as the *buffer cache*. If an application requests data that is not already in memory, AIX will read the data from disk into the buffer cache and then copy the data to a user buffer so that it can be used by the application. Therefore, each read request translates into a disk read followed by a copy of the data from the buffer cache to the user buffer.

Because the data is read from memory, I/O requests can be satisfied in nanoseconds instead of the milliseconds that would be required in order to

fetch the data from disk. In addition, AIX JFS file systems employ the use of a sequential read-ahead mechanism to pre-fetch data into memory when it is determined that a file is being accessed sequentially.

In non-database environments, the AIX buffer cache can significantly reduce I/O wait time for heavily accessed files. However, the performance benefits of file system caching in database environments are not so clear. This is due to the fact that most modern RDBMS systems also allocate a region of memory for caching frequently accessed data. The end result when using JFS file systems is that the data is double-buffered: Once in the file system buffer cache and once in the RDBMS cache. In most cases, the extra memory used by the file system buffer cache could be better utilized by the database buffers.

The primary benefit of raw logical volumes is that they bypass the AIX file system buffer cache entirely by directly accessing the underlying logical device. The extra memory saved by eliminating the file system cache can then be allocated to the database to increase the data buffers. In addition, overall CPU utilization is decreased due to the fact that the system no longer has to copy the data from the file system cache to the user buffers. Another benefit of raw logical volumes is that there is no node management overhead, as opposed to JFS where the node is locked when the file is accessed.

The main drawback of using raw logical volumes lies in the increased administration costs. Because raw logical volumes do not exist as files at the UNIX level, many of the traditional tools and utilities for managing data will not work. Backup and recovery operations can be especially difficult when using raw logical volumes. Many third party vendor backup applications (such as the Tivoli Storage Manager) cannot directly read raw logical volumes and must rely on the UNIX `dd` command to copy the raw data to a UNIX filesystem prior to backing up the data. Restores are equally complicated as the data must first be restored to a UNIX filesystem and then copied to the raw logical volume. If this approach is used, additional disk space will be required for the JFS file systems used to temporarily hold the data contained in the raw logical volume. However, if the raw logical volumes can be backed up directly to a locally attached tape drive using the `dd` command, this will not be an issue.

Some raw logical volume benchmarks point to an overall disk I/O throughput gain of 5-30 percent when compared to JFS file systems. However, the actual performance gains that can be realized in a typical database environment will vary depending on the I/O workload mix of the application. Applications that perform a large amount of random I/O operations, such as OLTP systems, benefit the most from the use of raw logical volumes. Applications that

perform a large amount of sequential I/O operations, such as DSS systems, benefit from the sequential read ahead feature of JFS file systems.

5.4 Asynchronous Communication adapters

Serial Asynchronous Communication was the first commercial communication method for the UNIX environment. Based on the teletype communication method, serial asynchronous ports permit the connection of different serial devices such as terminals, printers, fax machines, and modems. Serial asynchronous communication is associated with a hardware line between the devices and the host. To understand how it works, we will give a brief overview of serial communications.

Inside the CPU, data and addresses are processed as words (32 bits, 64bits, etc.). There is a chip to convert the parallel data to serial data for transmission via the serial link. At the destination end, the serial data is buffered and when all the bits of a byte have arrived, the buffer is converted to parallel data.

These streams need a signal to define when the data starts and when it ends. The synchronization is the process of timing the serial transmission in order to properly identify the data being sent. The two most common modes of synchronization are synchronous and asynchronous.

Synchronous communication is used for a continuous transfer of large amounts of data. The data blocks are grouped and spaced in regular intervals and are preceded by special characters called *sync* or synchronous idle characters. The sync character is used to synchronize the connection so data transmission can begin.

Asynchronous communication is used when the data transfer is randomly started and stopped. In asynchronous transfers, there is a start bit and stop bit that specify the beginning and the end of a character. Each character is preceded by a start bit and followed by one or more stop bits.

5.4.1 Terms used in serial communication

This section describes the terminology used in serial communications.

5.4.1.1 Bits per character

Indicates the number of bits used to represent a single data character during serial communication. With seven bits, it is possible to represent 128 characters that make up the standard ASCII character set. With eight bits, it is

possible to represent 256 characters that make up the ASCII extended character set.

5.4.1.2 Bits per second (bps)

The number of data bits (1s or 0s) that are transmitted per second over the communication line. It refers to the communication line speed.

5.4.1.3 Baud rate

The number of times a serial communication signal changes states per second. If the signal changes each time a data bit enters the device, then bps is equal to baud. This is the most common case in the serial asynchronous communication because the data transmission is made over a baseband modulation (a low voltage for a 1, a high voltage for a 0).

5.4.1.4 Parity bit

An optional parameter used in serial communication to determine if the data character being transmitted is correctly received by the remote device. Options are: none, even, odd, space, and mark. It is used for error detection on a communication line. Both sending and receiving systems must be configured identically.

5.4.1.5 RS-232-C, RS-232-D, and RS-422 standard

It defines the mechanical and electrical specifications for the most common connector designs in serial data communication. It defines 25 pins with assigned signals. With RS-232-C, devices are divided in two types; DTE (Data Terminal Equipment) such as computers and terminals, uses pin 2 (TxD) as an output, and DCE (Data Communication Equipment) such as modems, uses pin 2 as input. The RS-232-D is a revision of RS-232-C. The TxD is the transmit data pin, and the RxD is the receive data pin.

The RS-422 uses two pairs of cables to provide a differential signal for TxD and RxD. This differential signal makes the serial communication more resistant to electrical interference, meaning the RS-422 can provide longer serial communication cables than RS-232.

5.4.1.6 Simplex

It is the simplest connection between two devices. Simplex, or one-way communication, allows data to be transmitted in one direction only and requires only TxD (or RxD) and the signal ground (SG) to be connected.

5.4.1.7 Duplex

There are two forms of two-way communication; half duplex and full duplex.

Half duplex uses a single pair of wires to allow data to be transmitted in two directions, but not simultaneously.

Full duplex allows data communication to take place in two directions simultaneously over two separate lines or wires.

5.4.2 Flow control

Serial devices do not process data as quickly or efficiently as the CPU they are connected to. There are memory buffers associated with the data transmission that need some type of flow control to limit the amount of data transmitted by or to the CPU.

Flow control also is referred to as handshaking. There are two types of flow control; hardware and software.

With hardware flow control, wires and voltage levels are used for data transmission control.

RTS/CTS (ready to send/clear to send) uses pins 4 and 5 to control the flow. A low RTS signals the sending site to stop transmitting data. When the buffer is almost empty, RTS is raised again, issuing a signal to send more data to the other site.

DTR/DSR (data terminal ready/data set ready). This hardware flow control is normally generated by devices, such as printers. DTR indicates that the device is ready to communicate with the CPU. The CPU uses DSR to control the data flow.

DCD (data carrier detect) is a signal in pin 8 also referred to as received line signal detector. DCD is an output signal for a modem and an input signal for the CPU. When DCD is high, the CPU knows a modem connection has been made. With DCD in low state, the CPU knows the modem connection is terminated.

The software flow control involves sending of data transmission control characters along the data stream.

XON/XOFF (transmitter on/off). This flow control operates with the buffer capacity in both sides of the data transmission. Just before the device buffer reaches its maximum capacity, the device will send an XOFF character to the CPU. The data transmission is then stopped. When the device buffer is almost empty, it will send an XON signal character back to the CPU to restart transmission.

5.4.3 Asynchronous adapter overview

AIX is a multiuser operating system allowing many users to access system resources and applications. Each user must be connected through a terminal session. The connection can be local or remote through a serial port or network connection using Transmission Control Protocol/Internet Protocol (TCP/IP).

Each system unit has at least two standard serial ports available (some systems have three serial ports). These ports can support asynchronous communication and device attachment.

Two standard ports are sufficient for users requiring an additional ASCII terminal or a modem. Many users, however, require additional asynchronous ports.

Asynchronous communications products offer the advantages of low cost, multi-user, medium- to high-performance terminal and device communications. Asynchronous ports allow attachment of asynchronous peripheral devices that meet EIA 232 or EIA 422 standards such as:

- Asynchronous modems
- Bar code scanners
- Graphic and character printers
- Keyboard and display terminals
- Personal computers
- Plotters and printers
- Point-of-sale terminals
- Sensors and control devices
- Text scanners
- Time clocks

5.4.4 Evaluating asynchronous communications options

Expanded asynchronous capability can be added to the system unit with direct-attached adapters using the Peripheral Component Interconnect (PCI) bus, distributed subsystems, or local area network (LAN) attached communications servers. Several factors will influence the type of asynchronous connectivity you choose. Table 18 on page 180 summarizes these products.

Table 18. Asynchronous communications adapters

Asynchronous attachment	Bus Type	Feature code/ machine type (model)	Maximum data rate per port (KBits/sec)	Salient features
Standard serial port	System planar	n/a	Selectable based on baud rate generator clock speed of Universal Asynchronous Receiver and Transmitter (UART)	Standard feature
8-port EIA 232/422	PCI	2943	230.4	Greater efficiency
128-Port Controller	PCI	2944	230.4	Efficiency, higher device counts
232 RAN		8130	57.6	Remote capability
Enhanced 232 RAN		8137	230.4	Remote capability
16-Port RAN EIA 422		8138	230.4	Remote capability

The first feature in this table represents the attached serial ports that are standard with every system unit. The 8-port EIA 422 is a direct-attached adapter. The 128-port asynchronous subsystem includes the Remote Asynchronous Nodes (RANs) that attach to it.

5.4.4.1 Standard-attached asynchronous ports

Most system unit models have two integrated (standard) EIA 232 asynchronous serial ports as shown in Figure 69 on page 181. EIA 232 asynchronous serial devices can be attached directly to the standard serial ports using standard serial cables with 25-pin D-shell connectors. Some multiprocessor systems have a third serial port used for communication to the remote service center.

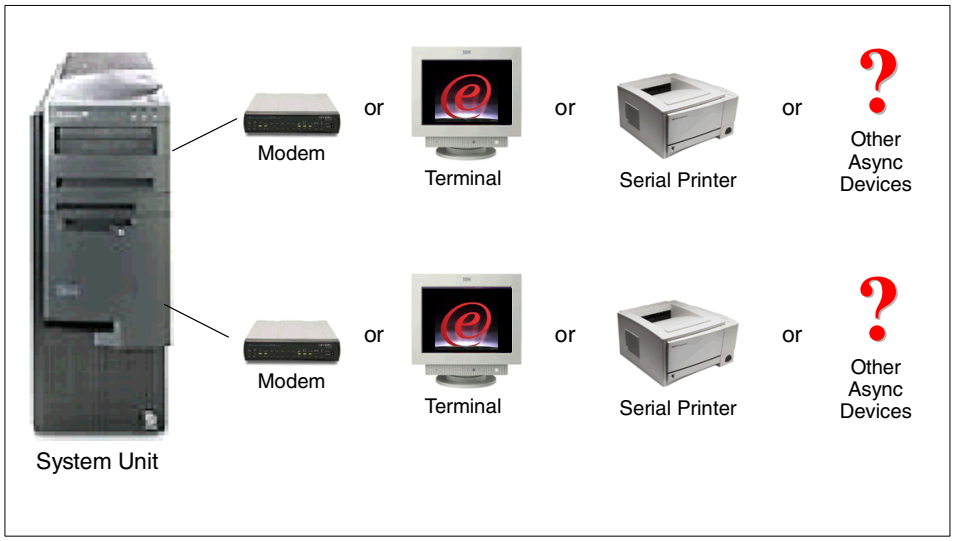


Figure 69. Standard-attached asynchronous devices

5.4.4.2 Direct-attached asynchronous ports

Each of the direct-attached adapters requires a bus slot, and can only be used in systems that support the required bus type. The 128-port and 8-port PCI adapters are intelligent adapters that provide significant offload of the main system processor. As shown in Figure 70 on page 182, the 8-port adapters use fan-out cables to connect to devices and require no additional power supply.

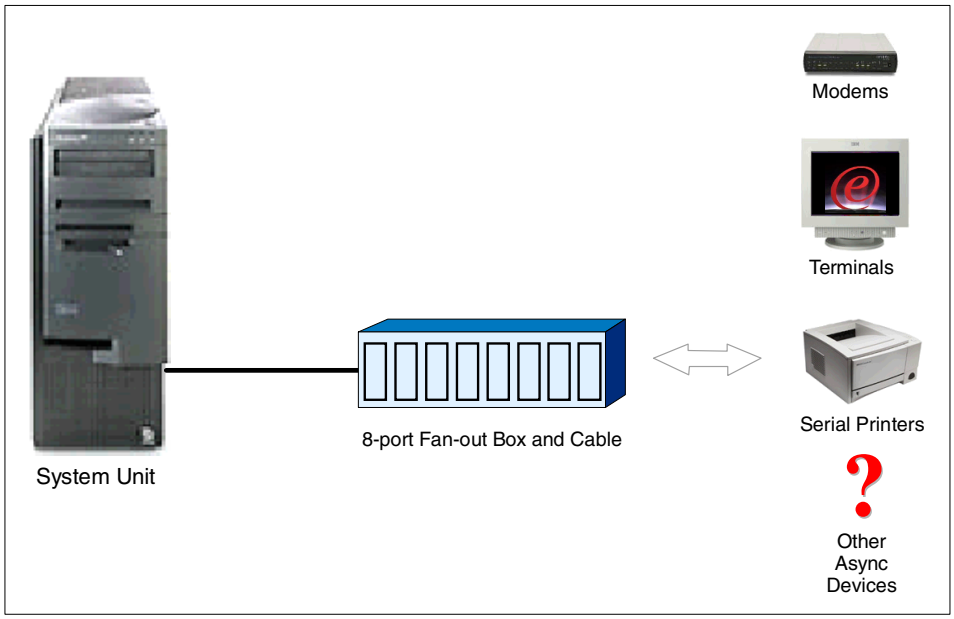


Figure 70. Direct-attached asynchronous devices

5.4.4.3 Node-attached asynchronous ports

The 128-port adapter allows attachments of one to eight remote asynchronous nodes (RANs). Each RAN has 16 asynchronous ports for connection to devices and is a separately powered unit. Up to four RANs can be daisy-chain connected from each of two connections on the 128-port adapter card (see Figure 71 on page 183). In AIX Version 4.2 (or AIX V4.1.5 with the latest update) and later, RANs support 16 EIA 232 devices or 16 EIA 422 devices.

The 128-port controller is an intelligent adapter that increases the number of asynchronous sessions possible at a given CPU utilization level. The following are additional characteristics of the 128-port feature:

- RANs may be located up to 300 meters from the system processor using 8-wire shielded cabling while maintaining full performance ratings.
- Distance may be extended to 1200 meters by reducing the data rate between the RANs and the system processor.
- RANs may be remotely located from the system processor using a synchronous EIA 232 and EIA 422 modem. Each four-RAN daisy chain is allowed only one modem pair at any point in the chain.

- System performance is enhanced by offloading tty character processing from the system processor.

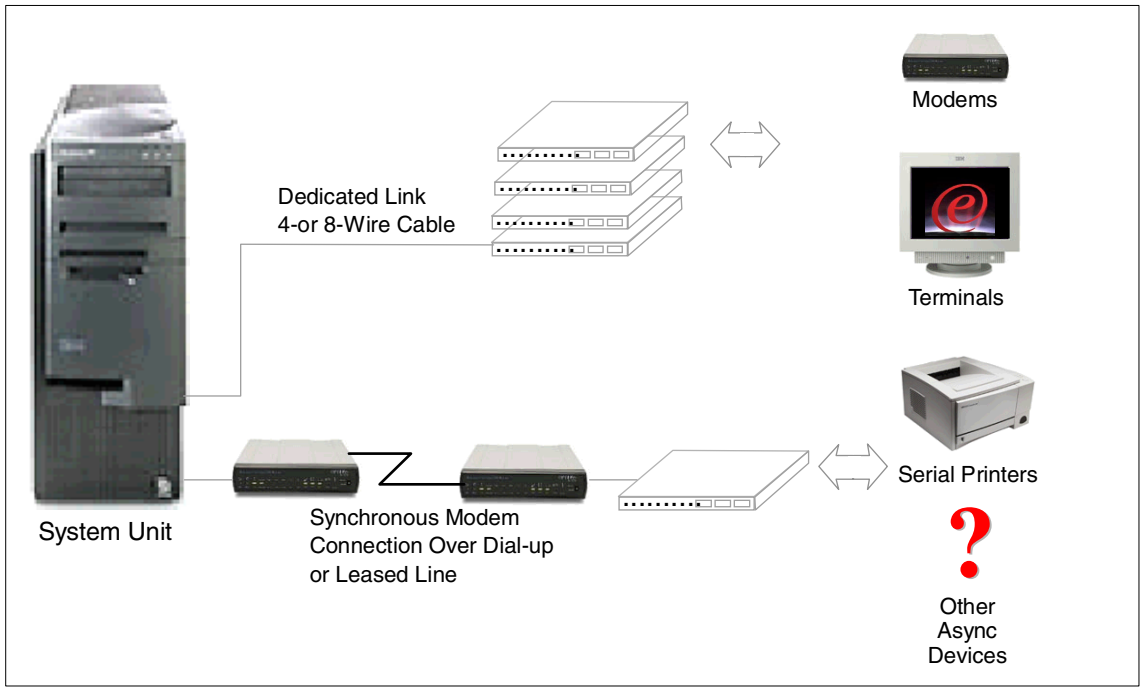


Figure 71. Node-attached asynchronous ports

5.4.5 Product selection considerations

This section will help you determine what asynchronous product you should choose for a particular situation.

5.4.5.1 Product Selection Aid

The following questions will help you choose an AIX offering for your installation.

Expandability

- How many asynchronous ports are needed?
- How many ports will be needed in the future?

Topology

- Will devices be in other buildings or remote locations?
- Does an Ethernet LAN exist?

- Where will system/network administration be done?
- Is there an HACMP cluster?
- What type of cabling is required or already there?
- Is connection to multiple hosts a requirement?

Performance

- Is your AIX application CPU-intensive?
- What types of devices will be attached?
- What is the relative asynchronous bandwidth demand for the aggregate sum of the devices?

Table 19. Relative device bandwidth demand

Low demand	Moderate demand	High demand
ASCII terminals, point-of-sale terminals, asynchronous modems	Printers, low-speed FAX/modems, bar code scanners	Serial X-terminals, high-speed FAX/modems, high-speed printers, file transfer applications

PCI bus slots

- How many slots are available for asynchronous adapters?

Device interface requirement

- What asynchronous interface is required; for example, EIA 232 or EIA 422A?
- Do the devices or applications require the full EIA 232 interface?
- Is a remote parallel port required for printing?

Security

- Is dedicated calling a requirement?
- Is a device name to a physical device mapping required?
- Is system assurance kernel (SAK) required?

Figure 20 shows the detailed product characteristics:

Table 20. Asynchronous attachment product characteristics

	Native Serial Ports	8-port	128-port with RAN
Number of asynchronous ports per adapter	n/a	8	128
Maximum number of adapters	n/a	8	8
Maximum number of asynchronous ports	2 or 3	64	1024
Maximum speed (KBits/sec)	230.4	230.4	230.4
Attachment method	standard	direct	node
Asynchronous electrical interfaces supported	EIA232	EIA 232 EIA 422A	EIA 232 EIA 422A
Standard connector	DB25M / MODU	DB25M	RJ-45
DB25 cable options	n/a	n/a	RJ-45 DB25
Rack mount option	n/a	n/a	yes
Power supply	n/a	n/a	external
Signals supported (EIA 232)	TxD, RxD, RTS, CTS, DTR, DSR, DCD, RI	TxD, RxD, RTS, CTS, DTR, DSR, DCD, RI	TxD, RxD, RTS, CTS, DTR, DSR, DCD, RI

5.4.5.2 Adapter applications

Each product offering is characterized by a representative scenario for its strengths:

8-Port PCI Bus EIA 232/EIA 422

- PCI slot available
- Fewer than eight ports required with little or no expansion
- Requires all EIA 232, all EIA 422, or a mix of EIA 232 and EIA 422 ports
- Offload character interrupt and terminal I/O processing from the main CPU
- Asynchronous speeds to 230 Kbps
- Maximum performance for high speed modems with data compression

128-Port Adapter

- A PCI bus slot available for asynchronous I/O
- Sixteen ports now with expansion of up to 128 ports without additional slots
- Most distant terminal located about 90 meters (300 feet) from the system could run at maximum data rate for the PCI adapter
- Terminals planned; nearby or on premises, distant on premises, and remote
- Need high asynchronous throughput with low processor demand
- Need terminal attached printer capability
- Need to connect to remote premises through fiber-optic or synchronous modems

5.4.5.3 Customer scenarios

The following represents some typical customer scenarios with suggested asynchronous solutions:

Real estate office

- Simplicity and cost are high priority.
- AIX server.
- Six to ten devices tied into the server accessing the database.
- One slot is available for asynchronous.
- Devices are less than 61 meters (200 feet) from the server.

Solution: 8-port asynchronous adapter.

Retail point-of-sale

- Cost per seat is high priority.
- AIX server.
- 20 or more ASCII terminals; for example, cash registers.
- One slot is available for asynchronous.
- Future expansion for additional terminals is planned.

Solution: 128-port asynchronous controller with two RANs. Future expansion with additional RANs.

5.4.6 Topology considerations

The asynchronous family of adapters offers a wide variety of choices where distance topology is concerned.

The maximum cable lengths from the standard- and direct-attach adapters is generally the distance between the port and the asynchronous device, operating at the maximum specified data rate. The 128-port adapter is measured from the adapter card to the daisy-chained RAN attached to it. With the 128-port, unlimited distances can effectively be achieved by using the EIA 422 synchronous modems to attach the RANs to the adapter.

Proper cabling is extremely important, and is unique to each environment.

5.5 LAN/WAN Adapters

Network-specific applications have grown beyond simple client/server computing to include multimedia, data warehousing, and internet and intranet access. Many of these applications require the transmission of large files over the network, which in turn adds additional bandwidth requirements to the network.

Network performance is dependent on the type of the network, such as Token Ring, Ethernet, FDDI, or ATM. But it is also highly dependent on the application, the frequency of data transfers, and the amount of data that is transferred through the network, as well as on the design of the entire network.

This chapter will describe the different kinds of LAN and WAN protocols, the adapters, and their related performance features. It also offers some considerations about performance tuning.

5.5.1 Ethernet

The Ethernet standard was defined by Institute of Electrical and Electronic Engineers (IEEE) in specification IEEE 802.3. The standard specifies the physical medium, *carrier sense multiple access with collision detection* (CSMA/CD) access method, and frame format. In CSMA/CD access method, each station contends for access to the shared medium. If two stations try sending the packets at the same time, a collision will result. The CSMA/CD access method is designed to restore the network to normal activity after a collision occurs, as collisions are normal in Ethernet shared networks. The original 10 Mbps shared Ethernet network was based on coaxial cable

physical medium, but later the standard was extended to shielded and unshielded twisted pair, and fibre optic.

Today, most Ethernet use the twisted-pair wiring (also known as unshielded twisted pair (UTP) or 10 Base-T) with RJ-45 connectors.

5.5.1.1 Fast Ethernet

Fast Ethernet is an extension of the popular 10 Base-T Ethernet standard, supporting both 10 and 100 Mbps media speed. Fast Ethernet retains the data format and protocols of 10 Mbps Ethernet, so no changes are required in higher level protocols and applications. Fast Ethernet standards provide for auto-negotiation of media speed, and Ethernet interfaces that can be installed and run at either 10 or 100 Mbps. With dual speed products, users who are planning future 100 Mbps implementations can purchase a 10/100 Mbps product today and use the 10 Mbps speed in their existing networks, then later upgrade to 100 Mbps when and where it is needed.

When using UTP (unshielded twisted pair), the network can be operated in either half-duplex or full-duplex mode. Most adapters now support both modes. With the proper Ethernet switch, full-duplex mode can double the network throughput for busy servers.

Most Ethernet hubs only support half duplex mode and are lower cost. Ethernet switches typically support both half and full duplex modes. They also allow multiple sessions to be running at media speed through the switch. For example, node A can be talking to node B at 100 Mbps and node C can be talking to node D at 100 Mbps. With a hub, only one node can be transmitting at a time.

For best performance with fast Ethernet, use a switch and adapters that support full-duplex mode. This allows a server to be receiving and sending at 100 Mbps concurrently.

5.5.1.2 Gigabit Ethernet

Gigabit Ethernet is an extension to 10 Mb and Fast Ethernet that will provide seamless inter operability with the existing 10 Mb and Fast Ethernet, and is compatible with existing networking protocols, networking operating systems, network applications, and networking management tools. It uses a combination of proven protocol technologies adopted by the original IEEE 802.3 Ethernet specification and the ANSI X3T11 Fibre channel specification. Gigabit Ethernet retains the standard 10/100 Base-T frame size and format and the same CSMA/CD (Carrier sense multiple access/collision detection) scheme. However, it uses Fibre channel's physical layer as the underlying transport mechanism, requiring the packets to be encoded in 8B/10B on the

physical media. The full duplex implementation of Gigabit Ethernet, as in Fast Ethernet, does not require the CSMA/CD scheme, but retains support for the Ethernet frame format. Gigabit Ethernet uses two methods of transmission; full-duplex transmission and half-duplex transmission. With full-duplex transmission, signals travel in both directions on the same connection simultaneously. This allows the aggregate data rate of a Gigabit Ethernet network to be doubled to 2-Gbps. With full-duplex, CSMA/CD access control mechanism need not be invoked as collisions are not possible. In half-duplex transmission, signals can travel in both directions on a wire, but not simultaneously.

IBM offers high performance 802.3z standard compliant Gigabit Ethernet Network Interface Cards (NIC) for high end RS/6000 and pSeries servers and workstations to help solve high bandwidth network needs.

The 10/100/1000 Base-T Ethernet adapter is a copper (UTP) based adapter using an RJ-45 connector. It can run any of the three media speeds. The Gigabit Ethernet - SX PCI Adapter fiber adapter operates at gigabit speed only.

Both adapters are 64-bit adapters and will work in either 32-bit or 64-bit PCI slots, but will perform best in a 64-bit slot. They also work at 33 Mhz or 66 Mhz bus speeds and should be used in 66 Mhz (or 50 Mhz in some systems) when possible. When running at 66 Mhz and in a 64-bit slot, the jumbo frame mode provides the best performance.

5.5.1.3 Ethernet performance tuning recommendations

Ethernet is one of the contributors to the *least common denominator* algorithm of maximum transmission unit (MTU) choice. If a configuration includes Ethernet and other LANs and there is extensive traffic among them, the MTUs of all of the LANs may need to be set to 1500 bytes to avoid fragmentation when data enters an Ethernet. Following are some guidelines:

- The gigabit Ethernet adapters support Jumbo Frames (MTU 9000). This can provide improved performance in special cases where all the nodes in the network can run the gigabit adapter and can thus use the larger jumbo frames. The larger frames allow the system to send more data with the same amount of CPU, or the same amount of data with less CPU.
- Applications should use block sizes in multiples of 4096 bytes.
- For 10 and 100 Mbit Ethernet, the tcp_sendspace and tcp_recvspace parameters should be set to 10 times the MTU size. Thus, for a 1500 byte MTU, 16384 would be a good choice. This rule of thumb does not apply for gigabit Ethernet.

5.5.2 Token Ring

The Token Ring model is a type of LAN that was developed under the auspices of the IEEE 802.5 Subcommittee. It is a token access procedure used with a sequential topology (ring). All stations in the ring can receive the token, but only one station, the one that holds the token, can transmit at a time. This avoids the possibility of data collisions because there is only one token. Ring speed in a Token Ring can be either 4 or 16 Mb/s.

5.5.2.1 Advantages

Token Ring is a convenient, easy-to-handle solution to connect machines in a LAN. It provides more capacity than conventional 10 Mbit Ethernet. However, today 100 Mbit Ethernet or Gigabit Ethernet performance far exceeds the 16 Mbps Token Ring performance and capacity.

Due to its architecture and the STP cabling, Token Ring allows more-flexible designs that are less susceptible to interference. Because of the sequential topology, even busy networks are safe from collisions, which can take a heavy toll on 10 Mbit Ethernet performance when using coaxial cable.

5.5.2.2 Disadvantages

Token Ring is relatively expensive due to the STP cabling and the required multistation access unit (MAU) to connect the stations. The network's capacity is limited to 16 Mb/s.

5.5.2.3 Token Ring (4 Mb) performance tuning recommendations

- The default MTU of 1492 bytes is appropriate for Token Rings that interconnect to Ethernet or to heterogeneous networks in which the minimum MTU is not known.
- Unless the LAN has extensive traffic to outside networks, the MTU should be raised to the maximum of 3900 bytes.
- Applications should use block sizes in multiples of 4096 bytes.
- Socket space settings can be left at the default values.
- If the workload includes extensive use of services that use UDP, such as NFS or RPC, *sb_max* should be increased to allow for the fact that each 1492-byte MTU uses a 4096-byte buffer.

5.5.2.4 Token Ring (16 Mb) performance tuning recommendations

- The default MTU of 1492 bytes is appropriate for token rings that interconnect to Ethernet or to heterogeneous networks in which the minimum MTU is not known.

- Unless the LAN has extensive traffic to outside networks, the MTU should be increased to 8500 bytes. This allows NFS 8 KB packets to fit in one MTU. Further increasing the MTU to the maximum of 17000 bytes seldom results in corresponding throughput improvement.
- Applications should use block sizes in multiples of 4096 bytes.
- Socket space settings can be left at the default values. If the workload includes extensive use of services that use UDP, such as NFS or RPC, the MTU must be left at the default because of interconnections, and *sb_max* should be increased to allow for the fact that each 1492-byte MTU uses a 4096-byte buffer.

Token Ring now supports full-duplex mode. This requires a Token Ring switch that also supports full-duplex (such as an IBM 8272 switch). Most of the IBM PCI Token Ring adapters support full-duplex mode. Some older adapters may need a firmware upgrade to support this mode.

5.5.3 Fibre Channel

Fibre Channel, a family of ANSI standards, is a common, efficient transport system supporting multiple protocols or raw data using native Fibre Channel delivery services. It is a highly-reliable, gigabit interconnect technology that allows concurrent communications among workstations, mainframes, servers, data storage systems, and other peripherals using SCSI and IP protocols. It provides interconnect systems for multiple topologies that can scale to a total system bandwidth on the order of a terabit per second.

Fibre Channel has been adopted by the major computer systems and storage manufacturers as the next technology for enterprise storage. It eliminates distance, bandwidth, scalability, and reliability issues of SCSI.

5.5.3.1 Key features of Fibre Channel

Some of the key features of Fibre Channel are:

- Performance from 266 megabits/second to over four gigabits/second
- Support for distances up to 10 km
- High-bandwidth utilization with distance insensitivity
- Greater connectivity than existing multidrop channels

5.5.3.2 Performance enhancing features

- Confirmed delivery, enhancing the reliability of the protocol stack, or the option of bypassing the protocol stack for increased performance.

- Complete support for traditional network self discovery. Full support of ARP, RARP, and other self-discovery protocols.
- Support for dedicated bandwidth point-to-point circuits, shared bandwidth loop circuits, or scalable bandwidth switched circuits.
- Full support for time synchronous applications like video, using fractional bandwidth virtual circuits.
- Efficient, high-bandwidth, low-latency transfers using variable length (0-2KB) frames. Highly effective for protocol frames of less than 100 bytes as well as bulk data transfer using the maximum frame size.

5.5.3.3 Advantages

The fibre channel adapter provides a high-performance connection between systems that support TCP/IP via fibre channel. Selected RS/6000 systems, RS/6000 SP systems, and disk arrays can be used with it.

5.5.3.4 Disadvantages

The range of attachable devices is limited to a few products within the RS/6000 system family.

Conclusion: Fibre channel is suitable for companies requiring large and high-speed file transfer.

5.5.4 ATM

Asynchronous Transfer Mode (ATM) is a connection-oriented packet-switching technology that uses fixed-size packets, referred to as *cells*, to carry the traffic in the network.

It provides a foundation for high-speed, multimedia networking. It has the ability to transport data, voice, and video communications over the same network.

5.5.4.1 Key features of ATM

ATM is, fundamentally, a connection-oriented technology. Before data can be transferred, a connection is established between the sending and receiving nodes. This is in contrast to connection-less standards such as Ethernet or Frame Relay, where a node sends data and the data is transferred based upon its packet address. Once the data leaves the sending node it is up to the network infrastructure to deliver the data as dictated by its packet's address. Connection-less data packets need to carry enough information in their header to allow them to be routed across the network.

The second key feature is that ATM is a cell-based design rather than packet-based. ATM consists of fixed-length cells of 53 bytes. The cell is comprised of a 5-byte header and 48-byte payload. The cells can contain data, image, video, and voice transmissions. These design features, in addition to typical transmission speeds of 155 Mbps and 622 Mbps, provide the ability to build high-performance switching systems for public and private networks.

5.5.4.2 How the ATM network works

In an ATM network, an end-system requests a connection to another end-point by transmitting a signaling request across the User Network Interface (UNI) to the network. This request is passed to a signaling entity within the network, which passes it across the network to the destination. If the destination agrees to form a connection, a virtual circuit is set up across the ATM network between the two end-systems. Mapping is defined between the Virtual Path Identifier (VPI)/Virtual Circuit Identifier (VCI) on both ends of the UNI, and between the appropriate input link-and corresponding output link-of all intermediary switch. When the virtual circuit is established between two nodes, the end nodes exchange information by sending ATM cells across their respective UNI. Each of these cells contains information and the VPI/VCI value assigned to that virtual circuit on each UNI.

Once the cells are transmitted across the UNI, they are relayed from link to link through ATM switches, each of which only changes the VPI/VCI values as appropriate, and guides the cells from an input port to the appropriate output port, and finally to the destination end-station across the destination UNI.

5.5.4.3 Advantages

- Supports all standards-based cabling system.
- ATM can be more efficient than Ethernet or FDDI if you use the default MTU size of 9180 and you are sending large amounts of data that can take advantage of the large MTU size. However, many companies run LAN Emulation (LANE) and thus end up running ATM with a MTU size of 1500 bytes. In this mode, ATM is actually less efficient than Ethernet due to the overhead that ATM has (such as cell overhead, overhead of the LANE layer, or overhead due to the switched virtual circuits).
- Because ATM is a switched network, the full bandwidth is provided for each exclusive connection between two stations in the network.

5.5.4.4 Disadvantages

- ATM is more complex than other network types. That results in a network and system that is harder to administer and troubleshoot. It can also result in more problems.
- ATM does not support broadcasts. Users that need broadcast capability have to use LAN Emulation (LANE) mode.
- If a user is going to use LAN Emulation, then there is normally no performance benefit from using ATM. Fast Ethernet (100 Gbps) is lower cost and much easier to maintain than using LANE. Gigabit Ethernet is much simpler and easier to use than ATM 622, and is also faster.

5.5.4.5 ATM performance tuning recommendations

- Unless the LAN has extensive traffic to outside networks, the default MTU of 9180 bytes should be retained.
- Where possible, an application using TCP should write multiples of 4096 bytes at a time (preferably 8 KB or 16 KB) for maximum throughput.
- Use `no -o` to set tcp_sendspace, tcp_recvspace, udp_sendspace, udp_recvspace, and sb_max to the values stated in Table 21 on page 196.

Conclusion: Users selecting ATM should do so very carefully.

5.5.5 General network tuning recommendations

There is a vast number of network parameters that can be tuned. Some of the most important ones (MTU size, tcp_sendspace, tcp_recvspace, udp_sendspace, udp_recvspace, sb_max, and RFC1323) are discussed in this section.

Tuning sb_max

In general, the sb_max parameter should be set to at least twice the size of the largest send or receive space, but must be at least equal to the largest tcp or udp send or receive space value. If the sb_max value is increased, the udp_recvspace value has to be increased as this is the parameter that actually controls how much socket buffer space can be used for each socket. The sb_max parameter is just the upper limit. The udp_recvspace parameter cannot be set any larger than sb_max. However, setting these values too large can hurt performance as well.

Tuning UDP

The udp_sendspace parameter should generally be set to 65536.

The setting of udp_recvspace is harder to compute as it varies by network adapter type, UDP sizes, and number of datagrams queued to the socket. The udp_recvspace should be set larger rather than smaller because packets will be discarded if it is too small.

Monitor the “dropped due to full socket buffers” statistic from `netstat -s` and adjust udp_recvspace upwards if there are a lot of these errors relative to the total UDP packets received.

The udp_recvspace required is dependent on the size of the receive buffer used by the network device driver. Larger receive buffers will consume more udp_recvspace.

Below is a list of the buffer sizes used by some drivers:

- Ethernet, normally 5 KB or 4 KB for gigabit jumbo frames.
- Token Ring, normally 2 KB for MTU 1492.
- FDDI, 4 KB buffers. Note that it will take two 4 KB buffers to receive a maximum size FDDI frame of 4352 bytes. PCI FDDI provides a SMIT option to use 8 KB receive buffers.
- ATM, 4 KB buffers for PCI ATM.

For applications like NFS that use a well known port (socket), the udp_recvspace number must be much larger. The same technique of monitoring `netstat -s` for “dropped due to full socket buffers” is the suggested way of determining whether you need to do some tuning. Besides increasing udp_sendspace, the following is recommended for NFS:

- Increase the number of nfsd’s (via the nfs_max_threads `nfso` option).
- If there are still dropped packets, then increase the NFS server UDP socket buffer size (via the nfs_socketsize `nfso` option). Note that this size must be less than sb_max, so you may need to increase sb_max as well.

Tuning TCP

Following are the “minimum” recommendations for best performance on various networks:

Table 21. Minimum parameter settings for best network performance

Device	Speed	MTU size	tcp_sendspace	tcp_recvspace	sb_max	RFC1323
Ethernet	10 Mbit	1500	16384	16384	32768	0
Ethernet	100 Mbit	1500	16384	16384	32768	0
Ethernet	Gigabit	1500	131072	65536	131072	0
Ethernet	Gigabit	9000	131072	65536[c]	262144	0
Ethernet	Gigabit	9000	262144	131072[c]	262144	1
ATM	155 Mbit	1500	16384	16384	131072	0
ATM	155 Mbit	9180	65536	65536[a]	131072	0
ATM	155 Mbit	65527	655360	655360[b]	1310720	1
FDDI	100 Mbit	4352	45046	45056	90112	0

[a] Certain combinations of tcp send and receive space will result in very low throughput (1 Mbit or less). To avoid this problem, set the tcp_sendspace to a minimum of 3 times the MTU size, or equal or larger than the receivers' tcp_recvspace.

[b] TCP only has a 16 bit value to use for its window size. This translates to a maximum window size of 65536 bytes. For adapters that have large MTU sizes (32 KB or 64 KB for example), TCP streaming performance may be very poor. This problem can be solved two ways. One way is to enable RFC1323. This option enhances TCP and allows it to overcome the 16 bit limit so that it can use a window size larger than 64KB. You can then set the tcp_recvspace to a large value like 10 times the MTU size, which will allow TCP to stream data and give good performance. The second option is to simply reduce the MTU size of the adapter. This can be done using `ifconfig at0 mtu 16384`, for example, to set ATM's MTU size to 16 KB. This will cause TCP to compute a smaller Maximum Segment Size (MSS) and with a 16 KB MTU size, it could still send 4 packets for a 64 KB window size.

[c] Performance was the same with either of these settings.

An excellent source for further network tuning information is the *AIX Performance Management Guide*.

5.6 Graphics accelerators

Graphics processing and graphical display of data is a fairly complex subject. In order to support graphic intense applications, special hardware and software is required.

Advanced graphics demand highly specialized *graphics subsystems* that consist of processing power, memory, I/O, and so on. Various different names are being used to describe these graphics subsystems; *graphics accelerator, graphics adapter* and *graphics processor card*.

Hardware and software components necessary for graphics processing can be compared to the hardware and software components in a computer.

These graphics subsystems or accelerators attach through an I/O bus into the computer system. Currently IBM uses a PCI bus for this purpose. Other buses available in the industry include PCI/X and AGP.

IBM offers a broad range of graphics accelerators to meet customers' needs. For simplification purposes one can define two "generic" categories:

- 2D only graphics accelerators
- 3D capable graphics accelerators

Accelerators that fall into the first category are exclusively used for server and business graphics with *no 3D needs* and usually only simple, entry level 2D requirements. These graphics accelerators are designed to focus on *g*raphical user interface (GUI) performance, simplicity, and cost-effectiveness. Use them if neither advanced 2D requirements, such as 24 bit colors and multiple color maps, nor any kind of 3D capability is required.

An example for this kind of graphics accelerator can be found in Section 5.6.1.1, "The GXT130P Graphics Accelerator - Entry 2D Graphics" on page 200.

The second category includes ALL graphics accelerators that offer any kind of 3D graphics capability.

Because this second category includes a wide variety of graphics accelerators, differentiated by functionality and use, we further divide this category into three classes. Before defining these classes we'd like to make some generic graphics processing statements that will help understand these classifications better.

Looking at the processing of a high-end 2D and/or 3D graphics applications, we can determine two very distinctively different "portions" of the graphics processing stream:

- Graphics geometry pipeline or engine
- Graphics rasterization pipeline or engine, also called graphics rasterizer

In a nutshell, the graphics geometry pipeline is the portion of an applications graphics processing stream that defines vertices, lines, polygons, colors, shading, rotations, zooming, etc. Basically, these are all parts of the application that define the “geometry” of the resulting 2D or 3D graphic or computer model.

The second part, the graphics rasterizer, picks up where the geometry pipeline leaves off. The rasterizer starts breaking down vertex, lines, and color definitions into pixels and their associated color values, preparing your 2D/3D graphic or computer model for display on your computer screen.

With this in mind we will define the three classes of 3D capable graphics accelerators:

Class I
Accelerators belonging to this class provide a frame buffer and multiple color maps for better 3D image display, but no further 3D graphics hardware. All graphics operations (both geometry and raster processing) are performed by the system CPU. This is possible through IBM’s unique *Soft Graphics* software algorithm, that enables the system CPU to perform graphics specific functions without requiring specialized hardware.

The implementation of IBM’s Soft Graphics is transparent to the application, because it simulates the full functionality of a 3D graphics adapter with complete support of IBM’s 3D graphics libraries (see Section 5.6.3, “Graphics APIs - The “softer side of things”” on page 204).

All advanced 2D and 3D graphics processing is done by the workstation’s CPU, so graphics performance scales directly with CPU performance.

Class II
Accelerators belonging to this class provide specialized rasterization hardware. However, these graphics accelerators still rely on the system CPU for geometry processing.

Therefore, when running 3D applications on these accelerators IBM’s *Soft Graphics* is required and application performance does scale with CPU performance.

Class III
Accelerators belonging to this class perform all 3D graphics operations, both geometry and raster processing, on the graphics adapter, using specialized and highly sophisticated hardware.

These adapters offer high speed graphics and applications that are not as dependant on CPU performance.

When using a class III graphics adapter, the graphics performance itself does *not* scale with CPU performance. However, even graphics intense applications and functions do include a noticeable amount of CPU intense code that will in fact scale with CPU capabilities and speed.

Figure 72 illustrates IBM's 3D accelerator classes as well as the graphics processing.

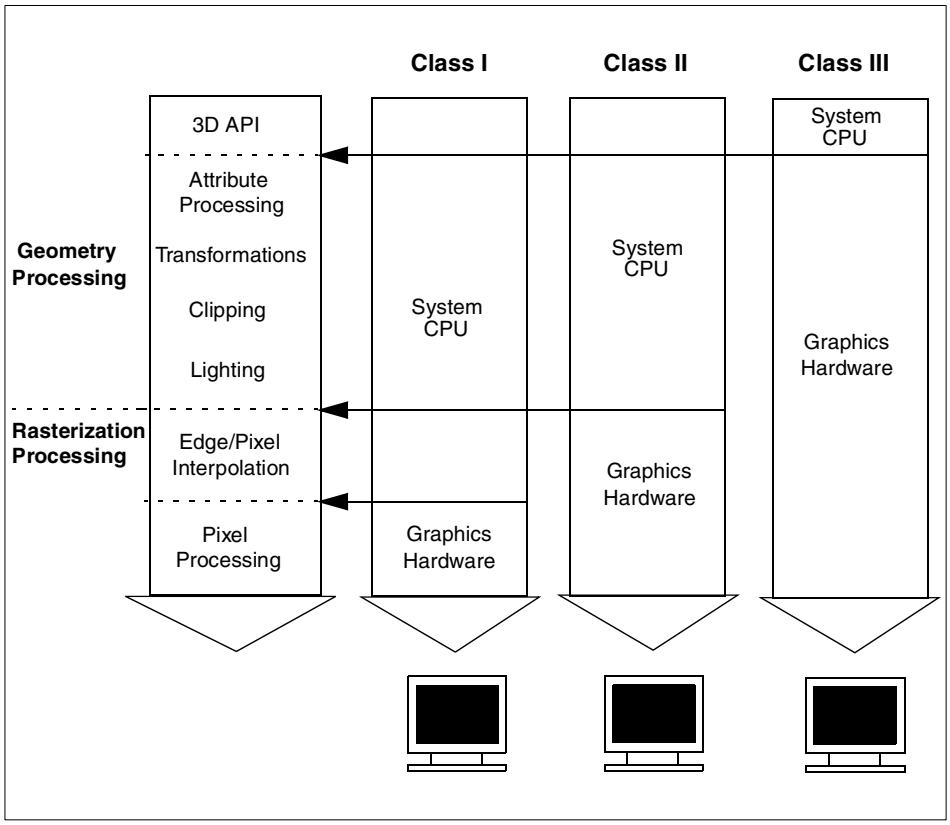


Figure 72. Graphics pipeline

5.6.1 Currently available RS/6000 graphics accelerators

IBM offers a broad range of cost-effective as well as powerful 2D and 3D graphics accelerators for the RS/6000 family. Below you will find a short description of the various adapters being offered.

5.6.1.1 The GXT130P Graphics Accelerator - Entry 2D Graphics

The GXT130P graphics accelerator is a versatile 2D only graphics accelerator for RS/6000 business and server graphics. Application possibilities include system administration, network management, operational activity monitoring, load balancing, performance analysis, capacity planning, business graphics, and web-browsing.

As many as four GXT130P accelerators can be installed per RS/6000. This is especially attractive for process control environments.

GXT130P facts and features:

- Supports 60 to 85Hz monitor refresh rates.
- Supports acceleration features such as scaling and color space conversion for video display.
- Supports screen resolutions of 640x480, 800x600, 1024x768, 1280x1024, and 1600x1200 at 8-bits per pixel.
- Satisfies the ISO 9241 standard of visual quality. Refresh rate of up to 85Hz for all resolutions.
- Supports Multisync monitors with at least a 64KHz horizontal scan capability.
- The GXT130P is a 2D graphics adapter only and does NOT support 3D soft graphics.

5.6.1.2 The GXT300P Graphics Accelerator - High-end 2D graphics

The POWER GXT300P brings exceptional performance and functionality to meet the needs of engineers and scientists requiring full-color 2D visualization. The advanced single chip graphics processor with its 32MB unified frame buffer, full 24-bit color, and four 256-entry color tables provides outstanding performance and functionality for applications such as ECAD, GIS, weather analysis, simulations, and process control. The GXT300P permits simultaneous display of up to 16.7 million colors. It supports multiple display resolution for reduced flicker, minimized reflections, and sharper complex 2D images. 3D graphics support is enabled via IBM's Soft Graphics implementation. Some of the hardware performance facts and features are:

- 3D functionality supported through Soft Graphics for applications using the OpenGL or graPHIGS APIs. (See Section 5.6.3, "Graphics APIs - The "softer side of things"" on page 204).
- Supports demanding applications that require *full 24-bit color*, multiple hardware color tables, and fast 2D rasterization.

- Features 32MB frame buffer and multiple software-selectable display resolutions up to 1600 x 1200 at 85 Hz for improved viewing characteristics.
- Up to four POWER GXT300P graphics accelerators are supported.
- The GXT300P is a 3D, Class I graphics adapter.

5.6.1.3 The GXT2000P Graphics Accelerator - Midrange 3D graphics

The GXT2000P is an entry to midrange graphics accelerator that was designed specifically to support the functional requirements of the OpenGL and graPHIGS 3D application programming interfaces (APIs), and offers many features demanded by the most widely used applications for 3D visualization. The GXT2000P is based on a highly innovative electronic design of a single chip solution for rasterization and all related functions. This design reduces cost as it increases performance. As part of the chip, the PCI interface accepts stream graphics commands and passes them to the on-chip setup block. The setup circuits calculate values for interpolating lines, smooth shaded polygons, and texture maps. An interface to the SGRAM video memory present on the adapter provides access to texture memory and frame buffer. From the setup block, the graphics elements are passed to the rasterizer section for text, line and polygon interpolation, blending, shading, and texturizing. The elements are realized as colored pixels into an SGRAM frame buffer and then double-buffered for seamless animation on displays of resolutions up to 1920 x 1200 pixels at refresh rates up to 76 Hz.

The result of this design is seamless display of the most complex 3D models and simultaneously excellent high-end 2D graphics performance.

The GXT2000P has competitive performance in standard industry benchmarks as well as application-level graphics ratings at a very attractive price.

The GXT2000P is a 2D and 3D Class II graphics accelerator. Application performance will scale with CPU speed.

5.6.1.4 The GXT3000P Graphics Accelerator - Midrange 3D graphics

The GXT3000P has a number of features that boost its texture and lighting performance well above that of previous generation IBM workstation graphics accelerators. The GXT3000P is Class II graphics, so graphics performance scales with CPU performance. However, even though the system CPU is responsible for calculating the geometry of a scene, the graphics accelerator does the lighting calculations, and texture mapping as well as the rasterization processing.

Because of the split in the work load, the GXT3000P offers faster graphics performance with faster RS/6000 workstation CPUs, such as the newly announced 44P-170 (333 Mhz, 400 Mhz, and 450 Mhz) and the 44P-270.

The GXT3000P provides a lighting and setup circuit that prepares the 3D objects for rasterization on the fly, funneling the elements to be rendered into four-way parallel rasterizers. The GXT3000P was designed to support the features of graPHIGS and OpenGL (including hardware texturing), plus many OpenGL extensions.

The four raster engines on the GXT3000P accept setup data through the APIs in the form of polygons to be rendered. The rasterization subsystem processes 3D graphics in parallel, reaching drawing speeds many times those of previous IBM graphics accelerators.

The GXT3000P graphics accelerator supports display resolutions of up to 1280x1024 and 1024x768, and refresh rates from 60 Hz to 85 Hz. It is based on the more traditional multi-chip graphics adapter design.

Even though the GXT3000P falls into the category of 3D, class II graphics, and scales with CPU performance, it does have some parts of the geometry engine implemented in hardware and is therefore a hybrid of class II and class III graphics.

Whenever an application uses extensively lighting functions, texture mapping and/or extra large models, one will see a stronger performance difference between the GXT2000P and the GXT3000P. Also, when supported by an extremely strong system CPU (e.g.: 44P-170, 450 Mhz) one will see a larger graphics performance improvement with the GXT3000P over the GXT2000P.

5.6.1.5 The GXT4000P Graphics Accelerator - Midrange 3D graphics

The GXT4000P is the follow-on and replacement product to the GXT2000P graphics adapters. As with the GXT2000P, this adapter supports high-end 2D graphics as well as entry to midrange 3D graphics at a very attractive price. This new graphics accelerator also implements the single chip rasterizer design introduced with the GXT2000P.

The rasterization chip the GXT4000P uses is an improved and performance enhanced version of the rasterization chip introduced with the GXT2000P. Additional functions include for example ones in the area of texture mapping. This new chip also operates at a higher clock speed than the original one. Performance improvements depend very much on the specific application and functions being used. Also refer to Section 6.2, “Graphics Performance Characterization (GPC) Committee” on page 225.

Even though the GXT4000P will offer performance superior to that of the GXT2000P, it will be offered at approximately the same price point, which makes it even more cost-efficient and attractive than the GXT2000P.

The GXT4000P has competitive performance in standard industry benchmarks as well as application-level graphics ratings at a very attractive price.

The GXT4000P is a 2D and 3D Class II graphics accelerator. Application performance will scale with CPU speed.

5.6.1.6 The GXT6000P Graphics Accelerator - High-end 3D graphics

The GXT6000P is the follow-on and replacement product to the GXT3000P graphics adapters. However, just as the GXT2000P and GXT4000P adapter, it follows the single chip rasterizer design, allowing for high rasterization performance at an affordable cost.

The GXT6000P is a “full blown” class III graphics adapter, featuring a single chip rasterization engine as well as a single chip geometry engine.

The rasterization chip used is identical to the rasterizer used on the GXT4000P. It is complimented by a single chip design of a hardware geometry engine.

The GXT6000P offers industry leading performance for “industry metrics” such as proCDRS as well as application specific benchmarks. It is being offered at approximately the same price point as the GXT3000P and therefore delivers industry leading high-end 3D graphics performance at an extremely affordable price.

Fore more information on performance of this or any other graphics adapter, please refer to the following URL:

`http://www.spec.org/gpc`

The GXT6000P is a 2D and 3D Class III graphics accelerator. Application performance will scale with CPU speed only for those functions and parts of the application that are CPU intense. All graphics intense functions are downloaded to the graphics adapter and processed by graphics hardware features.

5.6.2 IBM's graphics workstations

To fully round out IBM's graphics products offerings, IBM provides a number of different RS/6000 systems, which, combined with the above described graphics accelerators, make excellent graphics workstation packages.

The RS/6000 workstation family currently includes the RS/6000 43P-140, 43P-150, 44P-170, and 44P-270. These workstations offer a wide range of CPU speeds, including 32-bit, PowerPC based micro processors with solid integer performance and very affordable pricing, 64-bit Power3 based micro processors offering excellent integer performance, high-end floating point performance, excellent memory throughput, and clock rates up 450 Mhz featuring IBM's copper technology.

These workstations can be combined with the entry graphics products for affordable entry system graphics, or, in combination with the high-end 3D graphics products, can create industry leading graphics workstation offerings.

5.6.3 Graphics APIs - The "softer side of things"

In completion of IBM's graphics workstation solutions, IBM offers various different 2D and 3D graphics libraries and APIs with AIX.

All IBM graphics adapters support the X-Windows, X11 standard as well as X11 based windows managers and desktops. The current X11 version offered is X11R6 server. High-end 2D, 24-bit support is given through X11 and GXT300P, GXT2000P, GXT3000P, GXT4000P, and GXT6000P.

Additionally IBM offers two different 3D graphics APIs for advanced 3D graphics programming:

- IBM graPHIGS
- IBM OpenGL

IBM's implementation of the PHIGS standard is called graPHIGS. The graPHIGS API contains extensions beyond the PHIGS standard; those of the *proposed* PHIGS PLUS standard.

This graphics software contains a suite of advanced graphics functions for developing complex 3D applications in technical and commercial areas, including computer-aided design and manufacturing, industrial design, engineering analysis, and scientific visualization. For 3D graphics applications, the graPHIGS product provides inter operability in networked heterogeneous environments. A client/server implementation increases productivity by distributing 3D applications between workstations.

The graPHIGS API has been enhanced to include support for a multi-threaded graphics pipeline. This automatically takes advantage of SMP systems. Animations and interactive model manipulation would likely benefit the most from the performance improvements. These improvements can be observed without any changes to the application.

CATIA Version 4 is mostly written in graPHIGS, and defines the largest install base of graPHIGS based 3D graphics applications. The “automatic multi-threading” of the graPHIGS API allows CATIA and similar applications to take advantage of a multi-processor environment without any alteration to the application.

The second 3D graphics API IBM offers is IBM’s OpenGL implementation. It is based on the *OpenGL standard* as defined by the Architecture Review Board (ARB).

OpenGL is a full-featured, network- transparent Application Programming Interface (API) for developing 3D graphics applications. This suite of advanced graphics functions is ideal for developing complex 3D applications, including computer-aided design and manufacturing, industrial design, engineering analysis, and scientific visualization. OpenGL implementations can be found on various hardware and software platforms. Currently IBM supports OpenGL Version 1.3.

Most of the more recently developed applications or additions to available applications are written in OpenGL.

5.6.4 Graphics accelerator positioning

In order to do the graphics accelerator hardware sizing, one has to understand the user and application environment in which the graphics system will be used.

Many powerful combinations between workstation CPU type and graphics accelerator can be chosen. Often one can refer to the published graphics performance benchmark results and application specific benchmarks to determine the best graphics adapter/workstation package available for a particular application and end-user scenario. The following factors should be considered:

Application
This is the first and most important factor to know and understand. It is critical to know what applications the users are interested in using today as well as in the future.

Every graphics application requires a certain graphics API. Performance characteristics differ widely based on the API being used. If the user's application will be using X graphics calls, you may want to use XPC numbers, possibly in the form of a weighted average such as Xmark93 benchmark or, ideally, performance results from a similar application running on the platforms being considered. A good performance indicator for the PHIGS API is the PLB benchmark. For OpenGL performance, the OPC benchmark gives similar information. The different OPC view sets have different characteristics.

However, actual application performance comparisons are preferable and should always be considered. Customer specific benchmarking is recommended as well because end-user performance will differ based on specific models and specifications even within the same application.

Graphics accelerator hardware
Based on the information of the application and end-user requirements, determine which type and class of graphics accelerator and which CPU type will be needed.

If the application needs 2D graphics capabilities, we can choose a 2D graphics accelerator that is suitable for the application. If the application needs 3D graphics capability, the next question is what class of graphics accelerator is needed.

For most demanding 3D graphics applications, certainly, it will be appropriate to choose class II and III graphics accelerators. Certain 2D and entry 3D graphics adapters (Class I and II) can offer 3D graphics capability through software (Soft Graphics). The graphics performance of class I and II 3D graphics accelerators will scale with CPU performance.

Besides performance considerations, there are some additional characteristics that might be needed by the application, such as texture mapping, graphics resolutions, color resolutions and number and size of color tables, graphics bit planes, or the support of light sources.

Close attention needs to be given to the selection of system memory, as paging is detrimental to application performance. However, as graphics applications usually deal with an enormous amount of data, the danger of paging and the resulting performance impact is even more dramatic.

Conclusion
IBM offers a wide range of graphics accelerators, workstation CPUs, and graphics libraries. The different products can be combined in a variety of

ways offering low-cost, price performance champions as well as affordable, industry leading high-end 3D graphics workstation solutions.

5.6.5 References

For more information you can visit the following web pages:

- http://techsupport.services.ibm.com/catia/casil/casil.html
- http://www.rs6000.ibm.com/hardware/workstations

5.7 Network Station

IBM Network Station commonly known as *thin client* is designed to be easy to install, use, and manage. It operates without any internal disk storage, using supporting software and applications from associated servers. The Network Station can connect to a number of servers, including the IBM AS/400, RS/6000, S/390, or PC servers.

When the Network Station boots, it must first download a *kernel* from flash memory or one of the IBM servers using either the TFTP or FTP protocol. This is the basis for the Network Station operating system. In Network Station Version 2 code, the Network Station then mounts it's root file system from a server such as the RS/6000, or mounts it from a *flash memory* card on the Network Station itself. The decision whether to use flash memory or the NFS mounted file system from an AIX system can impact the sizing and the performance of Network Stations in AIX environments.

5.7.1 Network Station memory

Network Station is a real memory system, we must therefore be more careful while choosing the amount of memory required because virtual memory swapping cannot be used. In general, increasing the amount of memory increases the number of applications that can be run, but it does not improve the performance. If a low memory condition exists when an application is required, the application simply cannot be loaded. In critical low memory conditions, the kernel may close applications to free memory. The amount of memory required is therefore a matter of calculating the total number of applications that are expected to be run simultaneously and adding up the memory requirements of these applications. Table 22 on page 208 shows the minimum amount of memory a Network Station should have with a given application load. An application load consists of all applications currently open including any that may be in a minimized window on the Network Station work space. To determine the minimum amount of memory for your Network Station, add the memory from each of the applications that you plan to use.

Note

Results may vary based on application use, and use of extension functions.

Table 22. Minimum Network Station memory guidelines (in MB)

Network Station model	S/2200	S/2800	S/300	S/1000
Needed for a clean boot	21	18	22	23
5250 emulator after boot				
First session	4	4	4	5
Second session	1	1	1	1
3270 emulator after boot				
First session	4	4	4	4.5
Second session	1	1	1	1
VTxxx emulator after boot				
First session	3	3	3.5	4.5
Second session	2	2	2	2
Advanced Diagnostics after boot				
First session	1	1	2	2
Second session	1	1	2	2
Netscape after boot	19	19	21	21
ICA after boot	10	10	8	8
Xserver: rendering Netscape from AIX on NS	5	5	5	5
Allowance for printing a small file	5	5	5	5
Note: These guidelines recommend the minimum amount of memory needed. Memory use depends on how the Network Station is used. Plan on extensively testing your workload if you install less memory than indicated by this table. These recommendations are based on the NC operating system V2R1 PTF 8; other PTFs may have different requirements.				

5.7.2 Boot server performance

When tuning the server to support hundreds of clients, there are some techniques that can be used to ensure good performance from the network computers. Because the network computer users have their local files on the server, the user's perception of the network computer's speed is based on how fast they can launch applications and open their files.

The most effective area that can be tuned on an NC Server is the I/O and NFS performance. The most efficient way to improve performance is to put the NC applications and the user home directories on separate disk drives. In a one-drive configuration, the disk drive becomes a bottleneck. When a network computer requests a file or starts up an application, the requests are queued to one drive. Ideally, there would be one drive for the operating system and swap, another drive for the NC server software, and another drive for the user home directories. On a really busy system with more than 100 users active at a given time, it would be desirable to balance the user population across multiple drives.

To improve the NFS performance, the number of NFS daemons should be increased to eight per disk drive. To increase NFS daemons on a NC server, the nfs_server_flags variable in the /etc/rc.conf file needs to be changed. For example, on a two drive system, the line should be changed from:

```
nfs_server_flags="-u -t 4"
to
nfs_server_flags="-u -t 16"
```

One of the ways to double check how well the load is balanced on the system is to use `iostat` to ensure the drives have similar loads.

Note

For more detailed information on NFS and I/O tuning, refer to the AIX Performance Management Guide

Here are some general server setup guidelines:

- Use 100BaseT Ethernet. A 10 MB Ethernet was observed to become a bottleneck when trying to boot more than 12 Network Stations with operating system V1R3 (there is no data available for V2R1 at this point).
- Use at least 128 MB of server memory.

- Use a fast disk subsystem rather than a fast CPU.
- Use more memory (than 64 MB) rather than a fast CPU (the system will use it as buffer cache).
- Use multiple disks whenever possible, and have the operating system, the user files, and the NFS mountable partition with the Desktop on different disks.
- Use eight nfsd's per disk drive.
- Dedicate the server to its task (do not run interactive sessions or other services from it).
- Put the Netscape history file on a "RAM disk" on the Network Station itself and move it to the server only on logout and login.
  An alternative would be to put the history file on a local server that is on the same subnet as the Network Station and mount the user's home directories, or at least the user's .netscape directories from that server. This is especially important with WAN connected Network Stations that boot from flash memory and get their configuration and user space from an AIX system. The problem here is that every 10 seconds, or every time a user clicks, the history file is written back to the server. With several hundred Network Stations this can be a significant part of the network and disk traffic. For example, if one history file is 380 KB in size and there are 100 users on the network, there would be 3.8 MB of network traffic per second that is effectively not doing anything.
- Set up user directories on local servers when setting up Network Stations in a WAN environment. This can reduce the traffic load on the WAN as well as the disk usage on the server.

5.7.3 Boot performance considerations

The default boot protocol order chooses the TCP/IP protocol for downloading the kernel and applications to the Network Station. By default this is set to use NFS for downloading. Using the default settings will give a much faster download than switching to Trivial File Transfer Protocol (TFTP) download. This parameter can be changed from the IBM Network Station Setup Utilities, which are accessed on bootup by hitting the **Esc** key.

The NFS download is much faster than the TFTP download. Some of the reasons for NFS being faster are:

- During TFTP, there is a 'handshake' between each 512 byte packet. This handshake results in a latency between packets, and increased CPU utilization on the RS/6000, and the packet size is 512 bytes for TFTP.

- The default NFS packet size is 8192, which gets fragmented to the Maximum Transmission unit (MTU) size for the network. The advantage is that there is no handshake required from the network station between fragment packets. The default MTU size for Token Ring is 1492.

When using NFS there are a number of parameters that can be configured such as:

- Number of nfsd server daemons on RS/6000
- Number of NFS threads *nfs_max_threads* from the `nfso` command.
- MTU size for the network.

The NFS parameters will have an effect only if a large number of network stations are being booted. Increasing the threads may be required to reduce time-outs, but no testing was done with these parameters during the development of this redbook.

5.7.4 Application performance considerations

Since the Network Station does not contain a hard drive, the Network Station accesses the server to download the client operating system, client programs, and data. After the Network Station loads the client operating system, the Network Station displays a graphical user interface (GUI). The GUI can provide the user access to the following kinds of resources:

- 5250 emulator application
- 3270 emulator application
- Telnet application
- Web browser application (Netscape Communicator)
- Java applets or applications
- Local and remote printers

There are cases where certain applications can be run either natively or a similar application be used on a server. For example, if the user needs a 3270 emulator, the choice is to use the native 3270 emulator available on the Network Station rather than from the server.

In general, applications that are not used immediately should not be autostarted because they take up valuable memory resources even when not used but still residing in RAM. Network Station is a real memory system and code loaded in memory is not swapped out to disk. It is therefore preferable to load applications only when they are required.

Most NC users today do not dedicate a server for booting, but also use the RS/6000 as an application server. Some of the types of applications that the RS/6000 system will commonly act as a server for the Network Station are:

- ASCII text based applications that use the NC VT emulator
- Web based applications where the RS/6000 acts as the Web Server and possibly the Web Application Server
- X applications that run on the RS/6000, but display back to the NC
- X window session management with XDM or CDE

Each of these application types can also use other server functions of the RS/6000 such as database server, file server, font server, or mail server. In general, the display functions of the ASCII text based applications will have the lightest load on the server, with Web based applications being next, followed by the X based applications.

When running vt100 or Xterm sessions, a single RS/6000 server will be able to support several hundred Network Stations. The default number of PTYs in AIX is set to 256, and must be increased if this number is exceeded by the amount of Network Stations.

5.7.5 Using CDE with the Network Station

Network Station application performance also depends on the user environment. The more graphical the application and the more colors, the more network traffic and server CPU will be used. The CDE environment is a heavy consumer of CPU, memory, and network resources, and it is recommended to use the standard NC desktop when possible. Once booted, the standard NC desktop puts almost no load on the boot server except for loading NC applications such as Netscape or one of the emulators.

One hint for helping CDE performance is to turn off the blinking cursor in the dtterm. By default, a dtterm will send 10 packets a second to keep this cursor blinking. The other thing that sends a lot of traffic across the network in CDE/XDM is mouse movement. This is eliminated by using the local window manager with the standard NC desktop.

5.7.6 Performance summary

- The first time that the 3270 emulator is loaded after each boot, the load time is about 40 seconds. After this if the emulator is re-loaded, or an additional emulator session is started, the load time is only about 8 seconds.

- When booting and loading files with TFTP, the CPU of the server becomes the limiting factor.
- File load times for large files such as the kernel can be decreased by increasing adapter MTU size. However, this may effect the network performance of other users and cause fragmentation between subnets.
- Network capacity can easily become a limiting factor in a Network Station environment. The use of a 100 Mb Ethernet is generally recommended.
- Using the CDE environment takes a longer time from turning on the Network Station until the user can do productive work than any other window manager environment.
- The CDE environment has the largest amount of network traffic and places the greatest requirements on the server CPU and memory.

Chapter 6. Benchmarks

It is very important for customers as well as for computer manufacturers to compare the performance of different computers. However, it is very difficult to find an absolute measurement because, nowadays, computers are complex systems in which many components influence the overall performance of the system. System performance is especially dependent on the kind of application software that is running on the system. Moreover, benchmarks are necessarily abstract and simplified models of all those environments. For this reason, benchmarks represent a good yardstick for comparing different systems rather than a precise tool for capacity planning for a given customer application environment. No benchmark can fully characterize the performance of a system in a true production environment because:

- The behavior of benchmark applications is essentially constant on a given system. Real applications, when executed several times, almost invariably have different inputs, and consequently exhibit different behavior each time.
- Benchmarks are executed under ideal circumstances. The benchmark is typically the only application that is executed on a system dedicated to a single user. For this reason, system overheads, such as paging and context switches, are lower than in actual production use of a processor. Benchmark processors are often equipped with the latest and greatest memory and disk subsystems, features that may not exactly match a system that is of interest to a customer. In this sense, benchmarks represent the upper limit of system performance.
- Frequently, benchmarks are specified algorithmically (pencil and paper specification), or do not place restrictions on the amount of tuning that can be performed. In such situations, the application programming skill that is available to a vendor can play an extremely important role in determining the performance measured for the benchmark.

Nonetheless, some insight into the performance of a computer is provided by a benchmark:

- A computer that performs well on all benchmarks in a given class such as floating-point-intensive codes with data structures that are too large to fit into cache memory is likely to perform well in all applications that share these characteristics. Customers are usually well-informed of the characteristics of their primary computational load.
- A processor that performs well in *throughput* benchmarks (one in which many instances of many applications are executed) is likely to perform

better in a true production environment than a system that does not perform well in this context. A hardware vendor with adequate in-house programming skills can substantially improve application performance and may place these skills at a software vendor's or customer's disposal.

6.1 System Performance Evaluation Corporation (SPEC)

SPEC, the Standard Performance Evaluation Corporation, is a non-profit corporation formed to “establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers” (quoted from SPEC's bylaws). The founders of this organization believe that the user community will benefit greatly from an objective series of applications-oriented tests, which can serve as common reference points and be considered during the evaluation process. While no one benchmark can fully characterize overall system performance, the results of a variety of realistic benchmarks can give valuable insight into expected real performance.

Legally, SPEC is a non-profit corporation registered in California.

SPEC basically performs two functions:

- SPEC develops suites of benchmarks intended to measure computer performance. These suites are packaged with source code and tools and are extensively tested for portability before being released. They are available to the public for a fee covering development and administrative costs. By license agreement, SPEC members and customers agree to run and report results as specified in each benchmark suite's documentation.
- SPEC publishes news and benchmark results in *The SPEC Newsletter* and *The GPC Quarterly*. Both are available electronically through: `http://www.spec.org`

  This provides a centralized source of information for SPEC benchmark results. Both SPEC member and non-SPEC member companies may publish in the SPEC Newsletter, though there is a fee for non-members (note that results may be published elsewhere, as long as the format specified in the SPEC Run Rules and Reporting Rules is followed).

6.1.1 SPEC CPU2000

SPEC CPU2000 is a benchmark that measures computer performance for CPU-intensive computing. SPEC CPU2000 contains two sets of benchmarks. One is for measuring compute-intensive integer performance (CINT2000) and

the other is for measuring compute-intensive floating point performance (CFP2000). Both measure performance of a computer's processor, memory, and compiler.

SPEC CPU2000 requires a minimum of 256 MB of RAM to prevent the system from paging. Paging would effect the outcome of the benchmark and the aim is to only test the CPU. SPEC CPU2000 focuses on compute intensive performance, which means these benchmarks emphasize the performance of:

- The computer's processor (CPU)
- The memory architecture
- The compilers

SPEC CPU2000 is made up of two subcomponents that focus on two different types of compute intensive performance:

- CINT2000 for compute-intensive integer performance
- CFP2000 for compute-intensive floating point performance

SPEC CPU2000 is not intended to stress other system components such as disk drives, networking, and graphics, which are not included in the benchmarks even though these components may affect a system configured in a particular way.

CINT2000 and CFP2000 are based on compute-intensive applications provided as source code.

CINT2000 contains 11 applications written in C and one in C++ (252.eon) that are used as benchmarks:

Table 23. Applications for CINT2000 benchmark

Benchmark	Application area	Language Written In
164.gzip	Data compression utility	C
175.vpr	FPGA circuit placement and routing	C
176.gcc	C Compiler	C
181.mcf	Minimum cost-flow network	C
186.crafty	Chess program	C
197.parser	Natural language processing	C
252.eon	Ray tracing	C++
253.perlbmk	Perl	C
254.gap	Computational group theory	C
255.vortex	Object-oriented database	C

Benchmark	Application area	Language Written In
256.bzip2	Data compression utility	C
300.twolf	Place and route simulator	C

SPEC CFP2000 contains 14 applications (six FORTRAN77, four Fortran90, and four C) that are used as benchmarks.

Table 24. Applications for CFP2000 benchmark

Benchmark	Application Area	Language Written In
168.wupwise	Quantum chromodynamics	FORTRAN77
171.swim	Shallow water modeling	FORTRAN77
172.mgrid	Multi-grid solver in 3D potential field	FORTRAN77
173.applu	Parabolic/elliptic partial differential equations	FORTRAN77
177.mesa	3D graphics library	C
178.galgel	Fluid dynamics: analysis of oscillatory instability	Fortran90
179.art	Neural network simulation: adaptive resonance theory	C
183.equake	Finite element simulation: earthquake modeling	C
187.facerec	Computer vision: recognizes faces	Fortran90
188.ammp	Computational chemistry	C
189.lucas	Number theory: primality testing	Fortran90
191.fma3d	Finite-element crash simulation	Fortran90
200.sixtrack	Particle accelerator model	FORTRAN77
301.apsi	Solves problems regarding temperature, wind, distribution of pollutants	FORTRAN77

The numbers in the benchmarks' names serve as identifiers to distinguish programs from one another (for instance, some programs were updated from SPEC CPU95 and need to be distinguished from their previous versions).

More detailed descriptions on the benchmarks (with reference to papers, web sites, and so on) can be found in the individual benchmark directories in the SPEC benchmark tree.

6.1.1.1 Metrics and how to read them

SPEC CPU2000 incorporates run and reporting rules for baseline and optimized results for both CINT2000 and CFP2000 benchmarks.

Rates are calculated by timing from the start of the first copy of each code to the completion of the last copy of each code.

Metrics are defined as

- Base

  Base metrics refer to restricting the number of options on the compiler to try to represent ‘typical’ use. Four flags are allowed that are generally recognized as ‘safe,’ and which are used for all the benchmarks in the same language in a suite.

- Peak

  Peak metrics allow almost any compiler option, except that options may not name specific variables or functions.

- Non-rate or speed

  Non-rate or speed benchmarks execute a single program at a time, although this execution can use multiple processors if the compiler supports this.

- Rate

  Rate benchmarks execute more than one copy of each program at once to measure throughput of a homogeneously loaded system.

There are 4 metrics for CINT2000:

- SPECint2000

  This metric is produced from the geometric mean of twelve normalized ration (one for each integer benchmark) when compiled with *aggressive* optimization for each of the benchmarks.

  This is a peak metric. Almost any compiler options are allowed, except those naming specific variables or functions.

  This is a non-rate metric, so the benchmark executes one program at a time.

- SPECint_base2000

  This metric is produced from the geometric mean of twelve normalized ratios (one for each integer benchmark) when compiled with *conservative* optimization for each of the benchmarks.

This is a base metric. The compiler is limited to four options.

This is a non-rate metric so the benchmark executes one program at a time.

- SPECint_rate2000

  This metric is produced from the geometric mean of twelve normalized ratios (one for each integer benchmark) when compiled with *aggressive* optimization for each of the benchmarks.

  This is a peak metric. Almost any compiler options are allowed, except those naming specific variables or functions.

  This is a rate benchmarks, so more than one copy of each program is executed at once.

- SPECint_rate_base2000

  This metric is produced from the geometric mean of twelve normalized ratios (one for each integer benchmark) when compiled with *conservative* optimization for each of the benchmarks.

  This is a base metric. The compiler is limited to four options. It is also a rate benchmarks, so more than one copy of each program is executed at once.

There are 4 metrics for CFP2000

- SPECfp2000

  This metric is produced from the geometric mean of fourteen normalized ratios (one for each integer benchmark) when compiled with *aggressive* optimization for each of the benchmarks.

  This is a peak metric. Almost any compiler options are allowed, except those naming specific variables or functions. It is a non-rate metric, so the benchmark executes one program at a time.

- SPECfp_base2000:

  This metric is produced from the geometric mean of fourteen normalized ratios (one for each integer benchmark) when compiled with *conservative* optimization for each of the benchmarks.

  This is a base metric. The compiler is limited to four options.It is a non-rate metric, so the benchmark executes one program at a time.

- SPECfp_rate2000

  This metric is produced from the geometric mean of fourteen normalized ratios (one for each integer benchmark) when compiled with *aggressive* optimization for each of the benchmarks.

This is a peak metric. Almost any compiler options are allowed, except those naming specific variables or functions. This is a rate benchmark, so more than one copy of each program is executed at once.

- SPECfp_rate_base2000

  This metric is produced from the geometric mean of fourteen normalized ratios (one for each integer benchmark) when compiled with *conservative* optimization for each of the benchmarks.

  This is a base metric. The compiler is limited to four options. It is a rate benchmark, so more than one copy of each program is executed at once.

The ratio for each of the benchmarks is calculated using a SPEC-determined reference time and the actual run time of the benchmark.

The baseline reports are mandatory for reported results and restrict the number of compiler optimization that can be used for performance testing. Reporting of optimized results is not mandatory.

Performance measurement for system speed and throughput is provided by SPEC CPU2000. The speed at which a system completes all of the CPU2000 benchmarks is provided by SPECint2000, and the measurement of how many tasks a computer can complete in a given amount of time is provided by SPECint_rate2000.

The CINT2000 suite contains twelve application based benchmarks written in C++ and C languages that include floating point instructions. CPU intensive benchmarks are also written in Fortran 77 and Fortran 90.

The reference machine for SPEC CPU2000 is a Sun Microsystems Ultra5_10 workstation with a 300 MHz SPARC and 256 MB of memory.

6.1.2 SPEC JVM98

This benchmark measures the speed of the Java Virtual Machines (JVM) of Java bytecodes which is fundamental to overall Java performance in many application environments.

The intention of this benchmark is to measure the performance of standalone Java clients either with or without disks. Even though the benchmark will run on both Java clients and Java servers, only Java clients are benchmarked.

To run the benchmark it is required that the client be connected to a network, and have I/O and a graphics card. The performance of these resources will affect the outcome of the performance benchmark. It is not the intention of the benchmark that these resources will dominate the outcome of the results.

The benchmark requires version of 1.1 or later of the Java Virtual Machine and is applicable to 64-bit intermediate floating-point values and those using 80-bits.

The benchmark does not compare the performance of Java versus C or C++ programs.

6.1.2.1 Metrics and how to read them

Metrics are reported to SPEC in three categories of memory installed on the Java client. They are:

- 0 MB to 48 MB
- 48 MB to 256 MB
- Greater than 256 MB

The elapsed time in seconds for each benchmark on the tested system is divided into a reference time provided by SPEC so the system being benchmarked can be compared. The composite metric is calculated as a geometric mean of the individual ratios.

All the benchmark programs, except for _200_check (which is not included in the metric) are weighted equally. The programs are run several times according to the benchmarker, and the SPECjvm98 metric is calculated from the best ratios. The SPECjvm_base98 metric is calculated from the worst ratios.

The reference system is configured as follows:

- PowerPC 604, 133 MHz
- Number of CPU's: 1
- Separate Instruction (icache) and Data (dcache) caches
- L1 icache: 16 KB, 4-way associative, 32 Byte line size
- L1 dcache: 16 KB, 4-way associative, 32 Byte line size
- L2 cache: 512 KB, 1-way associative
- Separate Instruction & Data Translation Look-Aside Buffer (TLB)
  - ITLB size: 128 entry, 2-way associative
  - DTLB size: 128 entry, 2-way associativea
- Memory: 96 MB
- Disks: 2 x 2.2 GB
- Operating System: AIX 4.1.5.0
- JDK Version: Java Developer Kit (JDK*)* 1.1.4
  (Just in Time Compiler (JIT): off)
  The JIT compiles Java bytecodes to native machine code at run time.
- Uses Java executable options: -ms16m

All reported results must be validated correctly before results are published.

The following benchmark programs make up the test suite:

Table 25. Programs that make up the test suite for SPEC JVM98

Benchmark	Application
_200_check	Checks JVM and Java feature
_201_compress	A popular utility used to compress/uncompress files
_202_jess	A Java expert system shell
_209_db	A small data management program
_213_javac	The Java compiler, compiling 225,000 lines of code
_222_mpegaudio	An MPEG-3 audio stream decoder
_227_mtrt	A dual-threaded program that ray traces an image file
_228_jack	A parser generator with lexical analysis

6.1.3 SPEC SFS97

Nhfsstone (pronounced n-f-s-stone, the “h” is silent) has been replaced by SFS97.

SPEC SFS (System File Server) 2.0 is a benchmark that measures the throughput and response times of Network File Systems (NFS) file servers.

The difference between SPEC SFS and other NFS related benchmarks is that it compares NFS performance across different vendor hardware and operating systems. The benchmark was therefore written to be vendor-neutral and client-independent.

The benchmark exercises CPU, mass storage and the network with a high emphasis on I/O from an operating system perspective.

To improve the benchmark, the vendor will change the systems configuration, typically by adding hardware.

The following are hardware requirements for SPEC SFS97:

- Several servers and an appropriate number of clients.

- The server must have enough memory, disks, and networks hardware to saturate the CPU.
- A minimum of 64 MB per client. In most cases 128 MB is required.

6.1.3.1 Metrics

The metric for SPEC SFS consists of two metrics. SPECsfs97.v2 is used for NFS protocol version 2 and SPECsfs97.v3 is used for NFS protocol version 3. The throughput and overall response times are measured in both metrics.

6.1.4 SPEC web99

SPECweb99 is designed to measure the maximum simultaneous connections that a web server is able to support. The benchmark load is presented by client software on client machines networked to server machines running HTTP server software.

The benchmark runs a load generator that is multi-threaded on a number of client systems. It does a variety of GETs (to simulate the practise of rotating advertisements) and POSTs to the systems under test. The web server software is not provided by SPEC. This needs to be provided by the benchmark tester, however SPEC does provide the software for the HTTP1.0/1.1 load generator.

To establish the test, the clients needs to be configured on the network with the client software and the server needs to have the server code loaded. The connections under test are maintained at a specific maximum bit rate with a segment size intended to model conditions that will be characteristic of the Internet.

Table 26 depicts the workload percentage of the benchmarks.

Table 26. Percentage workload

Workload	Workload Percentage
Static-Get	70.00%
Dynamic-Get	12.45%
Dynamic-Get CGI	0.15%
Dynamic-Get with Cookie	12.60%
Dynamic-Post	4.80%

The workload is based on analysis of server logs from a variety of Internet servers and some smaller web sites.

The workload is based on file sizes of less that 1 KB, 1 KB to 10 KB, 10 KB to 100 KB and 100 KB to 1 MB. The benchmark directs 35 percent of its activity to files sizes of less that 1 KB, 50 percent to files sizes 1 KB to 10 KB, 14 percent to file sizes 10 KB to 100 KB and 1 percent to file sizes 100 KB to 1 MB.

6.1.5 Reference

Additional information can be located on the web at the following URL:
`http://www.spec.org`

6.2 Graphics Performance Characterization (GPC) Committee

The Graphics Performance Characterization (GPC) Committee has developed into an umbrella organization for autonomous project groups that develop new graphics benchmark methods and performance reporting procedures. In 1996, the GPC joined the SPEC organization. The GPC organization provides a forum for technical interchange, for identifying new projects, for coordinating activities of existing project groups, and for making recommendations regarding direction, priorities, and schedules for group activities.

Current GPC projects are the Application Performance Characterization (SPECapc) project, the Multimedia Benchmark Committee (SPECmedia) project, and the OpenGL Performance Characterization (SPECopc) project.

The SPECopc is set up to characterize graphics performance for computer systems running applications, not overall graphics performance. The SPECviewperf benchmark measures 3D rendering performance of systems running under OpenGL. The GLperf benchmark is designed to measure optimal performance of 2D and 3D graphics primitives.

6.2.1 SPECviewperf

SPECviewperf from the OpenGL Performance Characterization (OPC) organization is an industry standard benchmark for measuring OpenGL performance.

SPECviewperf parses command lines and data files, sets the rendering state, and converts data sets to a format that can be traversed using OpenGL rendering calls. It renders the data set for a pre-specified amount of time or number of frames with animation between frames. Finally, it outputs the results.

SPECviewperf reports performance in frames per second. Other information about the system under test - all the rendering states, the time to build display lists (if applicable), and the data set used - are also output in a standardized report.

A *benchmark* using SPECviewperf is really a single invocation of SPECviewperf with command-line options telling the SPECviewperf program which data set to read in, which texture file to use, what OpenGL primitive to use to render the data set, which attributes to apply and how frequently, whether or not to use display lists, and so on. One quickly realizes that there are an infinite number of SPECviewperf *benchmarks* (an infinite number of data sets multiplied by an almost infinite number of command-line states).

SPECviewperf can be run with different OpenGL Performance Characterization viewsets. Each viewset is comprised of different SPECviewperf tests from which a composite score is given by the weighted geometric mean. Viewsets available for comparison are:

- *ProCDRS* This viewset is intended to model the graphics performance of Parametric Technology Corporation's CDRS industrial design software.

Figure 73. ProCDRS model

- *DX* - IBM Visualization Data Explorer is a general-purpose software package for scientific data visualization and analysis.

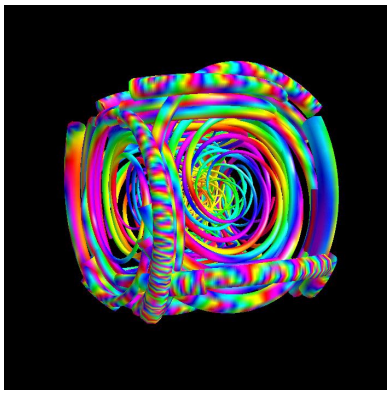


Figure 74. DX model

- *DRV* - Design Review is a 3D computer model review package specifically tailored by Intergraph for plant design models consisting of piping, equipment, and structural elements such as I-beams, HVAC ducting, and electrical raceways.

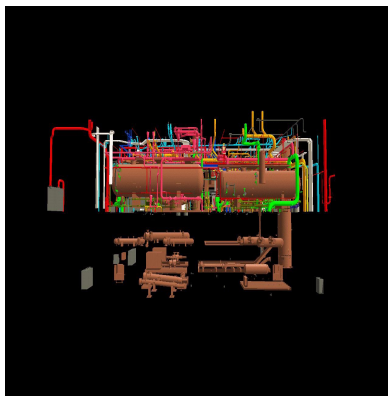


Figure 75. DRV model

- *AWadvs* - Advanced Visualizer from Alias/Wavefront is an integrated workstation-based 3D animation system that offers a comprehensive set of tools for 3D modeling, animation, rendering, image composition, and video output.

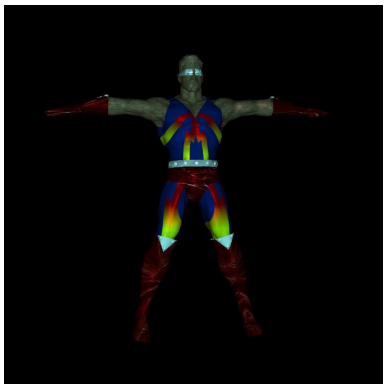


Figure 76. AWadvs model

- *Light* - The Lightscape Visualization System from Lightship Technologies, Inc. combines proprietary radiosity algorithms with a physically based lighting interface.

Figure 77. Light model

medMCAD - This is a 'generic' viewset that models the graphics performance of a range of immediate mode, MCAD applications suitable for medium sized models.

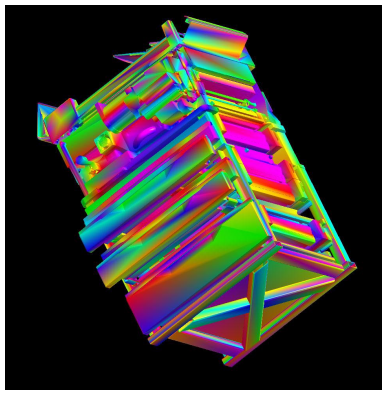


Figure 78. medMCAD model

6.2.1.1 OPC benchmark results

SPECviewperf measures performance for the following entities:

- 3D primitives, including points, lines, line_strip, line_loop, triangles, triangle_strip, triangle_fan, quads, and polygons
- Attributes per vertex, per primitive, and per frame
- Lighting
- Texture mapping
- Alpha blending
- Fogging
- Anti-aliasing
- Depth buffering

SPECviewperf is not a single-number benchmark. In order to use it to its fullest advantage, you need to relate the benchmark to their actual applications. Here are the five steps recommended for using SPECviewperf effectively:

1. Identify software code paths that are important to the application.
2. Identify the primitives used within the application.
3. Select data sets that are most appropriate to the application. The detests should reflect the level of geometry and rasterization found in the application.

4. Identify attributes and the level at which they are applied (per vertex, per primitive, or per frame).
5. Assign a weight to each path based on the percentage of time in each path and the importance of the path to the application.

6.2.1.2 References

For further information and actual benchmark results, see
`http://www.spec.org/gpc`

6.3 Transaction oriented benchmarks

The Transaction Processing Council (TPC) was founded to define transaction processing and database benchmarks. It also was charged with delivering objective and verifiable performance data to the industry.

TPC is a non-profit corporation of presently more than 40 hardware and software vendors, user organizations, and market research companies.

As TPC benchmarks are focused on the overall performance of a system in a transaction-oriented environment, the TPC numbers will be used for comparing computers in a commercial environment.

The actual benchmarks are: TPC-C, TPC-H, TPC-R, and TPC-W.

Attention

Do not compare different TPC benchmarks with each other, as the benchmarks are completely different and do not compare different major versions of the same benchmark.

For additional information, such as the versions of the published benchmarks, check the Transaction Processing Council’s homepage at `http://www.tpc.org`.

6.3.1 TPC-C

TPC Benchmark C (TPC-C) is an On-Line Transaction Processing (OLTP) workload. It is a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP application environments. It does so by exercising a breadth of system components associated with such environments, which are characterized by:

- The simultaneous execution of multiple transaction types that span a breadth of complexity
- On-line and deferred transaction execution modes
- Multiple on-line terminal sessions
- Moderate system and application execution time
- Significant disk input/output
- Transaction integrity: ACID properties (Atomicity, Consistency, Isolation, Durability)
- Non-uniform distribution of data access through primary and secondary keys
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Contention on data access and update

6.3.1.1 Metrics and how to read them

The performance metric reported by TPC-C is a *business throughput* measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of processing an order. The performance metric for this benchmark is expressed in transactions-per-minute-C (tpmC). To be compliant with the TPC-C standard, all references to tpmC results must include the tpmC rate, the associated price-per-tpmC, and the availability date of the priced configuration.

Although these specifications express implementation in terms of a relational data model with conventional locking scheme, the database may be implemented using any commercially available database management system (DBMS), database server, file system, or other data repository that provides a functionally equivalent implementation.

TPC-C uses terminology and metrics that are similar to other benchmarks, originated by the TPC or others. Such similarity in terminology does not in any way imply that TPC-C results are comparable to other benchmarks. The

only benchmark results comparable to TPC-C are other TPC-C results conformant with the same revision.

Despite the fact that this benchmark offers a rich environment that emulates many OLTP applications, this benchmark does not reflect the entire range of OLTP requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-C should not be used as a substitute for a specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

In TPC-C, throughput is defined as how many new-order transactions per minute a system generates while the system is executing four other transaction types (payment, order-status, delivery, stock-level). All five TPC-C transactions have a certain user response time requirement, with the new-order transaction response time set at five seconds. Therefore, for a 150,000 tpmC number, a system is generating 150,000 new-order transactions per minute while fulfilling the rest of the TPC-C transaction mix workload. This means, for example, that for every 10 new-order transactions, the required transaction mix will yield approximately 10 payment transactions, and one each of delivery, order-status, and stock-level

The price/performance metric is expressed in price-per-tpmC ($/tpmC).

The cost that the $/tpmC is based on is not only the cost of the computer or host machine, but encompasses all of the cost dimensions for an entire system environment the user might purchase. This cost includes communications equipment, software (transaction monitors and database software), operating system, computer systems (server and client), backup storage, and maintenance for a five year period. Therefore, if the total system cost is 7,500,000 USD and the throughput is 150,000 tpmC, the price/performance is derived by taking the price of the entire system divided by the performance (150,000 tpmC), which equals 50 USD per tpmC.

TPC-C is also a very convenient benchmark for symmetric multiprocessing (SMP) systems. The tpmC rates for SMP systems are listed by number of processors.

The performance metric for the TPC-C benchmark is expressed in throughput as measured in transactions per minute (tpmC).

6.3.1.2 Usage

TPC-C is most likely to be used to compare systems in a commercial environment.

Determining whether the TPC-C benchmark is applicable to a specific application or environment is extremely difficult. The best approach is to reach a deeper understanding of the benchmark and compare its model (user interaction, database design, database size, transaction complexity, processing requirements, storage/backup tests) with the relevant application environment. If there is a rough match, the benchmark data will probably be a useful and relevant tool for comparing different systems that may be installed in the appropriate environment.

6.3.1.3 Conclusion

Even though the benchmark offers a rich environment that emulates many OLTP applications, this benchmark does not reflect the entire range of OLTP requirements.

Benchmark results are highly dependent on workload, specific application requirements, and system design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-C should not be used as a substitute for a specific customer application benchmark when critical capacity planning and/or product evaluation decisions are contemplated.

6.3.2 TPC-D

The TPC-D benchmark is obsolete and was replaced by the TPC-H and TPC-R benchmarks.

The benchmark was replaced because some felt that it no longer measured the hardware speed and database efficiency well enough because long running queries could be answered in much shorter time by going directly to materialized views or Automatic Summary Tables (ASTs) that became available with some of the databases.

As a result of the committee's discussions it was decided to create two new benchmarks, *TPC-R* and *TPC-H*, which are identical apart from the following:

- TPC-R allows materialized views or Automatic Summary Tables (ATSs) and other features, such as many-column indexes and partitioning.
- TPC-H is much more restrictive in what is allowed; that is, it does not allow materialized views or Automatic Summary Tables (ATSs), narrow indexes, or restricted partitioning.

6.3.3 TPC-H

The TPC-H benchmark is one of the benchmarks that superseded the TPC-D benchmark (see Section 6.3.2, "TPC-D" on page 234).

TPC-H is an ad-hoc decision support benchmark that represents decision support environments. It is ad-hoc in the sense that queries are random and therefore no caching benefits will influence the benchmark results. Given this, the query times can be quite long as you cannot pre-empt the database for the query.

The benchmark consists of a suite of business related ad-hoc and concurrent data modifications. TPC-H evaluates the performance of a decision support system that performs very complex queries (more complete than OLTP transactions) on large volumes of data with a high degree of complexity.

TPC-H is composed of power and throughput runs. They should be executed under the same conditions.

- Power test: measures the raw query execution power of the system when connected with a single active user.
- Throughput test: measures the ability of the system to process the most queries in the least amount of time.

Systems today are used for both scale-up (supporting more users and therefore higher throughput) and speed-up (making a single task faster and

therefore reduce response time) of a workload. The power metric demonstrates the speedup while the throughput metrics shows the scale-up capacity of the system.

6.3.3.1 Metrics and how to read them

The performance metric reported by TPC-H is called the *TPC-H Composite Query-per-Hour.*

The TPC Benchmark H (TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates decision support systems that:

- Examine large volumes of data
- Execute queries with a high degree of complexity
- Give answers to critical business questions

TPC-H evaluates the performance of various decision support systems by the execution of sets of queries against a standard database under controlled conditions. The TPC-H queries:

- Give answers to business questions.
- Simulate generated ad-hoc queries (such as via a point and click GUI interface).
- Are far more complex than most OLTP transactions.
- Include a rich breadth of operators and selectivity constraints.
- Generate intensive activity on the part of the database server component of the system under test.
- Are executed against a database complying to specific population and scaling requirements.
- Are implemented with constraints derived from staying closely synchronized with an on-line production database.

The TPC-H operations are modeled as follows:

- The database is continuously available 24 hours a day, 7 days a week, for ad-hoc queries from multiple end users and data modifications against all tables, except possibly during infrequent (for instance, once a month) maintenance sessions.

- The TPC-H database tracks, possibly with some delay, the state of the OLTP database through on-going refresh functions that batch together a number of modifications impacting some part of the decision support database.
- Due to the world-wide nature of the business data stored in the TPC-H database, the queries and the refresh functions may be executed against the database at any time, especially in relation to each other. In addition, this mix of queries and refresh functions is subject to specific ACIDity requirements because queries and refresh functions may execute concurrently.
- To achieve the optimal compromise between performance and operational requirements, the database administrator can set, once and for all, the locking levels and the concurrent scheduling rules for queries and refresh functions.

The minimum database required to run the benchmark holds business data from 10,000 suppliers. It contains almost 10,000,000 rows representing a raw storage capacity of about 1 gigabyte. Compliant benchmark implementations may also use one of the larger permissible database populations (for example, 100 gigabytes). The performance metric reported by TPC-H is called the *TPC-H Composite Query-per-Hour.*

The performance metric reported by TPC-H is called the TPC-H Composite Query-per-Hour Performance Metric (QphH@Size), and reflects multiple aspects of the capability of the system to process queries. These aspects include the selected database size against which the queries are executed, the query processing power when queries are submitted by a single stream, and the query throughput when queries are submitted by multiple concurrent users.

The TPC-H Price/Performance metric is expressed as $/QphH@Size. To be compliant with the TPC-H standard, all references to TPC-H results for a given configuration must include all required reporting components. You must not compare different size databases.

If a system has a cost of 300,000 USD and TPC-H Price/Performance metric is 1000.00 USD per QphH@100GB, then the Price/Performance would be 300 USD (300000 / 1000).

The TPC-H database must be implemented using a commercially available database management system (DBMS) and the queries executed via an interface using dynamic SQL. The specification provides for variants of SQL,

as implementers are not required to have implemented a specific SQL standard in full.

TPC-H uses terminology and metrics that are similar to other benchmarks originated by the TPC and others. Such similarity in terminology does not in any way imply that TPC-H results are comparable to other benchmarks. The only benchmark results comparable to TPC-H are other TPC-H results compliant with the same revision.

6.3.3.2 Usage

The purpose of this benchmark is to reduce the diversity of operations found in an information analysis application while retaining the application's essential performance characteristics, namely the level of system utilization and the complexity of operations. A large number of queries of various types and complexities needs to be executed to completely manage a business analysis environment.

Many of the queries are not of primary interest for performance analysis because of the length of time the queries run, the system resources they use, and the frequency of their execution. The queries that have been selected exhibit the following characteristics:

- They have a high degree of complexity.
- They use a variety of access patterns.
- They are of an ad-hoc nature.
- They examine a large percentage of the available data.
- They all differ from each other.
- They contain query parameters that are selected at random across query executions.

These selected queries provide answers to the following classes of business analysis:

- Pricing and promotions
- Supply and demand management
- Profit and revenue management
- Customer satisfaction study
- Market share study
- Shipping management

6.3.3.3 Conclusion

Despite the fact that this benchmark offers a rich environment representative of many decision support systems, it does not reflect the entire range of decision support requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-H approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments.

Extrapolations to any other environment are not recommended. Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-H should not be used as a substitute for specific customer application benchmarking for critical capacity planning and/or product evaluation decisions.

6.3.4 TPR-R

The TPC-R benchmark is one of the benchmarks that superseded the TPC-D benchmark (see Section 6.3.2, “TPC-D” on page 234).

The TPC Benchmark R (TPC-R) is a decision support benchmark that is intended to represent an environment where queries are regularly run to generate report data and where materialized views and other means of improving times for regularly run queries might reasonably be employed. It consists of a suite of business oriented queries and concurrent data modifications. The queries and data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates decision support systems that:

- Examine large volumes of data
- Execute queries with a high degree of complexity
- Give answers to critical, frequently-asked business questions

The minimum database required to run the benchmark holds business data from 10,000 suppliers. It contains almost 10,000,000 rows representing a raw storage capacity of about 1 gigabyte. Compliant benchmark implementations may also use one of the larger permissible database populations (for example, 100 gigabytes). The performance metric reported by TPC-R is called the *TPC-R Composite Query-per-Hour.*

6.3.4.1 Metrics and how to read them

Performance Metric (QphR@Size) reflects multiple aspects of the capability of the system to process queries. These aspects include the selected database size against which the queries are executed, the query processing power when queries are submitted by a single stream, and the query throughput when queries are submitted by multiple concurrent users. The TPC-R Price/Performance metric is expressed as $/QphR@Size. To be compliant with the TPC-R standard, all references to TPC-R results for a given configuration must include all required reporting components. You must not compare databases of different size.

The TPC-R database must be implemented using a commercially available database management system (DBMS) and the queries executed via an interface using dynamic SQL. The specification provides for variants of SQL, as implementers are not required to have implemented a specific SQL standard in full. TPC-R uses terminology and metrics that are similar to other benchmarks. Such similarity in terminology does not in any way imply that TPC-R results are comparable to other benchmarks. The only benchmark results comparable to TPC-R are other TPC-R results compliant with the same revision.

6.3.4.2 Usage

The purpose of this benchmark is to reduce the diversity of operations found in an information analysis application, while retaining the application's essential performance characteristics; namely the level of system utilization and the complexity of operations. A large number of queries of various types and complexities needs to be executed to completely manage a business analysis environment. Many of the queries are not of primary interest for performance analysis because of the length of time the queries run, the system resources they use, and the frequency of their execution. The queries that have been selected exhibit the following characteristics:

- They address complex business problems.
- They use a variety of access patterns.
- They rely upon a large percentage of the available data.

- They all differ from each other.
- They contain query parameters that change across query executions.

These selected queries provide answers to the following classes of business analysis:

- Pricing and promotions
- Supply and demand management
- Profit and revenue management
- Customer satisfaction study
- Market share study
- Shipping management

6.3.4.3 Conclusion

Despite the fact that this benchmark offers a rich environment representative of many decision support systems, this benchmark does not reflect the entire range of decision support requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-R approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

Benchmark results are highly dependent on workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-R should not be used as a substitute for specific customer application benchmarking when performing critical capacity planning and product evaluation decisions.

6.3.5 TPC-W

TPC Benchmark W (TPC-W) is a transactional web benchmark. The workload is performed in a controlled internet commerce environment that simulates the activities of a business oriented transactional web server. The workload covers a range of system components associated with such environments, which are characterized by:

- Multiple on-line browser sessions
- Dynamic page generation with database access and update
- Consistent web objects
- The simultaneous execution of multiple transaction types that span a breadth of complexity
- On-line transaction execution modes
- Databases consisting of tables with a variety of sizes, attributes, and relationships
- Transaction integrity (ACID properties)
- Contention on data access and update

6.3.5.1 Metrics and how to read them

The performance metric reported by TPC-W is the average number of web interactions processed per second. An average is used because some interactions will be faster than others, as loading a home page will be faster than other interactions. Multiple web interactions are used to simulate the activity of a book store, and each interaction is subject to a response time constraint. The store size is chosen from among a set of given scale factors, which is the number of items in inventory and varies from 1,000 items to 10,000,000 items. The performance metric for this benchmark is expressed in *Web Interactions Per Second* at a tested scale factor expressed by WIPS@scale factor where scale factor is the number of items in the ITEM table. For example 123WIPS@100,000. All references to WIPS in this section mean WIPS@scale factor.

TPC-W simulates three different profiles by varying the ratio of browse to buy, primarily shopping (WIPS), browsing (WIPSb), and web-based ordering (WIPSo). All references to WIPS (WIPSb, WIPSo) results must include the primary metrics, which are the WIPS rate, the associated price per WIPS ($/WIPS), and the availability date of the priced configuration.

The following functions, if used in the benchmark, must be provided by commercially available products and be transparent to the application program:

- Multiplexing
- Routing
- Load Balancing
- Caching

The transparency requirement means that the application must not have code that directly references these functions during the measurement interval. To implement the electronic commerce function one may use commercially available products or implementation specific programs.

The electronic commerce function must include, at minimum, the following capabilities as defined in this specification:

- Secure Socket Layer (SSL)
- Shopping Cart
- Credit Card Verification
- Secure on-line payment authorization

Although these specifications express implementation in terms of a relational data model with a conventional locking scheme, the database may be implemented using any commercially available database management system (DBMS), database server, file system, or other data repository that provides a functionally equivalent implementation. The terms *table*, *row*, and *column* are used in this document only as examples of logical data structures.

TPC-W uses terminology and metrics that are similar to other benchmarks originated by the TPC or others. Such similarity in terminology does not in any way imply that TPC-W results are comparable to other benchmarks. The only benchmark results comparable to TPC-W is another TPC-W result using the same revision.

6.3.5.2 Usage

The purpose of this benchmark is to reduce the diversity of operations found in an internet commerce application while retaining the application's essential performance characteristics, namely the level of system utilization and the complexity of operations. A large number of functions have to be performed to manage an environment that supports browse and order processing. A representative set of functions are included. Many other functions are not of primary interest for performance analysis because they are proportionally small in terms of system resource utilization or in terms of frequency of execution. Although these functions are vital for a production system, they

merely create unnecessary diversity in the context of a standard benchmark and have been omitted in TPC-W.

The application portrayed by the benchmark is a book store on the internet with a customer browse and order scenario. Customers visit the company web site, the store-front, to look at products, find information, place an order, or request the status of an existing order. The majority of visitor activity is to browse the site. Some percentage of all visits result in submitting a new order. In addition to using the system as a store-front, it is also used for administration of the web site. Administration includes modification to the store-front.

6.3.5.3 Conclusion

Despite the fact that this benchmark offers a rich environment that emulates many web browsing and web-based ordering applications, this benchmark does not reflect the entire range of web server requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-W approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, systems design, and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-W should not be used as a substitute for a specific customer application benchmarking when performing critical capacity planning and/or product evaluation decisions.

6.3.5.4 References

Additional information can be located on the web at the following URL:

- `http://www.tcp.org`

Redbooks:

- *Ecina: Because No System Is an Island*, SG24-2512

6.4 ROLTP

Relative OLTP (ROLTP) is an estimate of commercial processing performance derived from an IBM analytical model. The modes simulate some of the system’s operations such as CPU, cache, and memory. However, the model does not simulate disk or network I/O operation. Although general database and operating system parameters are used, the model does not reflect specific databases, or AIX versions or releases. Unless otherwise indicated for a system, ROLTP is estimated only at the time the system is introduced and assumes the use of 32-bit applications.

The baseline reference system for the Relative OLTP Performance measurement is the RS/6000 Server Model 250, which has a Relative OLTP Performance Ratio of 1. The 250 is a 7011-250 66MHz PowerPC desktop workstation.

Although ROLTP may be used to compare estimated RS/6000 commercial processing performance, actual system performance may vary and is dependent upon many factors including system hardware configuration and software design and configuration.

All performance estimates are provided *as is*, and no warranties or guarantees are expressed or implied by IBM.

Other sources of information, including system benchmarks, should be consulted in order to evaluate the performance of a system.

6.5 LINPACK

LINPACK (LINear algebra PACKage) is the name of both a library of subroutines for linear algebra calculations and of a widely used benchmark measuring the performance of computers when solving a particular system of linear equations. The LINPACK library has been superseded by the LAPACK (Linear Algebra PACKage) library, but the name LINPACK is still applied to the

performance benchmark (even if the computations are done using the LAPACK library code). There have been four versions of the LINPACK benchmark over the years, three of which are still in use:

- LINPACK 100: solving a 100x100 system of linear equations using the unmodified source code. This is still used occasionally by some vendors, but it is an extremely small problem size for modern systems.
- LINPACK 300: a short-lived attempt at modifying the LINPACK 100 benchmark to be more suitable for supercomputers.
- LINPACK 1000 (also known as LINPACK TPP (Toward Peak Performance)): solving a 1000x1000 system of linear equations using any implementation of LU factorization. This is still a popular version of the benchmark for uniprocessors, but it is too small to be useful on parallel computers.
- LINPACK NxN (also known as LINPACK HPC (High Performance Computing) or LINPACK Parallel): solving an NxN system of linear equations using any implementation of LU factorization. The value of N can be chosen to be quite large so that many processors can be applied with good parallel scalability. This benchmark is used as the single figure of merit for the "TOP 500 List," which reports twice a year on the 500 most powerful supercomputers in the world.

6.5.1 Metrics

The LINPACK benchmarks indicate performance in the unit of millions of floating-point operations per second (MFLOPS). For MPPs, values are usually reported in giga of floating-point operations per second (GFLOPS).

6.5.2 Usage

The LINPACK 1000 and LINPACK NxN benchmarks represent the highest attainable performance level that a computer is likely to deliver. They are typically implemented by the various system vendors in ways that use caches extremely effectively and attain excellent parallel scalability. The results are difficult to compare with real-world scientific and technical applications in general without platform specific code optimization.

6.5.3 Reference

Additional information can be located on the web at the following URL:

- `http://www.netlib.org/benchmark/performance.ps`

6.6 NotesBench benchmark

The Lotus NotesBench for Domino R5 is a collection of benchmarks (workloads) for evaluating the performance of Domino servers. It measures many workloads by emulating the traffic that LAN-attached clients would generate when executing these workloads. The workloads (also called tests in the Lotus NotesBench for Domino R5 user guide) are software components that simulate the behavior of Domino workstation-to-server or server-to-server operations. They return measurements that let you evaluate server performance.

The NotesBench is only available to hardware vendors and Lotus Business Partners who have fulfilled the prerequisite NotesBench training. NotesBench is not available to customers.

6.6.1 NotesBench test

The seventeen NotesBench tests are described below. Use only these terms to describe NotesBench workloads.

1. Cluster Mail

   This test executes Notes transactions that model a cluster for mail users at sites relying on an N-way (multi-node) Domino cluster for messaging. The ClusterMail Test script models an active user sending and reading mail on a client. It contains an average of 8 minutes of waiting; therefore, an average user executes this script no more than 7 times per hour. For each iteration of the script, there are 5 documents read, 2 documents added, 2 documents deleted, 1 view scrolling operation, 1 database opened and closed, 1 view opened and closed, and some miscellaneous operations. One message is sent to the recipients approximately every 90 minutes.

2. Cluster Topology Impact

   This test models a server that initially is a standalone mail and shared discussion server, becomes a part of a cluster, and then executes server failover.

3. GroupwareB

   This test models a server for experienced Notes users who are sending large mail messages, adding documents with attachments to shared databases, and replicating changes from their local machine to the server. The test is designed to run for 6 - 8 hours.

4. Idle Usage

This test establishes an upper boundary on the number of sessions that a Notes server can support. You can use this metric to aid in setting up the other NotesBench tests.

5. Mail Routing Hub

   This test simulates a mail routing hub that routes messages to other servers (a *pure* router) and may also deliver messages to local users. The test consists of receiving messages from source driver systems and routing or delivering each message to a destination system.

6. OnlineUsers

   This test models a server for experienced Notes users who are sending large mail messages, adding file attachments to mail messages and adding documents with attachments to a shared database. The test is designed for Lotus Business Partners to benchmark their Domino Applications by determining attachment content and analyzing the impact of their Domino Application.

7. POP3 (Post Office Protocol 3)

   This test executes Notes transactions that model a server for POP3 mail users at sites that rely on the POP3 protocol for messaging. The POP3 test script models an active user retrieving and sending mail. It contains an average of 5 minutes of waiting; therefore, an average user executes this script a maximum of 12 times per hour. For each iteration of the script, there is a check and retrieval of POP3 mail messages. Each driver system (using NthIteration = 6), generates approximately (10 * Number of Addresses per message) SMTP messages per hour. Twenty percent of the users receive eighty percent of the mail messages sent.

8. Replication Hub

   This test propagates changes among a collection of other servers. The test for a replication hub consists of replicating changes to user databases. Typical replication hubs also have some amount of replication load for the Domino Directory, but that is not included in the NotesBench replication hub test.

9. Shared Discussion Database

   This test simulates a server for active users who are only performing heavy shared database operations. The test includes view operations in a shared database, navigation of unread documents, additions, and updates to documents in a shared database. It applies especially to sites that heavily utilize the collaborative features of Domino.

10. SMTP/POP3 (Simple Mail Transfer Protocol / Post Office Protocol 3)

    This test executes Notes transactions that model a server for mail users at sites that rely on SMTP and POP3 mail on the same server for communication.

11. WebBuyer

    This test simulates several interactive web browsers using an online catalog and making purchases. The process includes filling out an order form, selecting items, and completing the credit card purchase.

12. WebWalker

    This test simulates web users browsing a web site built on top of Domino. The user will peruse each link on the selected test database retrieving the *full* content of each web page, thereby providing a realistic load on the server.

13. WebMail

    This test executes transactions that model an HTTP server for WebMail users. These transactions include sending mail, retrieving mail, and deleting mail from a Web browser.

14. R5Mail

    This test executes Notes transactions that model a server for mail users at sites that rely only on mail and Calendar & Scheduling (C&S) for communication. It models an active user on a client reading and sending mail, scheduling an appointment, and sending a C&S invitation.

15. R5IMAP

    This test executes Notes transactions that model a server for mail users at sites that rely on IMAP mail for communication. This test stresses the IMAP protocol by receiving mail messages, and exercises SMTP and LDAP by sending SMTP messages to a number of recipients and performing LDAP lookups on those recipients.

16. R5MailDB

    This test models a server for active users who are only performing mail and simple shared database operations. The test includes mail-only activity plus view operations in a shared database and navigation of unread documents in a shared database. It applies especially to sites that rely primarily on mail for communication, or that have Notes users who do not yet use all Notes features.

17.Workgroup

This test models a server for experienced Notes users who send an SMTP message, perform LDAP lookups for recipients, schedule an appointment, send an invitation, delete a message, access a shared discussion database via HTTP, and update a local replica of the discussion database and replicate with the discussion database.

6.6.2 NotesBench test scenario

The NotesBench tests are described below. Use these terms only to describe NotesBench tests. The measurement scenarios are fixed, so they are *not* the same as the actual Notes user behavior.

1. Cluster Mail
   - Open mail database and a view for 5 - 10 seconds
   - Open five documents in the mail file and read each for 10 - 20 seconds
   - Categorize two of the documents
   - Compose two new mail memos / replies (taking 1 - 2 minutes to write them)
   - Stare at the view for 5 - 10 seconds and mark a few documents deleted
   - Close the view
   - Pause 1 - 5 minutes, repeat the operations above
2. Groupware B
   - Open mail database and a view for 5 - 10 seconds
   - Open five documents in the mail file and read each for 10 - 20 seconds
   - Categorize two of the documents
   - Compose two new mail memos/replies (taking 1 - 2 minutes to write them)
   - Stare at the view for 5 - 10 seconds and mark a few documents deleted
   - Close the view
   - Pause 10 - 30 seconds before switching to the local database
   - Pause at the desktop for 1 - 5 minutes
   - Open a discussion database and a view for 5 - 10 seconds
   - Page down the view two times, spending 3 - 10 seconds reading each window
   - Set the unread list to a randomly selected group of group of 30 documents
   - Open next three unread documents and read each for 10 - 30 seconds
   - Close the view
   - Pause 4 - 8 minutes, repeat the operations above
3. Idle Usage
   - Open the desired number of sessions

- Sleep two hours
- Close all opened sessions

4. Mail Routing Hub

   - Send an average size mail message with an average number of recipients
   - Pause 5 - 10 seconds, repeat the operation above

5. Online Users

   - Open mail database and a view for 5 - 10 seconds
   - Update two documents
   - Open five documents in the mail file and read each for 10 - 20 seconds
   - Categorize two of the documents
   - Compose two new mail memos / replies (taking 1 - 2 minutes to write them)
   - Stare at the view for 5 - 10 seconds and mark a few documents deleted within 1 - 5 minutes
   - Open a discussion database and a view for 5 - 10 seconds
   - Delete a couple documents
   - Page down the view two times, spending 3 - 10 seconds reading each window
   - Perform a full-text search of the view for either of two random words
   - Perform another full-text search of the view within 5 - 10 seconds
   - Set the unread list to a randomly selected group of 30 documents
   - Open next three unread documents and read each for 10 - 30 seconds
   - Add a new document with an attachment
   - Delete four documents every 2nd iteration and close the view
   - Pause at the desktop for 4 - 8 minutes, repeat the operations above

6. POP3 (Post Office Protocol 3)

   - Retrieve all pop3 mail messages
   - Pause 3 - 7 minutes, then repeat the operation above

7. Replication Hub

   - Update the local database replica of the shared discussion database
   - Change a few fields in some random notes
   - Delete a few random notes
   - Add some new notes with random data
   - Pause 5 - 10 minutes, then repeat the operations above

8. Shared Discussion Database

   - Create a discussion database
   - Reset initial note count by deleting a really large value of existing docs
   - Make sure there are enough notes in mail database (one time only)

- Open the current view and close the view
- Open a discussion database *1 and a view for 5 - 10 seconds
- Page down the view two times spending 3-10 seconds to read each window
- Set the unread list to a randomly selected group of 30 documents
- Open next three unread documents and read each for 10-30 seconds
- Delete two very old documents
- Close the view
- Pause for 4 - 8 minutes and repeat from *1

9. SMTP/POP3 (Simple Mail Transfer Protocol / Post Office Protocol 3)

- Send an smtp message
- Pause 4 - 6 minutes
- Retrieve all pop3 mail message
- Pause 4 - 6 minutes, repeat entire sequence all over again

10.WebBuyer

- Browse online catalog
- Return to home page in order to place order
- Wait 5 seconds and click the order button on the catalogs main home page
- Spend 2 minutes filling out the form
- Place the order
- Wait 4 - 8 minutes and repeat the operations above

11.WebWalker

- Spend 45 seconds browsing
- Wait 4 - 6 minutes and repeat the operation above

12.WebMail

- Send a message from the Web, taking 60 seconds to compose the message
- Wait 1 - 3 minutes
- Read the first five inbox messages, spending 1 minute on each message, deleting first
- Wait 4 - 6 minute, then repeat the operations above

13.R5Mail

- Open mail database and a view
- Read 20 documents from current location
- Wait 5-10 seconds to peruse the view
- Open five documents in the mail file and read each for 10 - 20 seconds
- Categorize two of the documents
- Send a memo (taking 1 - 2 minutes to write it)

- Add two items to the Inbox and pause 1 - 2 minutes
- Schedule an appointment and invitation
- Delete two documents and pause about a minute
- Send a response to an invitation
- Close the view
- Pause at the desktop for 5 - 13 minutes, then repeat the operations above

14.R5MailDB

- Open mail database and a view
- Read 20 documents from current location
- Wait 5 - 10 seconds to peruse the view
- Open five documents in the mail file and read each for 10 - 20 seconds
- Categorize two of the documents
- Compose two new mail memos/replies (taking 1 - 2 minutes to write them)
- Mark a few documents deleted and close the view
- Pause at the desktop for 1 - 5 minutes
- Open a discussion database and a view for 5 - 10 seconds
- Page down the view two times, spending 3 - 10 seconds reading each window
- Set the unread list to a randomly selected group of 30 documents
- Open next three unread documents and read each for 10 - 30 seconds
- Add one document to the database and close the view
- Pause 4 - 8 minutes, repeat the operations above

For more information about NotesBench scenarios, see the `http://www.notesbench.org` web site.

6.6.3 Metrics and how to read them

NotesBench generates the same throughput metric for each of its workloads (the value of the metric changes from test to test). This metric is called a NotesMark and is expressed in transactions per minute (tpm).

Along with a NotesMark value, each workload produces a value for the maximum users supported in the test and the average response time.

Thus, an audited report would look like this:

- Platform description (detailed hardware and operating system configurations)
- Results
- Analysis
- All the Notes parameter optimizations

Attention

Be aware that some vendors may be using their own, non-NotesBench workloads to define Domino server performance, and that they may be using terminology similar to NotesBench.

For example, one vendor used to claim a very high number of *power* users. The claims were not NotesBench workloads or audited NotesBench results. The vendor used mail messages as follows; 50 percent 1 K messages, 20 percent 5 K messages, and 30 percent 10 K messages. The NotesBench *power* user test (Groupware B) consists of 100 percent 532 K mail messages.

6.6.4 Usage

Notes benchmarking is designed to answer the question of *which server configuration is better*.

This enables an apples-to-apples comparison for a particular Domino version (for instance, AIX with 256 MB RAM versus AIX with 512 MB RAM) or across different server versions.

Important

NotesBench has been a volatile program since its inception. NotesBench can change with each build of the Notes product. The workloads can change, and historically, they have *lightened*, meaning later tests may produce better results than earlier tests.

You must always know the release (and even the build number for pre-audited results) of Domino/NotesBench used for particular runs.

The various NotesBench workloads have been designed to emulate different user behavior, from new Notes users (Mail), to moderate users (MailDB), to power users (Groupware B), to Web users.

However, it would be inappropriate and inaccurate to use these as exact capacity and performance results within a production setting. Technology factors (including CPU types, OS versions and peripherals) change too often to give a valid long-term answer. Therefore, the published results should be used primarily as baselines (from which adjustments are made up or down) for a customer's unique environment:

- Adjust the user count up or down by some ratio, based on customer's end user usage behavior.
- Use published *throughput* data (such as the replication hub workload) to determine if the bytes transferred mirror the customer's replication behavior.
- Use the hardware configurations documented in the benchmark reports as guides for selecting and optimizing hardware on proposed systems.
- Monitor the changes made to default settings (such as operating system settings and Domino server settings) to determine optimization settings for specific workload levels.
- Monitor network topology setups documented in tests to determine if unique components are being used for testing (such as PCI network interface cards (NICs) or dual NICs).

Furthermore, Lotus Consulting is able to provide professional assistance on using NotesBench and other tools within a customer's environment.

6.6.5 Conclusion

The Groupware B workload seems to be the most realistic workload for large organizations implementing Lotus Notes because it deals with both intensive mail and shared databases. Otherwise, depending on the estimated Notes usage, you should refer to the closest NotesBench workload. You should use the NotesBench only for sizing capacity of the Domino server because it was developed as a tool for hardware benchmarking. For sizing your Domino server, see Section 7.6, "Lotus Domino Server sizing" on page 325. You should not use the NotesBench for sizing a network.

6.6.6 References

Further details or numbers may be found at the followings documents:

- *Lotus Domino R5 on IBM RS/6000: Installation, Customization, and Administration*, SG24-5138
- *NotesBench Disclosure Report for the IBM RS/6000 Enterprise Server S80 with Lotus Domino R5.0.2 on AIX V4.3.3* from `http://www.notesbench.org`
- *Domino Capacity Planning* from `http://www.lotus.com/performance`
- `http://www.notesbench.org`
- `http://www.rs6000.ibm.com/resource/technology/notesSPcfg`

Chapter 7. Sizing

Any sizing task implies complex and error-prone procedures. There are many factors which make a sizing job inadequate or deviate from the goal. A few of the major factors are:

- **The speed at which technologies change continues to accelerate**
  Technology develops more quickly each year. Postponing a purchasing decision because better systems will be available next year is not an option. In almost all cases, configuring the latest and state-of-the-art technology is a good choice as each generation of systems improve the price/performance ratio and capacity features.
- **Every customer has his particular requirements**
  Following system sizing and installation models that others have made is not an easy job. Many organizations have common features at the moment that lead them to choose a certain technology, but there are many different factors that differentiate their sizing models:
  - Quantity and distribution of the application users
  - Required response times for end users and the administrative staff
  - Security, availability, and integrity levels for both applications and data
  - Application nature, data distribution, and sharing models
  - User-authorized use of system resources (such as applications, personal storage space, printer spoolers, mail, and backup tools)
  - Types of concurrent applications running on the same server
  - Quantity and distribution of data in the system
  - Type and version of enablers (like DBMS) and applications
- **Data for sizing is often insufficient**
  It is usually hard to find enough data concerning the resource consumption of an application (like processor, memory, or I/O). And even if you had this kind of information, there is no accurate model that is able to identify the most appropriate RS/6000 system(s) other than simulation.
- **New data processing needs may arise**
  Plans for a data processing system normally are performed following established requirements. However, new needs may appear during the installation or production phases. It is then necessary to provide additional resources.
- **Predicting the future is not easy**
  The organization's strategic plans, which form the basis of technological

projections, are affected by internal and market-driven decisions. Those modifications imply in many cases that a well-planned project or one in the developing stage becomes obsolete before completion. The sizing can also reveal itself to be under evaluated or the architecture unscalable.

Last, but not least, the goal is to satisfy all the customer's expectations and needs in the most reasonable way. In a well-sized system, the configured equipment is used adequately, and additional components are configured to support peak levels or marginal growth.

7.1 General sizing concepts

The goal of this part is to give a few general guidelines relevant for almost any type of sizing. We will review some of the most important concepts that influence the sizing procedure.

7.1.1 Guidelines

It is important to choose your system carefully so that you have scalability and expansion capacities left for the future. This should include:

- Processor
  Choose a processor for the peak load plus a safety margin. Make sure you have upgrade options available for a system that is likely to grow.
- Memory
  Estimate the amount for all of your applications. Memory should not be configured near its upper limit so that you can add more memory in the future.
- Storage (such as disk, tape)
  Estimate the amount of raw data to decide how many disks are needed and choose a tape system for backup data.
- Number of slots
  Try to have a few slots available for extra adapters to, for example, support additional disk drives in the future.
- Network
  Choose adequate network adapters so that this does not become a bottleneck.

  Communication is often forgotten in sizing procedures, but it is a major issue, especially as network bandwidth is strained by more-demanding client use and applications. Like users or applications, communication demands special performance tools and capacity planning. Network

hardware and related issues are documented in Section 5.5, “LAN/WAN Adapters” on page 187.

Whenever possible, try to split the different services available on the network to several servers.

It is generally known that the more complex the system, the more likely it is to perform badly due to a bad component that is not properly analyzed, sized, and configured.

As a general rule, real workload simulations are obviously more accurate than rules-of-thumb sizing or even industry standard benchmarks because even the latter does not represent a genuine customer application.

A complete model for hardware resource sizing must consider at least the following three factors:

- Server(s) sizing
- Client workstations or personal computer sizing
- Communication adapters and bandwidth sizing

Attention

Keep in mind that the load generated on a system does not only come from application-related tasks but also from administrative and/or system-management subsystems.

For instance, a large backup can require an important part of the CPU, I/O, and, for remote backups, network bandwidth.

7.1.2 Concepts

In this section, we discuss some points of view for general sizing concepts.

7.1.2.1 Workload

Workload can have different meanings, depending on the context. In benchmarks, workload is defined as a set of test executions of a particular application. Here, the term workload is used as the load a system has to bear due to one or more applications.

The nature of the typical workload must be known before any sizing plan can be implemented. Workloads can be categorized as follows:

- **Interactive**
  The user has a live session in the system, using a text-based or graphical terminal or a client computer to control the work.
- **Background**
  This kind of workload requires the users to be logged into the system until the job ends, but it does not require any interaction with the user. A background workload is usually made of processes and subsystems servicing interactive user processes, like printer spool daemons, TCP/IP or SNA daemons, or database server processes.
- **Batch**
  It is basically like the background workload, but the user does not have to be logged into the system to allow batch processes to run. These kinds of jobs do not need to interact with the user for input or output. The usual work done in batch is database file updates, reports created in text files, packets sent over the net, and so on.

A good recommendation is to avoid mixtures of different heterogeneous types of workloads. This should be done in order to help the performance-management and capacity-planning tasks, and also because of possible resource contention.

Traditional workload
Workload distribution throughout the day should be studied, as shown in Figure 79 on page 259, to be able to estimate the peak workloads.

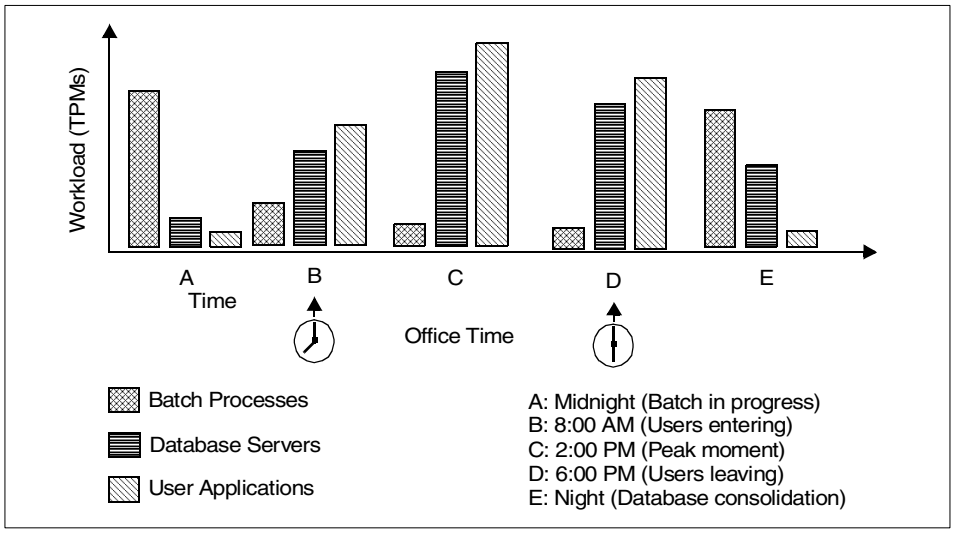


Figure 79. Example of Heterogeneous Workload Type in a Traditional System

A system should be configured with spare capacity to support both the typical workload and peak levels with acceptable response times.

Care must be taken with averages. For example, an average of 50 percent during a 4 hour period could hide peaks and troughs between 10 percent and 90 percent. The sizing should be for the 90 percent peak so that the machine is responsive at these times.

Also, note that the system might have extra peaks during the week, month, or year. These peaks need to be included in the system size.

For batch workloads without users (for example at night) the system must be sized to complete the tasks within the batch run period rather than response times.

e-Business server / large web server workload
e-business applications for the Internet are growing rapidly in popularity. For e-business applications, 24 hours and 365 days system may be required depending on the situation. For systems that are used worldwide there seems to be no peak time, since someone who lives in another country can access your e-business site during unusual hours for your timezone.

There may be peaks due to countries that have high numbers of Internet users like the USA and Europe waking up or in the evenings. But the very volatile nature of the Internet means the peaks are largely unpredictable.

Also, in order to prevent Internet users from going to other web sites, over-sizing is recommended.

A typical e-business site like a large virtual book shop needs a large web server that will be generally accessed all day long. Such an e-business application is called B2C (Business to Consumer) electronic commerce.

There are other types of systems that need 24 hours availability, like a very large global web server such as the Wimbledon Tennis Championship Official Web Site, the Olympic Official Web Site, and the Georgia Masters Golf Tournament Official Web Site.

As shown in Figure 80, very large global web servers have various peak times generated by the access pattern of users in different countries around the world.

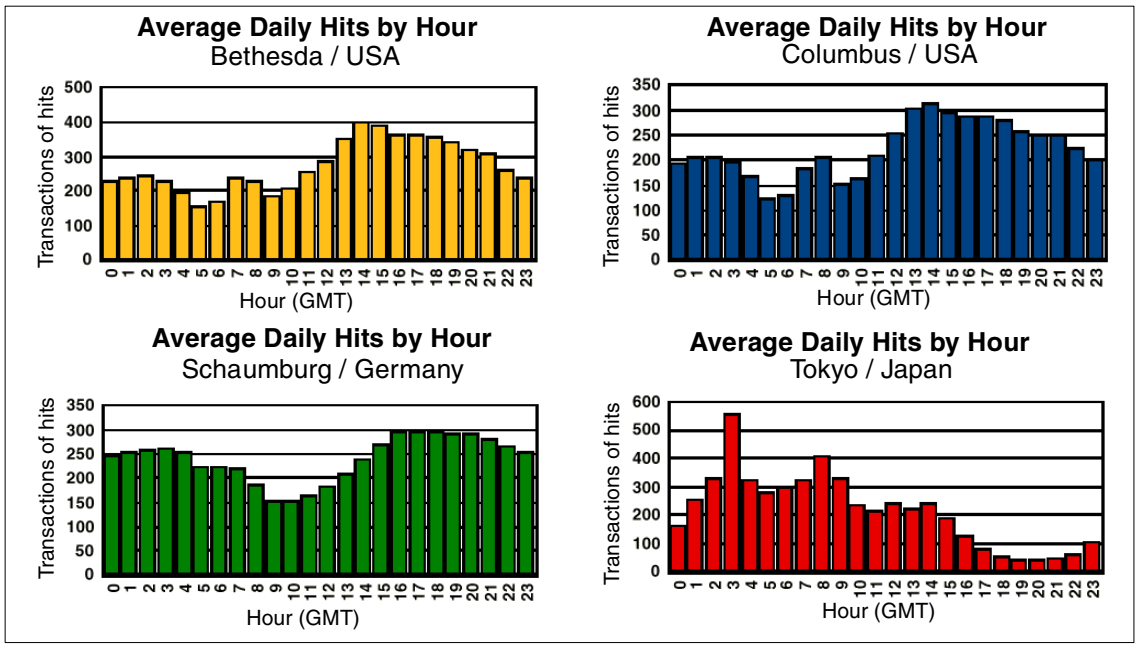


Figure 80. Nagano Olympic web servers daily hits

7.1.2.2 Applications

One of the most important tasks to do before sizing RS/6000 systems as servers or workstations is to know what kind of applications will run on the system.

The applications may be categorized depending on their program construction and interaction as follows:

- Monolithic
- Modular
- Intensive computing
- Interactive
- Batch
- Clients of a local or remote server
- Servers of local or remote clients

Monolithic applications are able to run by themselves. They need no external interaction (for example, complete application programs written and compiled in C or Cobol).

Modular applications are those that group a set of specific programs loaded one at a time. Each program belongs to a complete module, and a module is in charge of a specific end-user task. Such applications include data entry and report modules operating over the same data but separated in different programs and accessed through a menu.

Applications can be of more than one type at a time. For instance, an intensive computing application may be a monolithic program.

Table 27 gives an idea on the resource requirements of each application type

Table 27. Application types and system loads.

APPLICATION TYPE	PROCESSOR Demand	MEMORY Demand	I/O Demand	LAN TRAFFIC	MAX NUMBER IN THE SYSTEM
Monolithic	Application Dependent	**HIGH**	Application Dependent	Application Dependent	Application Dependent
Modular	Application Dependent	**LOW/MEDIUM**	Application Dependent	Application Dependent	Application Dependent
Intensive Computing	**HIGH**	Application Dependent	**LOW**	**LOW**	**LOW**
Interactive	**LOW/MEDIUM**	Application Dependent	**LOW/MEDIUM**	**LOW/MEDIUM**	**MEDIUM/HIGH**
Batch	**MEDIUM/HIGH**	Application Dependent	**MEDIUM/HIGH**	Application Dependent	**LOW**
Client	**LOW/MEDIUM**	**LOW/MEDIUM**	**LOW**	**LOW/MEDIUM**	**MEDIUM/HIGH**
Server	**MEDIUM/HIGH**	**MEDIUM/HIGH**	**HIGH**	**MEDIUM/HIGH**	**LOW**

Mixing different kinds of applications together on the same system can result in contention (keeping some resources out of the reach of other ones); this

should be avoided unless the AIX Workload Management feature is used to prioritize the use of system resources.

The alternative is a number of servers with special functions, like file servers or database servers. This approach can make sizing far simpler, and allows tuning for specific workloads.

7.1.2.3 Concurrent user

It is very important for sizing to understand the difference between a user who is authorized to log on to a system, a connected user, and a concurrent user who is actually logged in and using the application. When sizing the peak workload, you should focus on the maximum number of concurrent users, not the total number of users. When sizing disk and memory capacities, you should focus on the maximum number of connected users.

When speaking of concurrent users, you have to assume they have the same working habits so that they form a homogeneous population generating a steady workload. Otherwise, you may have to break users into various groups and size their workload individually, and then add up the totals for the system as a whole.

7.1.2.4 Response time

The primary response time question is, “What is the required application response time for a typical user of profile X?” If the sizing is for a machine upgrade the question may be also be, “What increase in system specification is needed to reduce the response time for user X to the required response time?”

7.1.2.5 Queuing concept

Internal physical resources of a computer are managed with request queues (processor, I/O adapters, physical disk drives). The problem arises when slow resources have many requests and the rate at which elements are queued becomes greater than the rate at which the requests are being dispatched. When this happens, the queue grows following an exponential model.

Response time soars every time the system utilization approaches 100 percent, and it could be necessary to reduce the active jobs to allow the system to come back to a steady state (see Figure 81 on page 263).

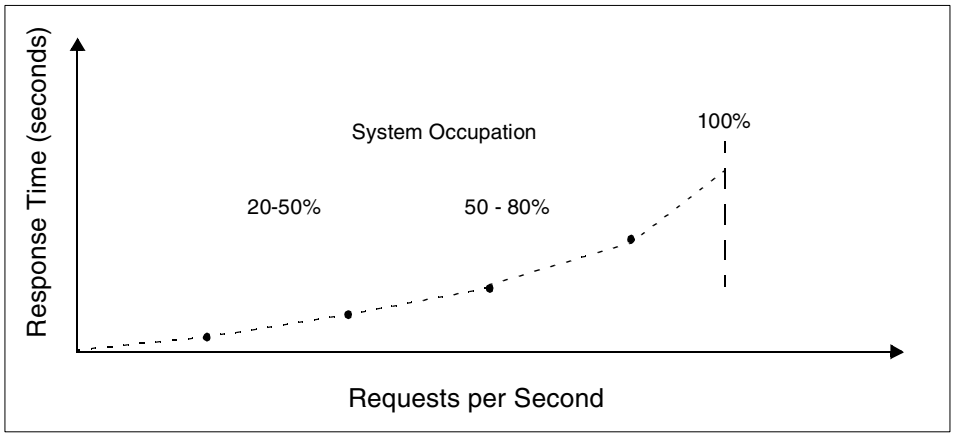


Figure 81. Response times and system occupation

As a general rule, you should not configure a system to use more than 75 percent of its processor capacity, or 40 percent of its disk throughput abilities. This provides surplus capacity to maintain response time during peak moments of workload.

7.1.2.6 Component speed-up

Overall application performance relies not only on processor speed but also on every component of the system architecture (like memory, disk drives or buses).

The global time an application needs to be executed is obviously the sum of all of the time spent in each computer component.

Effective_Application_Time = Sum(Processor time, Memory time, Bus time I/O time, Disk Drive time, LAN time)

Figure 82. Effective Application Time

Increasing the speed of one component may have no influence whatsoever on application performance. It depends on the dependence level the application has with the component.

Let us refer to each of the *Effective_Application_Time* component fractions by the following abbreviations:

PR = Processor time fraction
ME = Memory time fraction

BU = Bus time fraction
IO = I/O time fraction
DD = Disk Drive time fraction
NE= Network time fraction

The sum of these fractions is 1 (as in percentages).

Now, for example, if we increase the network speed by a factor of K, the time the application will spend in the network component will decrease by a factor of (1/K). This implies that the total application time will decrease by a factor of (1/S), S being the application speed-up (K>S>1). Figure 83 shows this law, known as Amdahl's Law for speed-up. Another example of this law is demonstrated in Section 5.3.1, "Performance view" on page 149.

Before Network $1 = PR + ME + BU + IO + DD + NE$

After Network $(1 / S) = PR + ME + BU + IO + DD + (NE / K)$

$$\Rightarrow \quad S = \frac{1}{PR + ME + BU + IO + DD + (NE / K)}$$

$$\Rightarrow \quad S = \frac{1}{(1 - NE) + (NE / K)}$$

The general form $$EAS = \frac{1}{(1 - FFM) + (FFM / SFM)}$$

Figure 83. Example of Amdahl's Law

Where:

EAS Effective application speed-up
FFM Fraction of the time the application spends in the improved component
SFM Speed-up of the improved component

Consider two applications in the same system, the first CPU-bound and the second I/O-bound. Consider the fraction the applications are utilizing the CPU as 80 percent and 20 percent, respectively. If the CPU is improved by a factor of 2, the speed-up values for both applications are 1.67 and 1.11, respectively.

7.1.2.7 Processor speed-up

An RS/6000 system model typically differentiates itself from its predecessor with its improved processor speed, cache speed, cache size, and bus bandwidth. You should not compare two different systems only using their processor speed; you also need to consider disk, I/O, and communication subsystem performance.

Figure 84 shows how a workload derived from a lab-tested end-user application performs in different RS/6000 systems belonging to the same family.

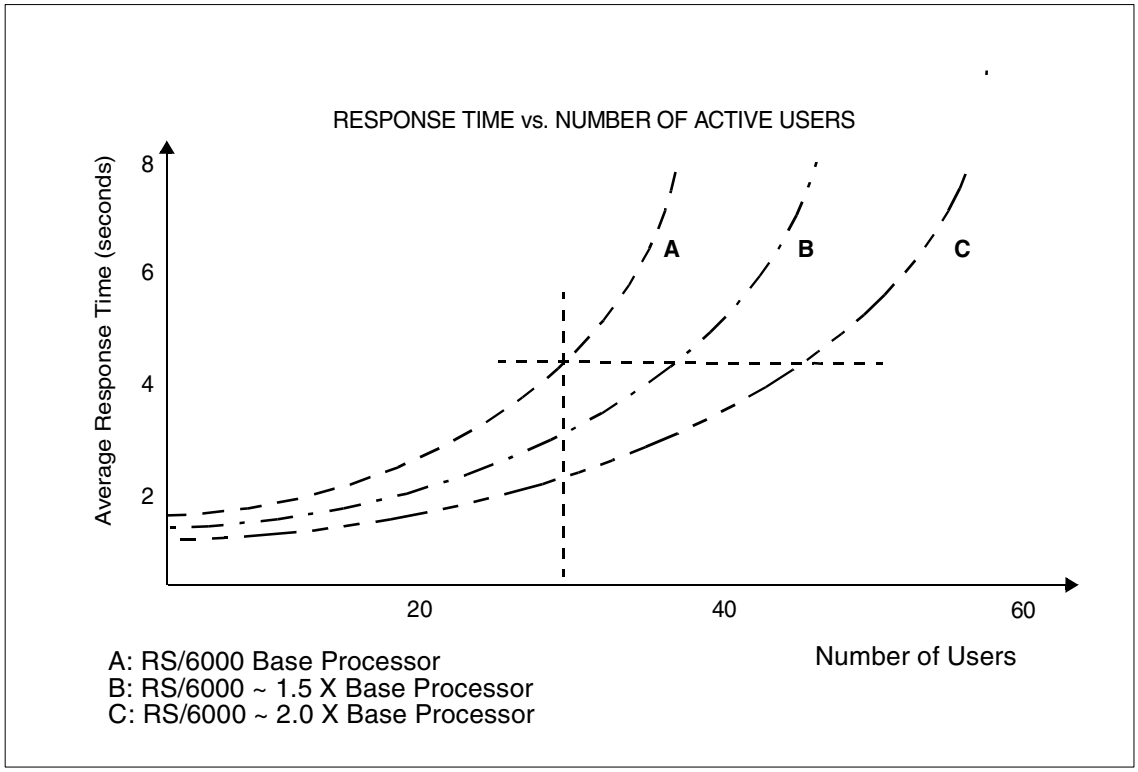


Figure 84. Response Time vs. Number of Active Users

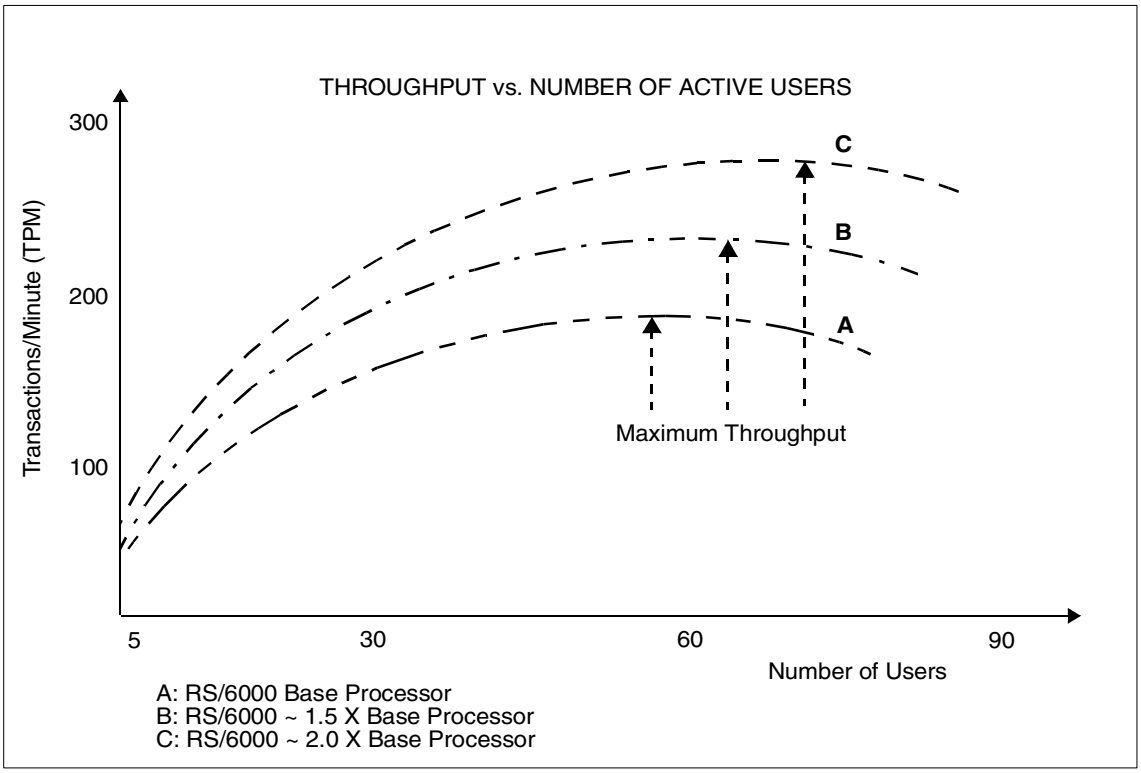


Figure 85. Throughput vs. Number of Active Users

Figure 85 shows the throughput point of view for the same example. Note that when the maximum throughput is reached in these systems, adding more users results in response-time degradation due to lock contention.

7.1.3 Using AIX Workload Manager (WLM)

WLM is an operating system feature introduced in AIX V4.3.3. It is designed to control CPU allocation, physical memory resources, and disk I/O. WLM is mostly used for large SMP systems in server consolidation environments. WLM can also be used in uni-processor systems or SP nodes.

WLM provides you with the ability to control the behavior of the AIX scheduler, the VMM, and the disk I/O requests that are sent to the device driver. You can separate users of different workloads of the system such as day time interactive or low CPU usage jobs, batch or high CPU consuming jobs. For example, a user who is executing a low CPU consuming interactive

job is protected from a high CPU consuming job such as data loading jobs of RDBMS.

If you use Net.Commerce or WebSphere on a single machine, WLM can help you. For example, WebSphere has components such as an HTTP server, Application Server, and Business Application Database. See Figure 86. If there is heavy workload on the database, a user accessing some static html files has to wait for a long time. In this case, WLM can control each workload of Web Server, Business Logic, and Database.

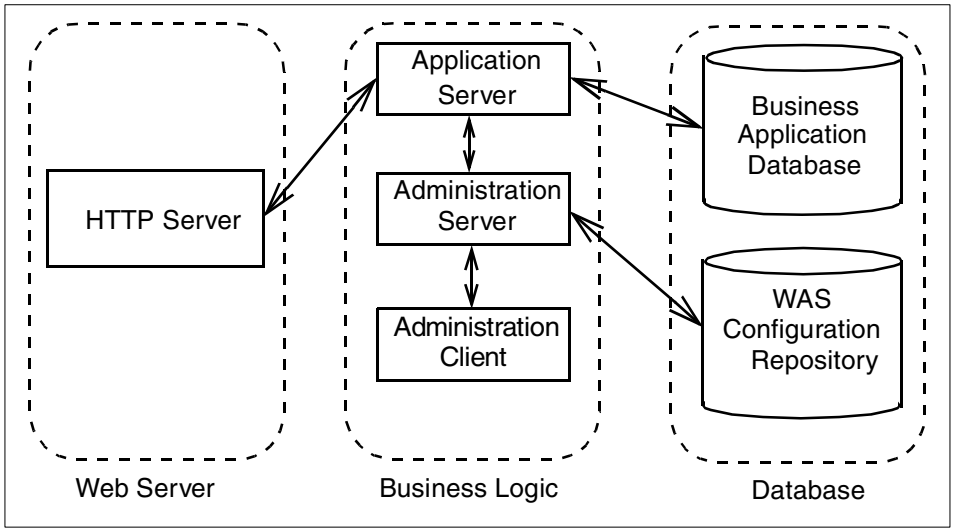


Figure 86. WebSphere Application Server (WAS) topology

7.1.3.1 Sizing in a WLM environment

Various workloads can be placed on a single large server to simplify system administration and share resources. WLM can then be used to allocate resources to these workloads. These resource allocations can be adjusted while running to maximize the response time or throughput of important workloads.

You have to study and anticipate the behavior of your applications. It is important to understand the user base and their computing needs.

Sizing steps

1. You need to size each application with general sizing methods. You need to size the processor, memory, network, and storage. Then sum the requirements to find the totals.

2. Add the estimate CPU, memory, and storage requirements for the OS. Then add a buffer so the machine can cope with workload peaks.
3. Define the allocation of processor, memory, and disk I/O workloads that you sized. Consider the allocation with the application behavior you studied. The WLM does not control network bandwidth, so you should secure enough network bandwidth for all applications.
4. Configure WLM.

7.1.4 Resources

- *Server Consolidation on RS/6000*, SG24-5507
- *AIX 4.3 System Management Concepts: Operating System and Devices*, SC23-4311
- *WebSphere V3 Performance Tuning Guide*, SG24-5657
- *AIX 5L Workload Manager (WLM),* SG24-5977

7.2 Multiuser system sizing

Multiuser configurations can serve a wide variety of purposes and experience a wide spectrum of user loads. As its name says, and by the traditional categorization, a multiuser environment is defined as a central system that is running applications and controlling the user interface directly.

That is, users are directly typing commands at a keyboard that the central system is processing.

This can be:

- Directly attached terminals
- Telnet sessions
- X Windows where the application is running on the central server and only displaying on the client machine
- The directly attached graphics console

A multiuser system can also act as a server for databases, files, or other classes of client/server computing environments. This section is intended to provide some guidelines about the multiuser issue (a single system with many users). Large multiuser configurations may have users operating a variety of applications such as databases, program development, or office automation in addition to the full range of AIX commands.

As described above, the hardware to support the end-user workload depends on the application's nature, quantity of data, number of users, and concurrence. The RS/6000 family of products offers a wide variety of options from entry desktop servers up to high-end deskside ones based on SMP technology.

7.2.1 Multiuser environment

Configuring multiuser environments depends on both application's nature and user habits. In order to understand multiuser configurations, it is necessary to understand software components in a multiuser environment. Let us use a top-down approach to review software components:

User applications
There are two categories for multiuser systems, depending on how the application programs share data resources:

- **Independent**
  AIX commands and other self-contained programs.
- **Local clients**
  AIX programs of this type use communication methods to share data inside the same machine. All of the applications belonging to this type have a similar architecture to that of the DBMS environments. The set of server programs running between the operating system and end-user programs is called the application enabler level.

Application enablers
They set the software level providing data management services to applications, such as a DBMS server.

Communication
The communication protocols and utilities are intended to allow end users to open sessions on the system.

System programs
All commands and high-level programs belonging to the operating system, such as compilers, SMIT, file utilities, and so on.

Support tools
All utilities necessary to maintain the system, support end-user tasks, and keep the system performing at the desirable response times.

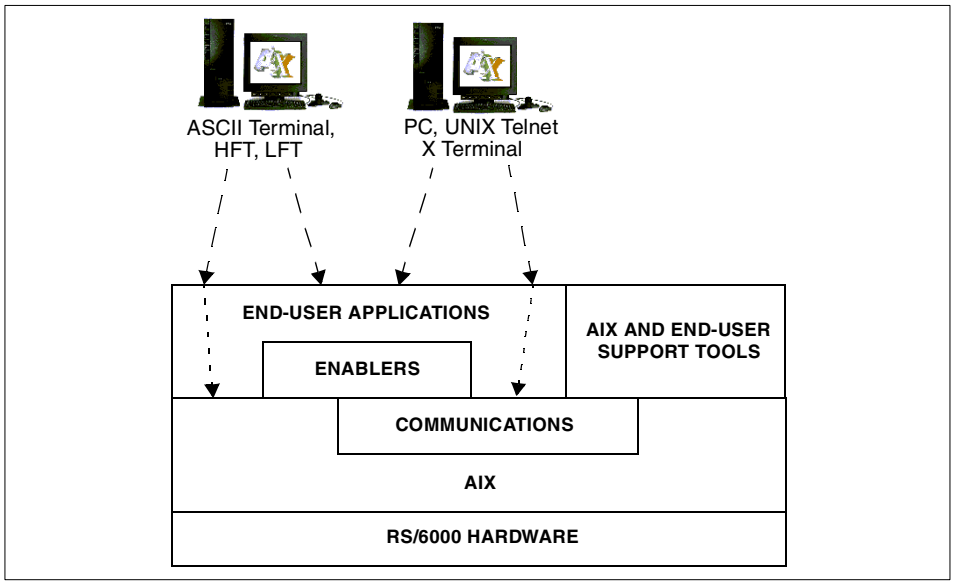


Figure 87. End user applications support

It is very important to know the specific resource needs for the type of applications you want to install on your system. Independent programs demand high levels of operating system attention and system resources when the quantity of users (active and passive) is high. Independent program environments also suffer of high lock contention when many users access shared files concurrently. So if you want to have a lot of users (70 or more) working together over the same resources, it is better to think about DBMS or a similar environments rather than independent programs.

Applications based on independent programs follow a near-linear tendency in memory consumption. This is not the same for local server-based applications. Usually these applications require some fixed amount of memory plus some memory for each user. After a certain number of users, the server-based approach consumes less memory than applications based on independent programs.

Using monitors or local servers in multiuser configurations helps to optimize the use of resources in the system. Many DBMS sizing considerations apply to applications based on local servers. There are architectural differences between the two implementations:

- **2N servers**

  Servers based on *2N* implementations create a server process for each local client, so an end-user process is really representing two processes, and the optimization in system resources is minimal.

- **Multi threaded server**

  Multi threaded servers are designed to avoid the extra expense of managing many processes. Rather than having one server process for each client, there are a few specialized processes with multi threaded capabilities so that they service requests from multiple clients.

Session support
An end-user session basically is composed of a light shell program with a prompt (where the user can start other programs or system commands) and a set of variables defining his environment. Shell demands for operating system resources are minimal. For telnet-based sessions, *telnetd* daemon resource consumption should be considered. In large systems supporting many users, the overhead caused by the asynchronous adapter devices drivers and *telnetd* daemons must be considered.

Programs that use graphical presentations through X11 libraries and the X Windows capability consume more memory and a bit more processor power than their respective ASCII versions.

7.2.2 Workload balancing

It is inevitable to have different workloads on a multiuser system. In those cases where interactive end-user, batch, and computation-intensive processes fight for the system resources, it is necessary to perform a workload separation (organization). Using the AIX performance monitoring tools (as described in Chapter 8, “Performance tools” on page 333), you can collect all the data you need to determine resource consumption for each application.

If your workload has, for example, batch and interactive processes, you should consider an SMP system to perform the workload separation. A batch process, which is not threaded, can use only one CPU at a time. This means the other CPUs are available for interactive workloads and the batch workload does not dominate the CPU usage. In some situations, it is convenient to use

WLM (AIX Workload Manager) for different workloads. Consider this possibility when your response times are critical for batch or scientific applications and you do not want to see end users' work affected. See Section 7.1.3, "Using AIX Workload Manager (WLM)" on page 266 for more information about WLM.

7.2.3 General sizing considerations

Given the diversity of applications, implementations, workloads, users, and requirements, it is necessary to have a detailed, in-depth analysis of an installation like the one you are sizing. Most of the time software vendors are able to give you some guidelines based on previous installations. When sizing a server, you should also consider the resource consumption of administrative tasks such as backup and recovery, problem determination, performance analysis tasks, software maintenance, and so on.

7.2.3.1 Processor

To maintain good response times the processor should have enough capacity to handle momentary peaks in user demand, such as lots of people hit the Return/Commit/Action key at the same time. In systems requiring very fast response times independent from the workload, you should configure your machine so that the processor will have enough idle time to handle the task (idle time > 40 percent). In systems where the average response time is the target, you should consider about 20 - 40 percent idle time. Only for systems in which user response time is not important should the idle time be planned to be less than 20 percent such as a multi-user system that simply starts batch or compute intensive applications that are expected to take minutes or hours to complete.

7.2.3.2 Memory

The required amount of memory in multiuser systems is dependent on the number of users, the type of applications, the type of enabler, and the quantity of data the applications will handle.

You should not estimate the memory a user needs with only the size of that person's program code in memory. This only applies to small, self-contained programs with no external library. Most of the applications extensively use shared libraries. To be able to determine the size of memory you need, you should consult the software vendor or use AIX memory performance related tools (see Chapter 8, "Performance tools" on page 333).

In order to increase performance when working with a file, AIX is able to *map* files in memory to increase access time. If there is not enough memory for buffering files, this will result in lower performance due to high I/O activity.

You should not forget that having more memory than necessary does not improve the system performance, but lack of memory will degrade performance dramatically.

Following all of these considerations, here is a convenient memory sizing formula for multiuser systems:

```
MEMORY_REQUIRED
= Fixed_AIX_Memory +
   Fixed_Enabler_Memory +
   Fixed_Application_Memory +
   Delta_Enabler_Memory* Number_of_User_Sessions +
   Delta_Files_Memory* Number_of_Open_Files
```

Figure 88. Memory Required for a Multiuser System

The terms in the formula are explained as follows:

Fixed_AIX_Memory
The memory needed to allocate to the AIX kernel and subsystems. This value starts at 64 MB for the operating system itself. The minimum amount of memory for AIX 4.3.3 is 32 MB, but at least 64 MB is recommended. AIX VMM also requires some memory to manage pages. This amount depends on the real memory of the system.

Fixed_Enabler_Memory
This is the amount of memory needed for the enabler code and data structures (like RDBMS). This value depends on the nature and type of the enabler. Most recent RDBMSs need from 30 to 128 MB for server binaries and from 16 MB to 2 GB for data structures (see Section 7.4, “Database sizing” on page 291). For other kinds of application enablers, you should consult the software vendor.

Fixed_Application_Memory
Memory needed for application code and data structures. This amount is generally composed of a fixed amount plus an additional amount for each user. The reason is that AIX reuses the same binary code in memory for all the sessions that have loaded the same program.

Usually this value is between 500 KB and 2 MB. Some self-contained Cobol-based and local Oracle or Informix client applications consume more

than 4 - 6 MB. It is unusual to have applications demanding more than 10 - 15 MB per user.

Delta_Enabler_Memory
This is the amount of additional memory needed by the enabler to register and handle a new user. This term in the formula does not have a linear behavior. It depends on the type of enabler, how it shares resources between the user applications, and the type of applications. Enablers with *2N* design may require between 100 and 700 KB per user, and multi threaded ones between 50 and 150 KB.

Delta_Files_Memory
Like the *Delta_Enabler_Memory* function, this one is used to obtain the amount of memory needed for file buffers in the system. It is very hard to find, but it is possible to use a fixed factor multiplied by the number of different, concurrent open data files in the system. This function applies in a different way for enablers or programs using raw devices rather than JFS. Usually 20 percent of the memory should be reserved for file buffering.

7.2.3.3 Disk

General considerations about disk storage configuration include the following:

Amount of disk storage for data
In multiuser configurations, data storage demands increase constantly. When choosing a server, you should consider the ability to increase disk storage.

Some software product specification letters talk about typical storage consumption for databases they handle. In most cases, those letters refer to basic database configurations, with just the required indexes and data structures. If you need to customize parts (or all) of an application of this kind, you might require more indexes, new tables, hash spaces for tables or data files, additional space for logical deleted records in files, reports, and so on.

In addition to data, you need storage space for operating system code, paging, program binary codes and libraries, file systems for temporary files, logs, and spooler areas.

UNIX files have an inode-based distribution. Each file's inode structure has a tree form, and one file is composed of many branches. When file sizes are increasing, or updates are frequent, fragmentation becomes evident. You should configure sufficient storage space to perform defragmentation tasks.

Data distribution
Data should be spread across disks to increase performance. We recommend to separate the system and paging space from user data. For large amounts of data, disks should be organized by data types; log, indexes, and data.

As demonstrated by statistical analysis, typically, less than 20 percent of data causes more than 80 percent of the I/O traffic in a system. This means that you need to pay special attention to *transaction data files* and paging spaces when configuring disk drives (refer to Section 5.3, “Storage” on page 148).

Paging spaces
Multiuser systems with computer aware users (for example university students or programmers) are particularly prone to running out of paging space, as these users often run multiple programs at the same time.

Paging space configuration depends on the amount of memory and the nature of applications. The problem is not the quantity and size for the paging spaces compared with real memory; the problem is how much the paging spaces are accessed, causing disk I/O bottlenecks or demanding excessive processor time. Your applications may have good locality of reference and use the same code segment in real memory all the time. In this case, paging spaces do not need the fastest disk drives. You can follow these considerations when configuring paging spaces:

- Allow at least sufficient disk space for paging spaces to be twice the amount of real memory.

  The general recommendation is that the sum of the sizes of the paging spaces should be equal to at least twice the size of the real memory of the machine, up to a memory size of 256 MB (512 MB of paging space). For memories larger than 256 MB, we recommend the following formula as a starting size:

  *Total paging space = 512 MB + (memory size - 256 MB) * 1.25*

Note

In AIX 4.3.2, a deferred page space allocation policy was introduced. If you use deferred page space allocation policy, the guideline above may tie up more disk space than actually necessary. The disk block allocation of paging space is delayed until it is necessary to page out the page, which results in no wasted paging space allocation. Adjust the paging space size according to your application. There is no formula which fits all possibilities.

- Configure only one paging space per disk drive.
- Allocate paging spaces on the fastest disks the system has.
- Use between two and six paging spaces for medium systems. Consider the possibility of having more than six paging spaces in large systems.
- Try to configure the paging logical volumes with the same size because AIX uses a round-robin algorithm.
- Do not allocate space for paging spaces in disks with high I/O activity caused by other I/O workloads.
- Configure the paging logical volumes just after installation of the operating system and always before the allocation of space for other logical volumes or file systems.

For more information about paging space, see *AIX Performance and Tuning Guide*, SC23-2365 or the *AIX Performance Management Guide*.

Availability
In a multiuser environment, users are totally reliant on the system to be available and able to provide local functionality. The more users in your system, the more availability it needs. Follow the indications described in Section 5.3, “Storage” on page 148 in order to configure:

- Disk arrays with mirroring for high performance, high availability, and large storage requirements
- Disk arrays with RAID5 for high-availability requirements
- Internal disk drives with AIX and user software mirroring for system availability.

7.2.4 Resources

- *AIX Performance and Tuning Guide*, SC23-2365
- *Performance Management Guide*
- *Capacity Planning for Computer Systems*, Tim Browning
- *Database Performance on AIX in DB2, UDB, and Oracle Environments,* SG24-5511

7.3 File server sizing

The following factors should be considered when configuring RS/6000 machines used as file servers:

- A file server should not be used as a workstation or vice versa. The peak workload of one of the environments can offset the performance of the

other, or the peak workloads of the two environments may occur simultaneously, resulting in unsatisfactory performance in both roles.

- The performance of RS/6000 file servers will generally be constrained by:
  - Processor speed and number of processors
  - Memory size for data caching
  - Speed and number of disks
  - Speed and number of disk adapters
  - Speed and number of network adapters
  - Workload characteristics

7.3.1 NFS sizing

Network File System (NFS) is a file system implementation that provides remote access to files and file systems and is probably the most widely used client/server application for sharing data.

The NFS protocol was developed by Sun Microsystems to allow programs on one system (the NFS client) to access files on another system (the NFS server). The remote directory on the server is mounted to a local directory on the client so the file system on the server looks as if it is resident on the local client. Applications can then access files and file systems located on a remote server without having to copy them locally.

Currently, there are two versions of NFS. Version 1 was never released, existing only at Sun Microsystems. Version 2 was the only version of NFS available for AIX prior to AIX Version 4.2.1. Limitations to NFS V2, such as the 4 GB file size limitation, the write throughput bottleneck due to synchronous writes, and the need for 64-bit file size support prompted the creation of NFS V3. Version 3 supports 64-bit file sizes and reliable asynchronous writes through WRITE and COMMIT procedures increased throughput considerably when compared to NFS V2.

Systems running AIX 4.2.1 or later have the option of either running NFS Version 2 or NFS Version 3 over either TCP or UDP transport protocols. The functionality and performance differences between NFS V2 and NFS V3, and between TCP and UDP, are some of the factors to take into consideration when configuring NFS. The combination of version and protocol to use is controlled primarily via mount options specified by the client.

Table 28 on page 278 shows the default ordering of version and protocol as requested by an AIX client during the mount process. The actual combination

used will be determined by what NFS versions and transport protocols the NFS server supports.

Table 28. Default ordering of version and protocol

AIX 4.2.1	AIX 4.3.X
V3/UDP	V3/TCP
V3/TCP	V3/UDP
V2/UDP	V2/TCP
V2/TCP	V2/UDP

Although less overhead is expected over an UDP mount, increased transmit errors or retransmit requests due to dropped packets or collisions when the network becomes saturated makes the TCP mount a more robust option in some cases. In the presence of dropped network packets, the more efficient retransmission algorithms of TCP also improve the performance. The default ordering of the protocols was reversed as of AIX 4.3 because of significant performance improvements in TCP over UDP.

7.3.1.1 Functionality

Figure 89 on page 279 illustrates the structure of the dialog between NFS clients and a server. When a thread on a client system attempts to read or write into a file on an NFS-mounted directory, the request is redirected from the normal I/O mechanism to one of the client's NFS block I/O daemons (biod). The biod then sends the request to the appropriate server, where it is assigned to one of the server's NFS daemons (nfsd). On the client, one biod is required to send any one read or write request to the server. On the server, an nfsd for each request is dedicated to the biod that sent the request until the request is satisfied and the results of the request sent back to the client. So for each biod to operate a request, an nfsd must be available to handle that request. Other nfsd's may be utilized by operations (e.g., lookups, getattrs) that were not initialized by a biod. The default number of biods per NFS V2 mount is six, and per NFS V3 is four. The number of biods may be controlled via the "biods" mount option. As of AIX 4.2.1, the nfsd's and biod's are multi-threaded.

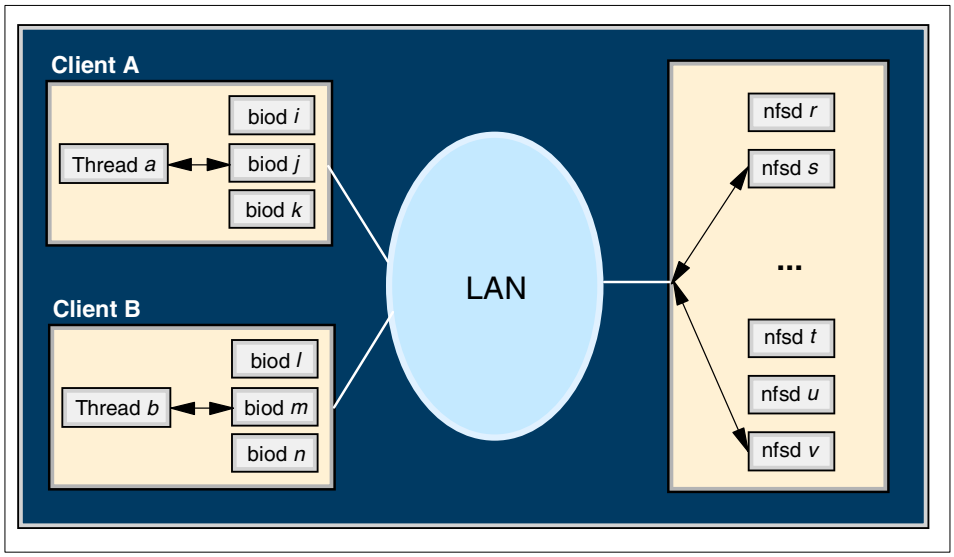


Figure 89. NFS Client/Server interaction

Cache management on an NFS client

In order to increase access performance to distributed files, an NFS client keeps the most-recently accessed information in its cache. The goal is to avoid other transfers over the network because the information is already on the client. However, cache coherency must be maintained. As the server does not keep any record on which clients it has been servicing, it cannot alert them when this information is modified. Therefore, it is the job of the client to manage cache coherency.

- Read access

  Each time a client accesses a file, it has to check the coherency between its cache copy and the server's original file. If the copy's last modification time stamp is newer than that of the server file, then it is considered to be good, and the data can be served to the application from the NFS client's memory (assuming the data has not been paged out by VMM). To find this information, NFS uses the open file attributes. That data may also be in the client cache. But these attributes have a limited validity; by default, three seconds for a file (*acregmin* parameter of the `mount` command), and 30 seconds for a directory (*acdirmin* parameter of the `mount` command). If this time information is outdated, the client must ask for it at the server. Then, it compares it to its copy date. If the copy is older than the original, the client has to make another call to the server asking for the data.

- Write access

In NFS Version 2, the only way to guarantee server data integrity is to execute the operation synchronously. So when an application needs to execute a write operation to a fileon an NFS-mounted directory, a biod generates an RPC call on behalf of the application to synchronously execute the write operation on the server. In NFS Version 2, the maximum RPC write size (*wsize* mount parameter) is 8KB. The call ends only when the server has written the data to non-volatile media.

In NFS Version 3, WRITE and COMMIT procedures allow reliable asynchronous writes and eliminates the synchronous write bottleneck found in NFS Version 2. The NFS client can send multiple WRITE requests and a single COMMIT request when it closes the file, allowing the NFS server to coalesce the client write requests into larger I/Os, which are more efficient than a series of small writes. The 8 KB data size limitation was also changed in NFS Version 3 to improve performance. The default for per-request size for reads and writes for NFS V3 is 32 KB, but can be decreased or configured up to 60 KB (60 * 1024) on AIX.

7.3.1.2 Performance considerations

Often data is moved to an NFS server because it is relatively easy to do so, but one must consider the number of users and file accesses across the network. For example, if many users are accessing a file on the NFS server, there may be lock contention on the file, thus preventing other users from writing to the file, thus risking a performance degradation. It is also important to consider the distance between the server and client (in terms of network topology and response times). Note that NFS data is cached in the virtual memory manager, as is any data page, but NFS data is never paged to disk space on the client. If a page is selected for pageout and later needed again, it will require another server access to read the data.

AIX Version 4.3 introduced the Cache File System (CacheFS). CacheFS can be used to improve performance of remote file systems or slow devices such as CD-ROM. When a file system is cached, the data read from the remote file system or CD-ROM is stored in a cache on the local system, thereby avoiding the use of the network and NFS server when the same data is accessed for the second time. An example where CacheFS would be suitable is in a CAD environment where master-copies of drawing components can be held on the server and cached copies on the client workstation when in use.

Selecting the version of NFS (V2 or V3) and the number of nfsd's (NFS daemons on the server) and biod's (block I/O daemons on the clients), increasing memory, and tuning the disk and logical volume configurations can enhance the NFS system's performance.

You have to take into consideration the server capabilities as well as the typical NFS usage on the client machines when determining how many biods and nfsds to run. Determining the number of nfsds and biods is an iterative process. Some facts to consider are:

- By increasing the number of biods and nfsds, you can avoid having threads blocked for lack of a biod or nfsd daemon because biod and nfsd daemons can handle only one request at a time.
- Increasing the number of daemons cannot compensate for lack of memory, slow processor, or insufficient disk bandwidth.
- NFS daemons are cheap in memory. A biod costs 36 KB of memory (36 KB of memory is pinned). An nfsd costs 28 KB of memory (8 KB of memory is pinned). Furthermore, an idle nfsd does not consume CPU time.
- All NFS requests go through an nfsd, while only read/write operations go through biod.

NFS is based on stateless protocols. A consequence from that is that performance monitoring and management is non-transparent. This means that performance of the clients cannot be measured on the server. AIX offers various commands for tuning and collecting NFS statistics, such as `netstat`, `nfsstat`, `netpmon`, and `nfso`. The UNIX command `netstat` does not give information about which are the most resource-hungry clients, although the AIX-specific `netpmon` command does.

There is a wealth of tunable NFS parameters. Please refer to the *AIX Performance Management Guide* for detailed information.

7.3.1.3 Method and sizing factors

The SPECsfs97 workloads provide one means of contrasting and comparing the NFS-serving capabilities of different machines. Table 29 contains recently published SPECsfs97 results on various RS/6000 machines.

RS/6000 Models	SPECsfs97 Results over UDP
44P-170	5550 SPECsfs97.v2 ops/sec
44P-170	3135 SPECsfs97.v3 ops/sec
7026-M80	27097 SPECsfs97.v2 ops/sec
7026-M80	16557 SPECsfs97.v3 ops/sec
S80	40218 SPECsfs97.v2 ops/sec
S80	29083 SPECsfs97.v3 ops/sec

Table 29. RS/6000 SPECsfs97 Results

For example, the table would tell us that an M80 has about five times the capacity of a 44P-170 as an NFS server. Note that this statement, however, is specific to the SPECsfs97 workloads! The SPECsfs97.v2 and SPECsfs97.v3 workloads are distinct, each characterized by its own mix of NFS operations.[1]

Therefore, the results do NOT tell you:

1. That the M80 will perform five times better than the 44P-170 on ANY NFS server workload.
2. That NFS V3 performance is worse than NFS V2 performance on these machines.

Nevertheless, a methodology similar to what is used to size the minimum amount of supporting hardware/clients required in a SPECsfs97 benchmark setup may be used for other workloads as described in the following sections. Note that the methodology attempts to remove all memory, I/O, and network subsystem bottlenecks. Therefore, in some cases the amount of equipment used to support this benchmark may seem excessive. The intent is to maximize CPU utilization and remove all I/O wait time.

Get a SPECsfs97 requirement

The easiest requirement to work with is one specified as a SPECsfs97 target. In this case, going to the SPEC web page (`http://www.spec.org`) and looking at the IBM SPECsfs97 results will give you a reasonable indication of a system that would be appropriate.

[1] See “The Advancement of NFS Benchmarking: SFS 2.0” by David Robinson, *Proceedings of LISA ‘99: 13th Systems Administration Conference*.

In the absence of this type of requirement, you must try to gather as many of the following as possible to improve the accuracy of the sizing.

In the case of server replacement/upgrades, you should analyze data acquired from the current production environment.

Get a system memory requirement
It is difficult to estimate the optimum amount of memory in an NFS serving environment. In general, the more memory the better!

In an established environment, you can use data from iostat (see Section "Get a storage subsystem throughput/space requirement" on page 283 for disk throughput requirements) to decide if additional memory would be helpful. If iostat indicates a large percentage of I/Os are reads, and the reads are for filesystem data, then adding memory will allow for more caching of this data, reducing the amount of disk I/O.

Get an NFS ops/sec requirement
Find an estimate of the number of NFS operations per second the server must handle by measuring the following over some interval(s):

- Measuring client activity (`nfsstat -rc`)
- Measuring server activity (`nfsstat -rs`)

Pay close attention to the mix of operations.

Get a storage subsystem throughput/space requirement
Use iostat to gather disk throughput data, including:

- Disk utilization (the tm_act column)
- Transactions per second
- Kbytes per second
- Percentage reads vs writes

For good performance on SPECsfs97, enough disk adapters and disks must be used to ensure that no disk shows a utilization of more than 50 percent busy.

Typically, customers will base their storage subsystem needs solely on the amount of space they require. It is important to point out that for optimum performance, having sufficient adapters and disk arms (to avoid excessive I/O wait times) and planning for a good filesystem layout (to spread out I/O load and avoid hotspots) must also be considered. Individual adapter and disk performance characteristics must be understood.

Get a network subsystem throughput requirement

Use the statistics gathering commands for network adapters (e.g., `entstat` for Ethernet) to get network throughput data, including:

- Packets per second (receive and transmit)
- Kbytes per second (receive and transmit)

In terms of Kbytes per second, it is reasonable to expect no more than 60 percent to 80 percent of the line capacity per adapter in workloads consisting primarily of large sequential reads/writes (e.g., 10000 Kbytes/sec for a 100-Mbps Ethernet adapter running in half-duplex mode). For workloads characterized by accesses which are smaller and more random in nature, you may get no more than 30 percent to 50 percent of the line capacity. In these cases, the limiting factor may end up being the number of packets per second that the adapter can handle. As with disk adapters, it is important to understand the performance characteristics of the network adapters.

Table 30 shows estimates of the amount of SPECsfs97.v2 throughput that different network adapters/interfaces can comfortably sustain. These are based on internal benchmark runs.

Network Interface	SPECsfs97.v2 ops/sec
100-Mbps Ethernet	3000
100-Mbps FDDI	3000
155-Mbps ATM (9000-byte MTU)	5000
1-Gbps Ethernet (1500-byte MTU)	10000
1-Gbps Ethernet (9000-byte MTU)	15000

Table 30. Network interface SPECsfs97.v2 capacity

Putting it all together

Here we will look at some of the equipment used for the 44P-170 SPECsfs97.v2 UDP workload run. System data used in this analysis was gathered during internal benchmark runs at the peak throughput.

- 1GB of system memory. The percentage of I/Os that were reads was close to 50 percent. Based on runs on other systems with larger amounts of memory, better performance could have been achieved with more memory.
- A single 1-Gbps Ethernet adapter (9000-byte MTU). Given the estimates from Table 30 on page 284, this single adapter was more

than sufficient. Alternatively, note that two 100-Mbps Ethernet adapters could have been used instead.

- A single SSA 160 SerialRAID adapter and 48 18.2GB 10K RPM SSA drives. The total disk throughput was approximately 13200 Kbytes/sec and 3000 transactions per second. There were 8 disks per volume group; one disk contained the filesystem logs, and the other seven contained filesystem data. The data disks were about 35 percent busy, and the log disks about 20 percent busy. This is well below the 50 percent busy target mentioned earlier. Therefore, the storage subsystem layout was adequate.

7.3.2 AIX Fast Connect sizing

In this chapter, we discuss the AIX Fast Connect which was introduced as a feature for AIX 4.3. It is a file and print server for AIX 4.3.2 or later systems. AIX Fast Connect provides throughput that is among the fastest of the commercially available AIX-based file and printer servers.

An evaluation version of the AIX Fast Connect product is included in the Bonus Pack for AIX version 4.3, announced June 8, 1999.

7.3.2.1 General sizing considerations

The AIX Fast Connect provides file and printer services for Windows or OS/2 clients by implementing the Server Message Block (SMB) protocol. However, it does not provide the services themselves. It just provides SMB protocol and the services such as file and printer that are provided by the AIX operating system.

Processor

The most heavy processor workload for the AIX Fast Connect is login of new users. Our test results indicate that file operations and print jobs do not need much processor power. You need to estimate how many users connect at peak period. Generally, morning office hours from 8:00 - 10:00 will be your peak period. Estimate the number of new connections per second.

Memory

To estimate the amount of memory for the AIX Fast Connect, use the formula below. You need to estimate the amount of memory for print jobs, file cache, user application, and so on.

Memory = 64 MB for AIX + 0.5 MB * the number of active users

Figure 90. AIX Fast Connect memory

To allocate enough file cache may increase the cache hit rate. You can tune AIX VMM using `vmtune` command. A new `vmtune` option has been added in AIX 4.3.3.

Network
Estimate the peak network workload. The network utilization should be below 30 percent to avoid network collision. Your system may need more than one network adapter. For instance, the network workload may be heavy just before the lunch time because users save their files at that time. Consider the conditions like that and estimate carefully.

Allocating enough memory for *thewall* with the `no` command may increase network performance. The maximum value of *thewall* is 1 GB in AIX 4.3.2 and later. For more information about mbuf pool performance, see the *AIX Performance Management Guide*.

Disk
For the AIX Fast Connect, at least 50 MB of disk space (/usr, /etc, and /var) is required for installation. AIX will write temporary files to the /var filesystem. After temporary spool files are printed, AIX will erase them. After studying your users printing behavior, you should estimate enough disk space for /var filesystem carefully. If you need large disk space for spool files, it is useful for administration to create file systems and mount. See Table 31.

Table 31. Spooling temporary files

Directory name	Explanation
/var/spool/lpd/qdir	Print job description files
/var/spool/qdaemon	Print job copies
/var/spool/stat	Printer queue description files
/tmp	System temporary files

You should consider the balance of system such as processor power, disk I/O, PCI bus bandwidth.

7.3.2.2 Resources
- *AIX and Windows NT: Solutions for Interoperability*, SG24-5102
- *AIX Performance Management Guide*
- *Problem Solving and Troubleshooting in AIX Version 4.3*, SG24-5496
- *AIX Fast Connect Functions and Sizing Guide*, SG24-5527

7.3.3 Client/Server sizing

It is important to understand that the terms *client* and *server* refer to software, not to hardware.

7.3.3.1 Client/Server environment

A software client usually consists of two pieces. The first piece is the client application software; the second is what we will call *client enabling software*. Both communicate using a carefully specified common language, called an application program interface (API). The client enabling software takes any request the client application software makes via the API and verifies it for correctness. It then decodes the request and forwards it to one or more servers for action.

Usually, the servers reside somewhere else on a network, so the client enabling software also creates links or sessions over the network to the servers. When the servers have finished, they send the results back to the client enabling software. The client enabling software then interprets these results and gives them back to the client application software via the API. For the rest of this chapter, we will refer to the combination of the client application program and the client enabling software as the client.

A network, usually a LAN (but a WAN is also feasible), carries interactions between client enabling software and the servers. The server software can usually accept requests from dozens, hundreds, or even thousands of clients concurrently. Clients may request services from one server or from many servers, depending on the application's needs. Client/server computing can take place within an organization or between organizational or enterprise boundaries to support a business process.

Like clients, servers are also software and can coexist on a single computer or be set up on separate computer systems. This provides flexibility.

There are five models of client/server computing:

- **Front end model**

  This is the simplest client/server model, where only a part of the presentation layer is distant.

  This approach provides a graphical front end for existing applications. Some call it the *face lift* approach, because the looks improve while everything else remains the same. The front end model increases user productivity and reduces training costs without changing the original software.

The back end applications may provide data access services, transaction services, locking services, and other, similar functions. The applications already know how to present data to users and accept textual data from users. They format streams of data, most often block-mode screens, for output, and interpret keystrokes as input. Other users may use the same back end applications using block-mode or character-mode terminals while workstation users concurrently use the graphical front end. Either way, the application does not know that the data is coming from another program.

- **Remote presentation model**

  This second model puts the whole presentation process on the client systems.

  First, visual output generated by an application on one system gets displayed on another. Next, the system that displays the output also takes the user actions and turns them into input for the application. Several examples of the remote presentation model exist, but two approaches dominate today: the X Windows system and Web browsers.

  X11 defines a graphical interface server that runs on the user machine. The application processing is done on the client machines. The server provides a graphical output device for the client applications and takes the user's mouse movements, keystrokes, and menu choices and sends them on to the correct application. In this model, the client application doesn't know how to display graphical output or grab mouse movements, so it asks for help from the server.

  Another common remote presentation application is the World Wide Web. The World Wide Web is a way of interacting with data stored on machines attached to the Internet. People use a browser to search for and retrieve data, including text, images, and video, from servers located throughout the Internet.

- **Resource sharing model**

  In this case, presentation is local to the clients and data is centralized.

  This model covers most of the client/server marketplace today. File servers, printer servers, client/server database software, fax servers, and similar products all fall into this model. The client/server software makes remote devices and data appear local to personal computer applications and users.

AIX Fast Connect

AIX Fast Connect is a file and print server for AIX 4.3.2 or later systems. AIX Fast Connect is integrated with and exploits AIX's key features to provide fast, efficient, and scalable file and print services to Windows95 / 98 / NT, OS/2, and other clients that use the Server Message Block (SMB) and Microsoft Common Internet File System (CIFS) protocols on Network Basic Input / Output System (NetBIOS) over IP transport.

- **Data staging model**

  Like the resource sharing model, presentation is local to the clients, but data is stored at different levels.

  Sometimes, sending all the data needed from a central site is too costly or time consuming. Replicating the data to each workstation, though, is also unwieldy. In these cases, it might be useful to duplicate the data at several sites if little of the data changes regularly. When these conditions fit, the data staging model is a good choice.

  It is fair to say this approach optimizes the costs and performance of a centralized data storage and retrieval design. It retains the elements of centralized control over the data, but it allows access to the data quickly. This makes it suitable for use, given the right conditions, for workgroups and for critical data.

- **Distributed logic model**

  In this approach, neither part of the application can stand on its own. Some processing must be local and some centralized. Using the distributed logic model commits a business to using workstations and other programmable devices instead of ordinary terminals. This model maintains its data centrally. This makes it suitable for critical applications and data.

7.3.4 General sizing considerations

An added complexity in client/server environments is dealing with multiple sites.

A distributed environment needs to be studied as a group of different components, each needing to be examined first individually, then as a part of the global group. The different components are the network, the clients, and the server(s). You can further split these components into their hardware and software parts, like CPU, memory, I/O, operating system, and application. These various elements can be measured to help size the environment.

The main factors that characterize the workload of an application are:

- **Functional level**
  - Type of activity of the customer (such as distribution, banking, pharmacy)
  - Type of application (such as bookkeeping, inventory)
  - Kind of user (such as secretary, engineer)
- **Middle ware level**
  - Type of middle ware used: database, transaction processing monitor, message queuing, and so forth
- **Application level**
  - Languages and tools used to develop the application (such as C, FORTRAN, 4GL)
  - Complexity of the queries in each program
  - Complexity of the algorithmic and computational parts of each program
  - Complexity and number of fields on each screen (determine how many characters are typed and sent between the terminals and the CPU to size the network)
  - The quality of program coding
- **User activity**
  - Number of connected terminals
  - Number of active users (average and peak)
  - Transactions/programs used for each kind of user
  - Think time for each transaction/user (time delay spent by the user to think about what he is going to type)
  - Keyboard time for each transaction/user

CPU

It is extremely difficult to offer general sizing recommendations for a client/server CPU configuration because the size is strongly dependent on the application itself.

Memory

The same can be said of memory. It is completely application-related.

Disk

The general idea is to balance the I/O workload between your disk drives.

If the application is I/O intensive, then you should consider using fast disks (like SSA), several adapters and only a few disks per adapter. You should also distribute your data so that there will not be contention on one disk while the other drives are idle.

Network
The network is a big issue in client / server environments because your choice affects performance profoundly. It should not be underestimated, or a bottleneck will occur. You need to evaluate the mean and peak network traffic.

When using a WAN, network latency may become an issue for the exchange of information between the server(s) and the clients. You have to know whether the application is interactive (which induces much traffic between client and server) or not, and you need to know what effects a long waiting time will have on the application or the users.

7.3.5 Resources

- *AIX and WindowsNT: Solutions for Interoperability*, SG24-5102

7.4 Database sizing

We provide the general concepts for sizing database systems based on the IBM RS/6000 platform. There are a few widely used RDBMSs such as DB2 UDB, Oracle, Sybase, and Informix. Although they have the same goal, their implementations are quite different.

Note

This chapter *does not* provide detailed performance, tuning, and sizing information for a specific RDBMS product. For further information refer to the following website: `http://www.redbooks.ibm.com`.

7.4.1 Database environment

Higher query performance and scalability is required for present RDBMS environments such as data warehouse, DSS (Decision Support System), and OLAP (Online Analytical Processing). Even traditional OLTP (Online Transaction Processing) systems become much more complex and larger each year, and demand higher transaction rates. Furthermore, the explosive growth of the Internet generates new database requirements such as new marketing methods for world wide customers.

The largest demands of query or transaction performance can require further scalability. This is provided by the RDBMS vendors via parallel processing database products and can require the assistance of transaction monitors.

Database architectures can be classified into the following types:

Fundamental database
The simplest type is a single processor, memory, and one disk subsystem. There is only one copy of the OS on such systems.

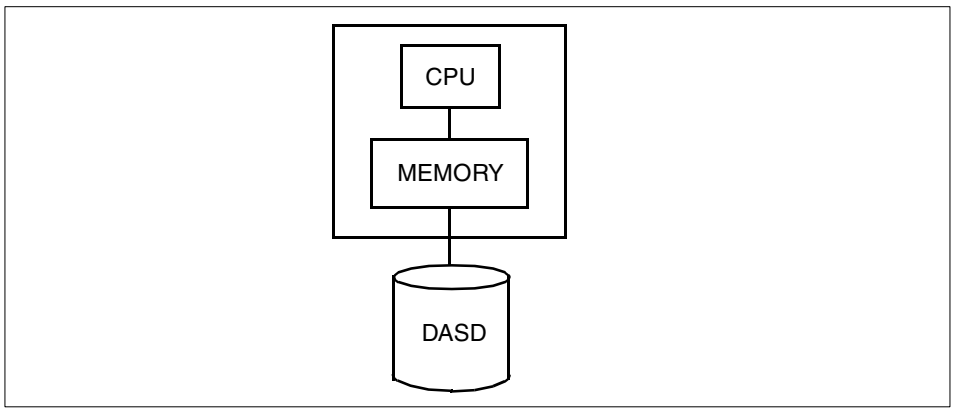


Figure 91. Fundamental database environment

Shared nothing
Loosely coupled processors are linked by a high-speed interconnection. Each processor has its own memory, runs its own copy of OS, and accesses its own disks.

Examples of machines that implement this architecture are RS/6000 SP or clusters of systems.

This type of architecture offers the following advantages:

- Scalability in terms of database size and number of processors
- Performance gains from not sharing resources across a network
- Use of heterogeneous environments

This architecture is best suited for parallel queries. The query is divided among processors. The advantages are that processing is more distributed and that the database can manage a larger amount of data.

If the data can be evenly distributed and there is little inter-processor data movement, the performance gains can be scalable to huge data volumes. In these systems the task (function) is send (shipped) to the data to reduce network traffic. This is called function shipping.

If the same data is required on many processors, this can take further inter-process communication and processor resources.

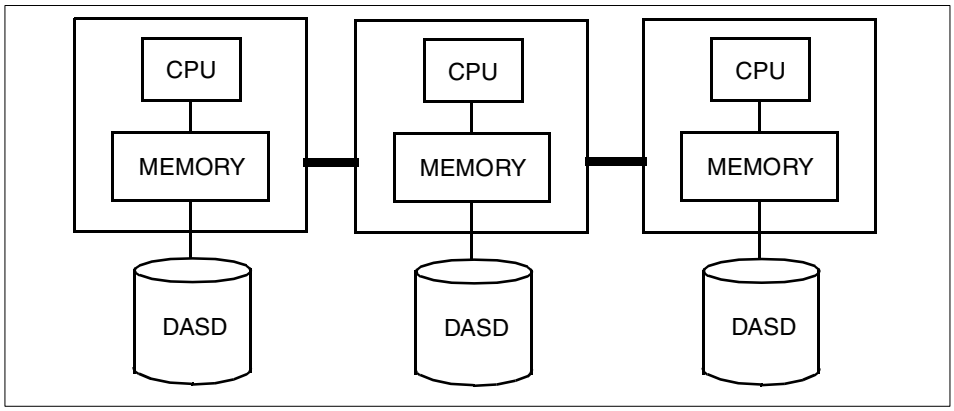


Figure 92. Shared nothing database environment

Shared disk

Every processor has its own memory, but it has a global view of all the data. This can be implemented either by hardware or software.

Examples of machines that implement this architecture are clusters of RS/6000 using HACMP concurrent logical volume manager or RS/6000 SPs using Virtual Shared Disk (VSD). See Section 4.2.4, “Shared disk components of Parallel System Support Programs” on page 110.

Possible advantages of this architecture are availability and the capacity to use a heterogeneous environment.

Shared disk architecture is more I/O-shipping oriented. I/O shipping means that any CPU can access any part of data and the data is moved (by I/O) to the CPU performing the operation. I/O shipping is implemented by shipping data to one or more processors and then executing the database operation. It requires more movement of data because the data is transferred before any operations are performed. As there may be more than one copy of any data item, a global locking mechanism is needed. This can take further inter-process communication and processor resources.

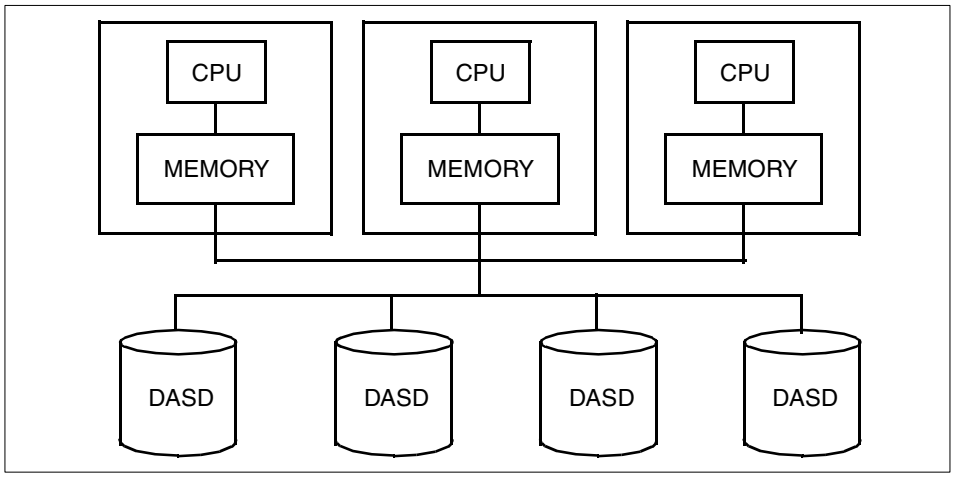


Figure 93. Shared disk environment

Shared memory
This is also called shared-all architecture, as multiple processors access the same memory and disks. IBM SMP servers are among the machines that implement this architecture.

The advantages of this architecture are:

- Simultaneous processing for concurrent queries
- Parallel processing for a query
- Easy administration

SMP systems scale very well up to database sizes of a terabyte but, are ultimately limited by bottle-necks in memory access as the number of CPUs increases.

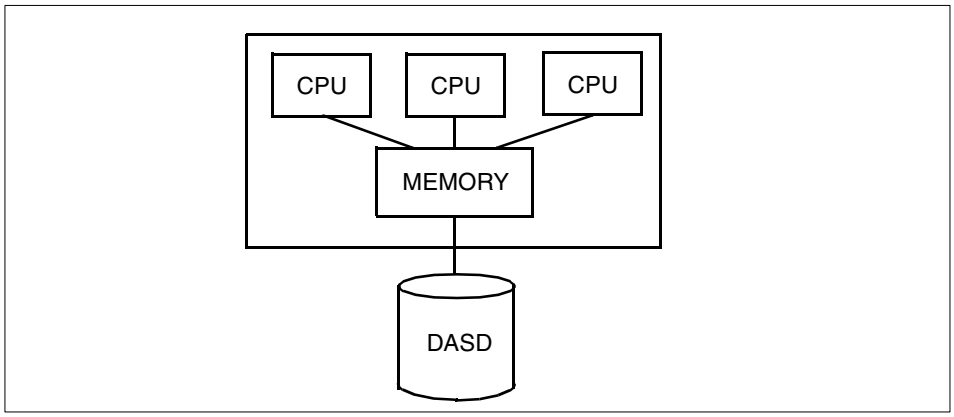


Figure 94. Shared memory environment

Transaction parallelism may be divided into two types:

Inter-Transaction parallelism
Inter-Transaction parallelism is achieved when different transactions are operated simultaneously against one database. This is accomplished by having each available processor perform a different transaction.

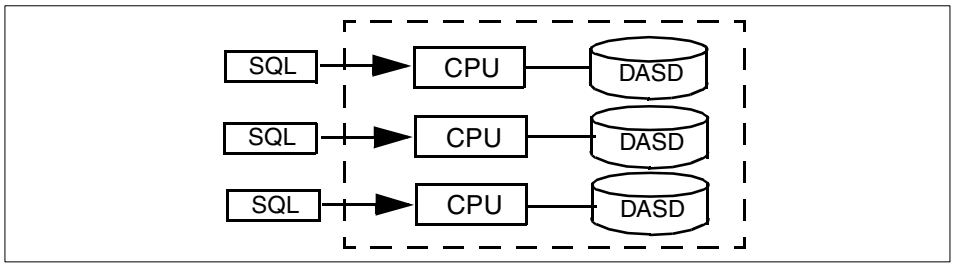


Figure 95. Inter-Transaction parallelism

This type of parallelism is beneficial when there are many different concurrent transactions, none of which are heavily computational. Good results may also be obtained if the database is small but frequently accessed. Global elapsed process time is reduced.

The hardware architecture that fits best in inter-transaction parallelism is shared memory.

Intra-Query parallelism
A single query is split across many processors.

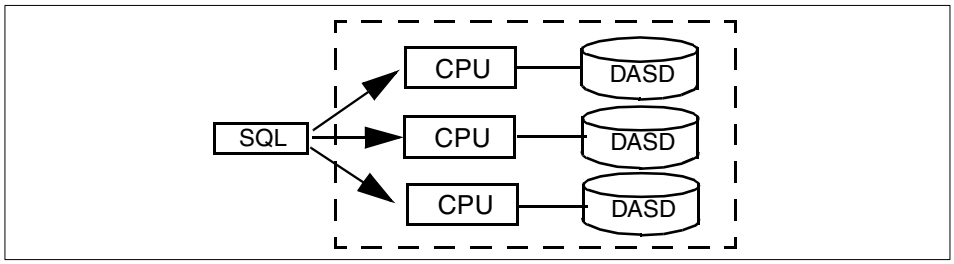


Figure 96. Intra-Query parallelism

The benefit of intra-query parallelism is a speed-up in processing time. This type of parallelism enables more complicated and/or more computation-intensive operations to be performed in a reasonable time span.

This architecture is well suited for low numbers of large queries.

Intra-query parallelism can be achieved in two forms; partition parallelism or pipelined parallelism. Partition parallelism is a query decomposition. A single query is subdivided into several subqueries, each of which processes a subset of the data. On the other hand, pipelined parallelism involves dividing the query into a series of operations. The output from one operation is used as input to the other.

The hardware architectures most suited to this type of parallelism are shared memory, shared nothing and shared disk.

7.4.2 Transaction processing monitor environment

A transaction processing (TP) monitor is a layer between the RDBMS server and the application. The application must be written according to the monitor's language, making transactions instead of directly accessing the database with SQL queries.

TP monitors are generally used to provide better performance for a given configuration, or when the global workload is too heavy for an RDBMS to manage it on its own. They supply better performance because they regroup many client requests into one directed to the database server. Thus, fewer processes or threads are used on the server side, and there are fewer concurrent RDBMS clients, which means less demand on resources, particularly memory. Moreover, it also implies less network traffic, which is essential when clients and server are connected through a slow WAN or a busy LAN.

7.4.3 Sizing RDBMS

The real problem of sizing RDBMS is accuracy. Initial sizing is a prediction based on few or no facts. Ideally, you should understand the following when sizing an RDBMS server:

- Complexity of the transactions
- Transaction rate
- Read / write ratio
- Number of concurrent connections
- Number of concurrent transactions
- Sizes of the largest table
- Performance objectives
  - Throughput
  - Maintenance windows such as backup, re-org, load

We provide some sizing rules of thumb in this section such as:

- The number of physical nodes
- The number of database partitions
- Amount of memory
- Amount of disk space
- The number of disk drives

In the following sections we are going to cover the general rules based on the results of experience in system sizing, benchmarks, and performance tuning.

7.4.3.1 RDBMS utilization

These utilization figures are the results of benchmarks and monitoring well balance and high performance production systems.

Table 32. RDBMS utilization

Utilization	Good	Bad	Ugly
CPU	< 70 percent	85 percent	> 90 percent
Disk	< 30 percent	< 40 percent	> 50 percent
Paging	zero	< 10 pages / sec (for each CPU)	> 10 pages / sec (for each CPU)
Network	< 30 percent	> 30 percent (details see below)	
Run Queue	< 2 * CPUs	N/A	
Paging Space	2 * memory	N/A (see below)	

- Network:
  30 percent of theoretical network bandwidth stops collisions on an Ethernet type network from becoming a problem, which reduces throughput. Token ring type networks can be driven to 60 percent with no drop in throughput.
- Paging Space:
  If your system is small and has many users, we recommend three times memory. If your system is large, some systems go below this size because lots of memory is used for the RDBMS cache and it therefore does not need paging space.

7.4.3.2 Sizing processor

To estimate the power of CPU, keep in mind that you should size the server for the peak workload that can occur. You should understand your peak workload and your application type. In this section, we provide a method for sizing CPU with the workload and relative OLTP power rating (Rel.OLTP).

The IBM publishes Rel.OLTP, which is an internal benchmark for all servers in the RS/6000 family. Rel.OLTP is the Relative Online Transaction Processor Power. The RS/6000 Model 250 is taken as a power rating of 1. This number is an IBM Austin Labs supplied number that is used to show the performance of RS/6000 processors relative to each other. It is based on an OLTP application workload with simple transaction, a mix of insert, update, delete, and select transactions using a RDBMS, and lots of users. Rel.OLTPs are helpful for sizing the CPU of RDBMS servers. For more information about Rel.OLTP, see Section 6.4, “ROLTP” on page 244.

Database I/O

Estimate the number of database logical read and write operations. Database logical read operation means the number of database read operations performed for each transaction. Read operations include reading index and data blocks. Database logical write operation means the number of database operations that this transaction performs for each transaction. Write operations include writing data blocks, updating all indexes, and writing logs.

Transactions

You should understand and define the transaction type, peak time, the number of users, and the number of transaction each user will perform that happen in the peak period. For example, there are three transaction types of a typical application; light, medium, and heavy. The number of concurrent connecting users for each transaction type should be assumed. The peak time is from 10:00 to 11:00, and the numbers of transactions each user will perform in the peak period are:

- Light = 120 transactions/user
- Medium = 60 transactions/user
- Heavy = 15 transactions/user

Define your peak time transaction carefully.

CPU seconds per transaction

After studying the transaction, estimate the amount of CPU seconds used to run each transaction on a 1 Rel.OLTP machine from the transaction you estimated. If a Model 250 can support 10 transactions per second, each transaction takes 0.1 CPU second. It is critical for correct sizing. If you have no information on the CPU seconds in Rel.OLTP terms, use Table 33 as a guide.

Table 33. Typical CPU seconds per transaction

Application	Description	Transaction name	CPU seconds / transaction
Financial	Typical Complex Configurable Financial Package	Journal Entry	1.5
		Account Inquiry	0.4
		Journal Inquiry	0.6
Typical	Typical Simple Application (in C or COBOL) Transactions	Light	0.1
		Medium	0.5
		Heavy	5.0
4GL Form	Typical PC based GUI Forms Application written in 4 GL	Heavy Complex	9.0
		Moderately Heavy	4.5
		Moderate	3.0
		Moderately Light	1.0
		Light	0.5

These will help you estimate your CPU seconds per transaction. You should consider your complexity of transaction and estimate your CPU seconds per transaction carefully.

Estimate your Rel.OLTP
The Rel.OLTP you need can be estimated by the formula below:

```
Needed Rel.OLTP = Sum (NU * TX * CS / PP) / MC
```

Figure 97. Needed Rel.OLTP formula

- Needed Rel.OLTP: Needed Rel.OLTP value for your system
- NU: Number of users at peak time
- TX: Number of transactions per user at peak time
- CS: CPU second per transaction on a Rel.OLTP = 1 machine
- PP: Peak time period
- MC: Maximum CPU utilization (under 70 percent is recommended)

This calculation formula is a little bit difficult to understand. We provide an example estimation in Section 7.4.3.6, “Example” on page 303.

7.4.3.3 Sizing memory

To size the amount of memory required, the number of concurrent users and the query workload characteristics should be considered. More users require more memory. Large DSS style scan queries may not require as much because no amount of memory can contain the huge volumes of data. Smaller and indexed based complex queries can make use of more memory to cache data.

```
Memory = AIX
         + RDBMS_code
         + RDBMS_data_cache
         + ( User * Application_Resident_Set )
         + Filesystem_Cache
```

Figure 98. RDBMS memory rules of thumb

- AIX operating system: recommend 64 MB
- RDBMS_code: recommend at least 32 MB
- Application Resident Set:
  This is the code and data of the application that each user needs to run. The resident set refers to the fact that a paging system does not need to have the entire program in memory to run and usually only a proportion of the application is required. Also, AIX shares code between copies of the application, saving memory. The `ps aux`

command will tell you the size of an application resident set. If no information is available, the following is a start point:

- We recommend for a simple application coded for example in an efficient develop language such as C to use 2 MB per user.
- We recommend for a more complex application with a lot of functions or written in a modern 4GL environment to use 6 MB for each user

Note

Application code is shared, so most of this 2 - 6 MB is application private data. If there is a low number of screens in the application use the lower size. If are hundreds of screens or a complex algorithms use the higher size. If you have no idea, use 4 MB.

- RDBMS data cache
  AIX provides superior use of memory and balancing the dynamic allocation based on demand. Any unused memory is used to speed up reading and writing of files to the UNIX file systems. Some memory is always used for this purpose. If the RDBMS data is held in file systems, a large file system cache is required. If the RDBMS data is held in *raw disks* such as VSD, then the file system size can be reduced in favour of more RDBMS cache. Rules of thumb are:

  - A minimum of 32 MB is required for the combined caches.
  - For an OLTP application a good starting point would be to use five percent for data disk size (means raw data size plus header of RDBMS, 5 - 10 percent is recommended as a rule of thumb) for the RDBMS data cache.

  For setting the size of the RDBMS cache in practice production the following is used:

  - Start point if you have no information is half of memory for the RDBMS cache.
  - In OLTP systems the cache is often higher, such as 70 percent of memory, to ensure the most often used tables and rows are always in memory.
  - In many complex CPU heavy applications and Business Intelligence (BI) or DSS workloads use less memory, such as 30 percent of memory, for the RDBMS cache if they effectively have to read large proportions of the database to answer SQL statements.

- File system cache
  AIX file system cache is used by the AIX operating system to save copies of recently used disk blocks. It avoids disk I/O for performance. Even if the RDBMS uses raw devices (including Oracle on SP using the VSD raw devices), the system needs some file system cache.
  When the RDBMS uses the AIX file system cache, a balance must be made between RDBMS cache and filesystem cache (in the ratio 3 to 1). The combined size will be similar to the RDBMS cache size for raw device databases.

If you do not have any information such as the number of concurrent users and the query workload characteristics, memory can very roughly be sized as five percent of the raw data size plus 64 MB for AIX and 32 MB for RDBMS processes.

7.4.3.4 Sizing disk space

The disk space can be very roughly sized via following raw data to disk space ratios as a rule of thumb. If you do not have any information, use 1 : 3 raw data to disk space ratio.

- OLTP Ratio 1 : 3 to 1 : 4
- DSS Ratio 1 : 4 to 1 : 5
- Data Warehouse Ratio 1 : 5 to 1 : 7

For example, your raw data size was 300 GB on a DSS environment, 1.2 TB - 1.5 TB non-mirrored or 2.4 TB - 3.0 TB mirrored disk space is required. We recommend using AIX mirroring or RAID 5 for disk protection.

The prime RDBMS data parts of any RDBMS are:

1. Data: User data are contained.
2. Index: This is almost the same size as data.
3. Tmp/sort: This is used for creating indexes and sorting temporary tables during SQL statements, which can be larger than the largest table in the database.

You should add the RDBMS header for each part mentioned above. It can be roughly estimated from raw data size plus the header of RDBMS; 10 percent is recommended as a rule of thumb.

7.4.3.5 Sizing the number of disks

There are 9.1 GB, 18.2 GB, and 36.4 GB disk drives available for RS/6000. You should decide which disk drive is good for your system based on database size and considering I/O requirements. Many small disks increase

the I/O throughput but require extra adapters and PCI bus slots. Larger disks are generally less expensive per GB of storage.

The OLTP environments require random I/O in general and tend to drive disks harder than DSS environments. We recommend eight to 15 disks for each CPU for I/O parallelism. There is no substitute for multiple disk arms.

The DSS environments usually require continuous I/O of large data. We recommend 6 - 10 large disks for each CPU.

The read and write ratio of different databases is important to understand. The read / write ratio is important to chose disks. See Table 34.

Table 34. Typical read / write ratios

Application type		Read	Write
OLTP		80 percent	20 percent
BI	Day	99 percent	1 percent
	Night (loading)	50 percent	50 percent

Attention

You should consider the balance of your system. A good balance of processor, memory, and I/O creates the best performance. You should not refer only to benchmark results. For instance, if Rel.OLTP is supported by a 43P-270, you should examine the PCI bus, network adapter, and the I/O bandwidth such as SSA and SCSI adapters. Your system may need more CPUs or I/O bandwidth via PCI adapter slots. Furthermore, you should consider the growth of data and the number of users within the system life time.

7.4.3.6 Example

We are going to size the database system for an imaginary corporation *Corporation.com*. The assumed transactions of Corporation.com are:

1. Peak Time: 10:00 - 11:00 = 1 hour = 3600 seconds
2. Transaction type: Typical simple application without complex queries
3. The number of users
   - Light = 2000
   - Medium = 50
   - Heavy = 5

4. The number of transaction per user in the peak period
   - Light = 120 transactions / user
   - Medium = 60 transactions / user
   - Heavy = 15 transactions / user
5. CPU seconds per transaction on a 1 Rel.OLTP machine
   - Light = 1
   - Medium = 3
   - Heavy = 15
6. Maximum CPU utilization: 60 percent
7. Raw data size: 100 GB

Attention

For estimating the real production database system, you should consider a natural increase in the amount of data and the number of users for 3 - 5 years.

Step 1: Sizing needed Rel.OLTP

Calculate the needed Rel.OLTP using the formula mentioned in Section 7.4.3.2, “Sizing processor” on page 298.

1. Light transaction: NU * TX * CS / PP = 2000 * 120 * 1 / 3600 = 66.0
2. Medium transaction: NU * TX * CS / PP = 50 *60 * 3/ 3600 = 2.5
3. Heavy transaction: NU * TX * CS / PP = 5 * 15 * 15/ 3600 = 0.3
4. Sum (Rel.OLTP) / MC = (66.0 + 2.5 + 0.3) / 0.7 = 98.3 Rel.OLTP

Therefore, the needed Rel.OLTP is 98.3, but do not forget other workloads such as network, I/O, PCI bus bandwidth, and SCSI / SSA adapter bandwidth when you decide on hardware models.

Step 2: Sizing memory

We can calculate the amount of memory using the formula mentioned in Section 7.4.3.3, “Sizing memory” on page 300. We assume the RDBMS is based on raw devices, and application resident set size is 4 MB.

```
Memory = AIX
        + RDBMS_code
        + RDBMS_data_cache
        + ( User * Application_Resident_Set )
        + Filesystem_Cache
        = 64 + 32 + (100 * 0.05 * 1024) + (2000 + 50 + 5) * 4 + 32
        = 13468 MB = 13.5 GB
```

Figure 99. Required amount of memory

Step 3: Sizing disk

We assume the RDBMS header is 10 percent of raw data size. The application of Corporation.com is a typical OLTP application. So, we can use 1 : 3 ratio, but in this case we consider the RDBMS header and we assume the ratio of *Data : Index : Tmp / Sort = 1 : 1 : 1*. The total size of data is estimated as follows:

- RDBMS Data = 100 GB + 10 percent = 110 GB
- RDBMS Index = 100 GB + 10 percent = 110 GB
- RDBMS Tmp / Sort = 100 GB + 10 percent = 110 GB

Therefore, the total size of data is 330 GB. You should consider the disks for AIX, paging space, and RDBMS log. We recommend one disk for each.

Step 4: Sizing the number of disks

The application of Corporation.com is a typical OLTP application, so 10 - 20 small disks for each CPU are recommended. We chose 9.1 GB SSA disks. We decide to use mirroring of AIX LVM for disk protection.

- Data: 110 GB / 8.75 * 2 mirror = 26 disks
- Index: 110 GB / 8.75 * 2 mirror = 26 disks
- Tmp / sort: 110 GB / 8.75 * 2 mirror = 26 disks
- 1 RDBMS log * 2 mirror = 2 disks per node

Moreover, four 9.1 GB SCSI internal disks are needed for AIX and paging space using LVM mirroring.

Step 5: Configure the system

The estimated resources for Corporation.com are:

- Rel.OLTP: 98.3
- Memory: 13.5 GB
- Disk space: 330 GB plus disk protection

- Number of disks: 80 disks for data and log, 4 disks for AIX and paging space

> **Note**
>
> Some RDBMS products do not support large memory such as over 2 GB per node, and each hardware model has a maximum memory size. You should be careful when deciding on the hardware model or number of nodes.

For Rel.OLTP many current RS/6000 models can support it, for example F80, H80, M80, S80, and the IBM @server pSeries 680. These models also support sufficient memory and I/O bandwidth. The smaller models will mean reduced price, but the large models with allow for future growth.

26 disks are for data, index, and tmp / sort. This means two 7133s are needed for each. A 7133 can contain 16 disks, so 13 disks are placed at each 7133. Three empty seats are available for the future. Therefore, data, index, tmp / sort have two 7133s each. Six 7133s are needed. Since we use mirroring, using two separated SSA loops are recommended for SSA adapter protection. The needed number of SSA adapters is 2 * 6 = 12.

You should chose more than one machine or node for availability. This could be a pair of SP nodes, a pair of RS/6000's of the same size, or a production and smaller backup machine (if reduced take-over mode performance is acceptable).

7.4.4 Resources

- *Database Performance on AIX in DB2, UDB, and Oracle Environments* SG24-5511
- *DB2 UDB EEE as an OLTP database? Absolutely!* from IBM Toronto Lab
- *TPC-D Benchmark Experience RS/6000 SP with DB2 Universal Database*
- `http://w3.aixncc.uk.ibm.com`
- `http://www.rs6000.ibm.com`
- `http://w3.aixncc.uk.ibm.com` (IBM Intranet)

7.5 Web server sizing

The purpose of this section is to give guidelines on how to choose and size IBM RS/6000 Web server machines. The information in this document

represents a set of guidelines that can be used to approximate the size of a server. This guide is only one of many resources available to assist in developing IBM Web server solutions. This chapter will not discuss other aspects of Web servers, such as security and guidelines on how to choose Web server software.

7.5.1 Introduction

Since NCSA's introduction of the Mosaic in 1993, the Internet has undergone incredible growth. For many people, the Internet is an important aspect of daily life today. It links abundant resources and information across the world and enables everybody to travel the Net (known as surfing) in a very simple way. It also introduces new ways to do business and makes online information accessible.

One of the Internet technologies that has been exploited widely is the World Wide Web. It enables us to see, search, and post information across the world. By implementing some new technologies such as Java, it even enables Web documents or information to be interactive.

Web technology is based on the Hypertext Transfer Protocol (HTTP). It is layered on top of TCP/IP in order to guarantee good data transfer. The machine that provides the HTTP service is usually called the Web server. This Web server can run on many platforms. On most UNIX machines, the process that provides HTTP service is called *httpd*. It usually runs as a UNIX daemon process, and normally uses and listens to TCP port number 80.

Most of the existing Web servers run on UNIX machines because TCP/IP is integrated with the UNIX operating system and many tools to support Web servers on UNIX are available as public domain or shareware software.

7.5.2 Sizing preparation

Common questions when choosing a Web server machine include:

- How big an RS/6000 will be needed for a Web server?
- How many hits per day can an RS/6000 handle?
- What is the maximum number of clients that can be supported by an RS/6000?

Before we can answer these questions, we need to answer:

- Is the Web server going to be an Internet or Intranet server?
- What is the potential demand for access to this site?
- What is the speed of the connection to the Internet or Intranet?
- How many pages will the system be serving?

- What is the average file size of the pages?
- Will the Web server be generating data for access?
- What kind of the web application is required?

7.5.3 Sizing factors

There are some important factors for sizing Web server. They are discussed in this section.

7.5.3.1 Target environment

When sizing a Web server, the most important consideration is the size of the target audience.

Internet

Sizing a Web server for the internet can be a very difficult task. The Internet includes millions of interconnected individuals who are navigating from one Web server to the next in search of information that has value to them. Sometimes it is very hard to estimate how popular a site may become. Usually, for initial implementation the Web server machine is chosen based on certain maximum accepted connections at an acceptable response time. Later, based on the average statistical log, the system can be expanded according to the growing usage. Therefore, if the growth factor is going to be considered, then upgrade ability and scalability of the Web server machine should be considered.

Intranet

Intranets are private nets that use the same standards and protocols as the public Internet. Intranets are rapidly displacing internal Web sites as the new network-centric corporate computing platform. An Intranet Web site dissolves all departmental, geographic, and technical boundaries by creating a universal way to connect people to people and people to information.

Sizing a Web server for an Intranet is considerably easier than sizing one for the Internet. The total number of potential users can be determined more accurately by using the total number of employees in the relevant department or the entire company.

7.5.3.2 Network bandwidth

In sizing the Web solution, it is important to understand the implications of the speed of the networking connection to the Web server. More often than not, many potential Web content providers are very focused on the vague hits-per-day quantity. The level of traffic that a particular Web server can support will be dependent on the server type, the content accessed on the

server and the speed of the connection of the server to the Intranet or Internet environment.

An Internet service provider (ISP) will deliver a connection of a defined speed. Three of the most common WAN speeds are ISDN (128 Kbps), T1 (1.544 Mbps), and T3 (45 Mbps). For an Intranet environment, common LAN speeds are 10 Mbps (over Ethernet) and 100 Mbps (over Fast Ethernet or FDDI). It is possible for the WAN or even the LAN to become the bottleneck of a high-performance Web site. This is especially common when non-Web traffic occupies the same network, degrading the site's performance. When this occurs, the network backbone must be scaled up to achieve maximum performance. Figure 100 shows the interrelationship between the average Web transaction size, the speed of the networking topology, and the maximum theoretical hits per second. To translate this into a number of hits for an approximately eight-hour peak usage period, multiply the hits per second by 28,800.

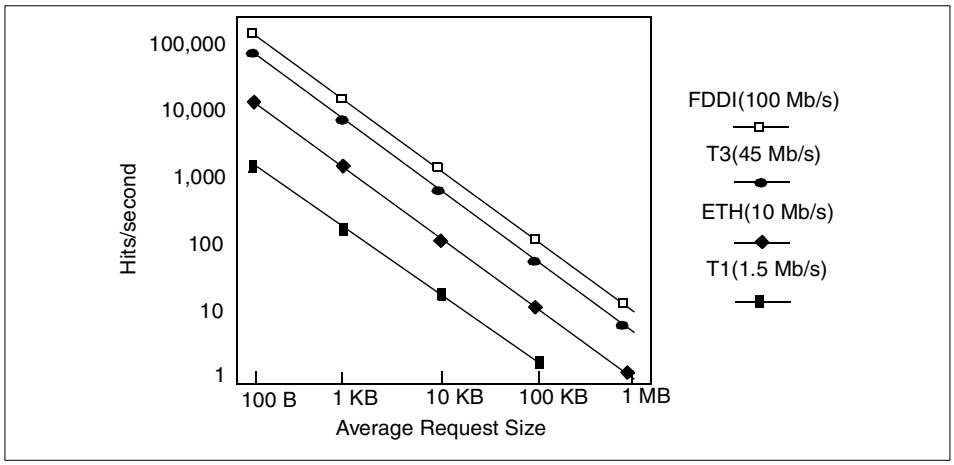


Figure 100. Relationship of Network Speed, Request Size and Maximum Hits

As the average Web transaction size increases, the maximum number of transactions decreases. Sites that plan on being mostly text-based will have average transaction sizes around 1 - 5 KB, while most well-designed sites with a mix of text and graphics intended for access by modem users handle transactions of about 10 KB each and sites with a substantial portion of multimedia content can exceed 100 KB per transaction.

7.5.3.3 Server content

The content being served will, to a large degree, dictate the overall performance of the site. A Web site's content ranges from text to graphics to

more complex multimedia file types such as video and audio. The type of content is closely related to the size and number of data transmissions, as text files are usually smaller. Graphics files are several times the size of text files, and multimedia data types are several times larger than graphics files.

It is not necessarily true that smaller files are better for performance because, if a Web server is required to serve large numbers of small files, the server performance may degrade. If the site is made up of large files, and most of the users are connected over low-bandwidth connections such as modems, the site's performance will be unacceptable to those users. The balance of the size and number of files required for a Web site must be considered as the appropriate server is chosen. The physical size of the Web content is important in determining the data storage requirements.

Pages can be static (they already exist on the server and are waiting to be requested) or dynamic (created on the fly by the Web server based upon the user's input, often in combination with the results of a database query). Most of the content that exists on the World Wide Web today is static (marketing-brochure type information, etc.), but the trend is toward interactive, dynamic Web sites. Sites that are integrated with other business applications or that are created for the purpose of doing e-business are, by nature, dynamic and interactive.

Obviously, dynamically created pages require more server power, and any interaction with a database engine also requires additional server processing power. We discuss sizing methods for the WebSphere and the Net.Commerce as typical examples of the dynamic Web sites. See Section 7.5.6, "Sizing WebSphere Application Server" on page 316 and Section 7.5.7, "Sizing Net.Commerce" on page 322.

7.5.3.4 User interaction, hits, and connections

Web sites can interact with users, usually by means of CGI (Common Gateway Interface) script programs, JAVA, and so on. The scripts capture user input and customize some aspect of what the user sees from that point on based on the user's input. The user's input can also be used to formulate database queries that retrieve specific information the user has requested from external databases that may reside on the server machine or on an attached machine. This information is then displayed on a dynamically created Web page. The programming involved in capturing user input as well as the processing power required to perform or send and receive a database query will have an effect on Web server performance.

The complexity of a page determines how many connections (hits) are required of the server. A page that consists of a single HTML text file would

require one hit. If that file also referenced three GIF graphics files, four hits would be required to serve the page. If the page includes a user input area, additional connections would be required, and so on. The number of connections required to serve a single page adds up quickly. This can be seen on some Web pages that count how many connections were made.

7.5.3.5 Number of clients

The number of simultaneous users of a site is very challenging to characterize. Unlike other types of client/server architectures, the weight of an individual client on the Web server is quite small and short lived. Connections to a Web server are traditionally stateless sessions that begin with an open from the client (a request for data), the server replies with data, and the session closes. Depending on the speed of the network connection, the size of the data requested and the server load, this session can last from less than a second to many seconds.

7.5.4 Web server performance

Web server performance measures several important areas that directly impact user experience and cost of ownership of the Internet solution. Usually, the following categories are used for the measurement:

- Response time
  This measures how long it takes the server to answer a client request. This is an important measurement to analyze, especially as the number of client connections increases and as the type of requests varies from static HTML to dynamic content creation using APIs and RDBMS. Usually, for a server to be fast, its average response time must be well under one second.
- Throughput
  Simply put, this measurement establishes the maximum amount of data the server can send through all open connections during a given amount of time. If the throughput is close to the bandwidth of the network (LAN or WAN), then the network is probably saturated.
- Connections per second
  This measures how many HTTP requests a server can establish, service, and then close during a given period for a specific set of HTML files. To adequately capture a server's performance in this area, the server must be tested against small, medium, and large static and dynamic HTML files and applications. This metric also appears as hits per day. However, connections per second are more helpful in planning for peak loads.
- Errors per second
  This measurement identifies how many HTTP requests were not serviced

or were dropped by a server. High error rates translate into an unreliable server that cannot handle the load at which the errors were generated. Ideally, no errors should occur.

7.5.5 Sizing IBM HTTP Server

In this section, sizing information for the IBM HTTP Server is provided. We do not discuss the installation, initial setup, security, and performance tuning. Also, we do not discuss sizing method for the network bandwidth.

We recommend using WebSphere or Net.Commerce if you need to use RDBMSs. For information about WebSphere, see Section 7.5.6, “Sizing WebSphere Application Server” on page 316. For information about Net.Commerce, see Section 7.5.7, “Sizing Net.Commerce” on page 322.

7.5.5.1 General considerations

We strongly recommend creating a cluster of machines. Large web sites workload patterns are characterized by bursts of activity. See Figure 101. Such ‘burstiness’ means that planning for average volumes is ineffective.

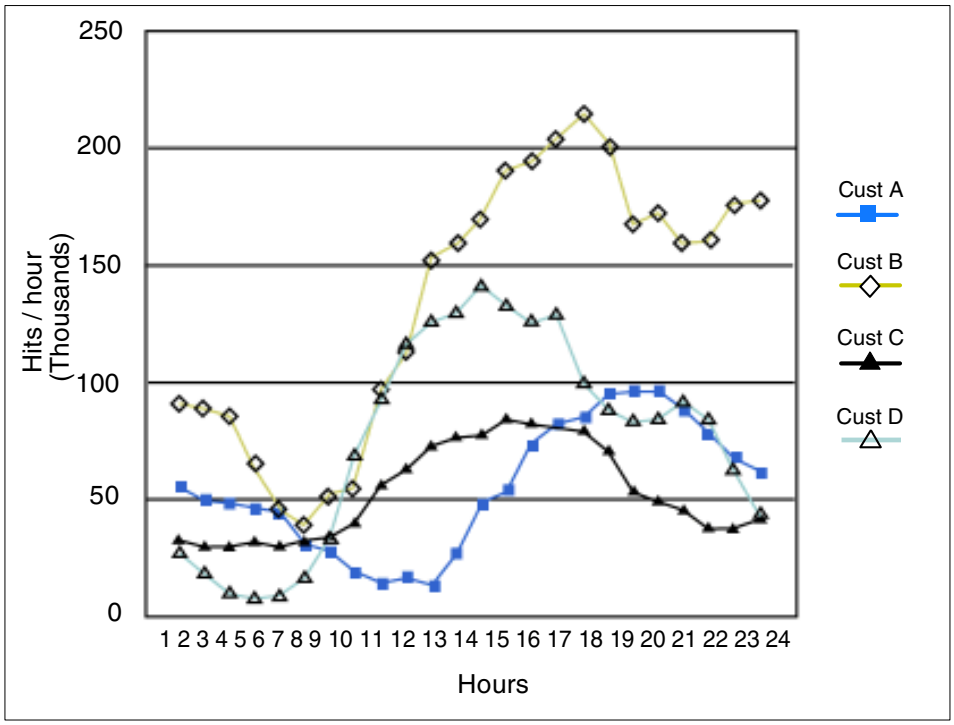


Figure 101. Some typical Web site loads over 24 hour period

You should prepare for unpredictable growth and the ability to have solutions ready for unknown problems. Planning clustered web servers is a good solution. For the Web server, the main technique for growth path is to add more machines. It is therefore appropriate to start with the expectation of more than one Web server with a dispatcher in front, such as the IBM SecureWay Network Dispatcher. Adding more machines then becomes painless and nondisruptive.

7.5.5.2 IBM HTTP Server

The IBM HTTP Server is an web server based on the Apache V1.3.3 with some additional adjustment. The IBM HTTP Server comes as a compiled and tested version of the Apache server for the specific platforms.

You can get the latest IBM HTTP Server from the IBM web site:

```
http://www.ibm.com/software/webservers/httpservers/
```

If you need to know about the Apache http server, you can get the information from:

```
http://www.apache.org/
```

For more information about the IBM HTTP Server, the redbook *IBM HTTP Server Powered by Apache on RS/6000*, SG24-5132 is helpful.

7.5.5.3 Estimate the workload

After studying your contents, estimate the workload as below:

- Peak hit rate (HR)
  The expected number of hits per hour to the server at peak. A hit is a request for a file (not a web page) to a web server. A file contains information such as text or an image. A web page is composed of one or more files. On average, a web page contains ten files. In other words, a request for a page could result in ten hits.
- Average file size (FS)
  An average size of a file based on an average web page. A web page consist of multiple files with various content such as text and images. The size of each file is based on its content. For example, a logo image is 47 KB to 68 KB, HTML is 6 KB to 800 KB, GIF is 8 KB to 1 MB, JPEG is 37 KB to 800 KB. If you have no information, use 4 KB.
- Complexity of dynamically generated pages (CF)
  Dynamic content usually refers to the execution of a sequence of instructions on the web-server to convert / format information from a data store system into HTML so that it can be viewed with a web browser.

Generally, rendering dynamic content to a browser requires significantly more processing power than static content from a filesystem. Define how many times resources needed compared to that for generating static pages. The CF of CGI is at least 15 and the CF of API is at least 2, but it depends on your application, so there is no recommendation.

- Percent of dynamically generated pages (DR)
  We are observing that dynamically generated pages ratio is between 0 and 90 percent.
- Size of disk space of contents (DS)
  The amount of disk space on the local web server required to house the content that is to reside on that web server.

You need to calculate the Adjusted Hit Rate (AHR) with the following formula:

```
AHR = HR * (1 - DR) + HR * DR * CF
```

Figure 102. Adjusted hit rate formula

- HR is the number of hits per hour of a peak period.
- DR is the ratio of dynamically generated pages.
- CF is complexity factor of dynamically generated pages.

7.5.5.4 Sizing processor

You can select the RS/6000 model with AHR. As a rule of thumb, a 1 Rel.OLTP machine can support 150 - 250 AHR. However, you should not select *one* large model that can support your AHR, as you need to prepare for unpredictable growth as discussed in Section 7.5.5.1, “General considerations” on page 312. We recommend clustered web servers. Clustered servers with 10 / 100 Mbps Ethernet adapter are better than a high performance server with Gigabit Ethernet adapter for scalability and availability.

Clustered web servers provide scalability. See Figure 103 on page 315. The eND is IBM SecureWay Network Dispatcher. The AFS server is IBM AFS Enterprise File System server. These IBM products provide file sharing and load balancing function to your web site. For more information about them, see *IBM WebSphere Performance Pack: Load Balancing with IBM SercureWay Network Dispatcher*, SG24-5858.

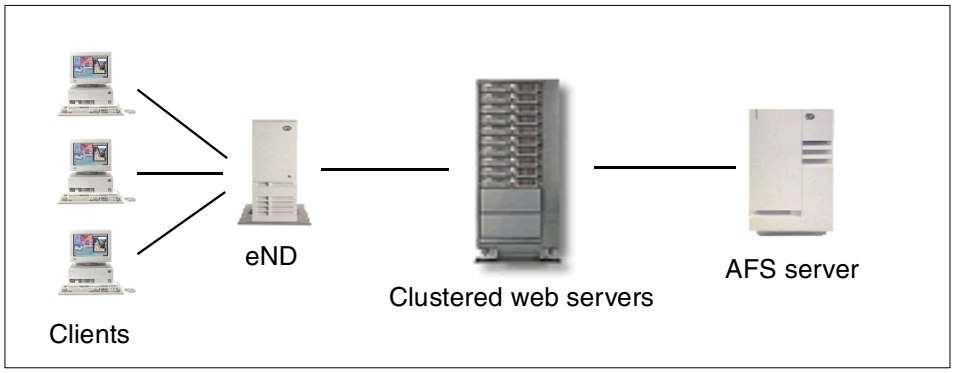


Figure 103. Clustered web servers

The clustered RS/6000 B50s may be a good solution for typical small to medium web sites. You just need to add more B50's when the workload increases.

7.5.5.5 Sizing memory

There is no information for sizing the amount of memory of IBM HTTP Server. The formula below is a recommendation from our test results.

```
Memory = 20 MB + max_child * 1.0 MB + application + AIX + file_cache
```

Figure 104. Memory formula for IBM HTTP Server

- max_child
  The number of maximum child processes.
- application
  The amount of memory which your applications need. Estimate the amount of memory for your applications carefully.
- AIX
  We recommend 64 MB memory for AIX.
- file_cache
  The amount of memory for file cache. At least 32 MB is recommended.

7.5.5.6 Sizing disk

There are 9.1 GB, 18.2 GB, and 36.4 GB disk drives available for RS/6000. You should decide which disk drive is good for your system, considering I/O

throughput. Generally, web contents are small files such as html and gif files, so having a few large disks has a tendency to create disk I/O bottle-necks.

```
RPF = (FS - 32) / 32 + 4
DBS = 32 + {(FS - 32) / 32 + 1} * 32
Disk I/O rate = AHR * RPF
Disk bandwidth = AHR * DBS
```

Figure 105. Disk workload formula

- RPF: disk read per file.

  If FS is 0 - 4 KB, RPF is 1. If FS is 4 - 16 KB, RPF is 2. If FS is 16 - 32 KB, RPF is 3.

- DBS: disk block size per file in KB

  If FS is 0 - 4 KB, DBS is 4 KB. If FS is 4 - 16 KB, DBS is 16 KB. If FS is 16 - 32 KB, DBS is 32 KB.

As rules of thumb, for disk I/O rate per disk we recommend the rate be below 85 reads / second. For disk bandwidth, we recommend the maximum bandwidth be 6 MB / second. After studying your contents carefully, you should consider the disk I/O balance to avoid *hot spots*. For example, common access counter data should be stored in separated disks.

7.5.6 Sizing WebSphere Application Server

This section provides an overview of concepts for sizing WebSphere Application Server (WAS) systems based on IBM RS/6000. There are some rules of sizing considerations for WebSphere Application Server on AIX.

7.5.6.1 The WebSphere e-Business benchmark (eBusBM)

The WebSphere Performance team has built a benchmark for characterizing performance of the WebSphere Application Server called the *WebSphere e-Business Benchmark* (eBusBM). The eBusBM was built to emulate an online brokerage firm. The application workload is a collection of Java classes, Java Servlets, Java Server Pages, and Enterprise Java Beans. For the purpose of this sizing method, workload complexity is measured relative to the performance of the eBusBM workload.

The business logic that is executed consists of:

1. Get input parameters for database access from form (userid, password, and so on.)

2. Connect to database using connection manager
3. Set transaction type and amount properties in account bean
4. Perform the transaction
    a. Build the SQL
    b. Execute the SQL
    c. Adjust the balance as per the client request
    d. Build the SQL
    e. Execute the SQL
5. Get new account information
6. Call the JSP to show account information

The complexity factor of this workload is 1 because all other workloads are measured relative to it.

7.5.6.2 Sizing methodology

The model which was used to develop the WAS sizing method was first outlined in *WebSphere Application Server Capacity Model*, in which a number of customer engagement scenarios are described where sizing information may be needed:

- For a known peak transaction rate (PTR) and transaction complexity factor (CF)
- For a known PTR, but unknown CF
- For a known CF but an unknown PTR

The laboratory test results provide us with PTR for eBusBM for selected models of the RS/6000. Using knowledge of RS/6000 processor relative scalability and other evidence from the benchmarks, it is then possible to construct a table of peak transaction rates for all RS/6000 models for the eBusBM workload. This table then gives us the capability to select, for any desired transaction rate, the RS/6000 server(s) that will provide the desired throughput.

You need to calculate the Adjusted Peak Transactions Rate (APTR) with the following formulas.

APTR = PTR * CF * PCC * SSLC

Figure 106. Adjusted peak transactions rate formula

- PTR supplied by customer or sized per other Formulas.
- CF supplied by customer or sized per other Formulas.

- PCC is the modifier to keep the server capacity from exceeding *X* percent.
- SSLC is the modifier for *X* percent capacity reduction due to SSL.

PTR = CV * SC * D * PF

Figure 107. Peak transactions rate formula

- CV defines the number of client visits (unique sessions) at the site per unit time.
- SC defines the number of pages a client will access per session. (A client profile needs to be defined in terms of pages / client / time.)
- D is the percent of dynamic content (The static content is not important from a server capacity question relative to dynamic content generation requirements).
- PF is the Peak Factor. It is used to estimate the peak workload requirements based upon the average workload. Five times is common.

7.5.6.3 CF (Complexity Factor)

The CFs are relative to the eBusBM benchmark workload. CF = 1 means the workload is the same as the eBusBM benchmark workload. Use the tables below for deciding your CF. See Table 35 below and Table 36 on page 319.

Table 35. CF table

Workload	Description	eBusBM ratio	CF
No transactional work	JDBC + JSP (with pre-built SQL)	1.5	0.65
More complex transactions	JDBC + JSP + complicated/heavy transactional workload	N/A	2.0
No transactional or JSP	JDBC (with pre-built SQL)	3.8	0.3
No transactional or JSP AND AIX commserver for SNA connection to dedicated S/390	JDBC + SQL (to DB2 on SNA connected S/390 server) (dedicated S/390)	2.1	0.48

Here is some test results using a 166 MHz 4 way F50 with WSAS AE V2.02. These data were tested on the previous version and a slow machine, but are helpful for estimating complexity factors.

Table 36. Test results

Configuration	Results [ops / sec]
JDBC request of single SQL to DB2 on TCP/IP AIX server	237
JSP retrieval	159
TxJSPDB2 to DB2 on TCP/IP AIX server*	62
JDBC request of single SQL to DB2 on SNA connected S/390 server (dedicated S/390)	130
JDBC request of single SQL to DB2 on SNA connected S/390 server (shared S/390)	60

Implications: JSP + JDBC (without transactions) are calculated as 1 / (latency of JSP + latency of JDBC), so JSP + JDBC (without transactions) are 62 / 95 (= round to 0.65) as complex as TxJSPDB2.
JDBC requests are 62 / 237 (round to 0.3) as complex as TxJSPDB2 (eBusBM).

* eBusBM is TxJSPDB2 to DB2 on TCP/IP AIX server.

7.5.6.4 Example

We provide some examples for estimating APTR in this section.

For a known PTR and CF

Let us suppose that your PTR is 50 ops / sec (operations / second) and your CF is 1 (means your business logic is the same as eBusBM business logic). If you decided that machine should not run beyond 75 percent of capacity, your PCC is 1.33 (100 / 75 = 1.33). If you do not use SSL, your SSLC is 1. In this case your APTR is:

```
APTR = PTR * CF * PCC * SSLC
     = 50 * 1 * 1.33 * 1
     = 66.5
```

Figure 108. APTR for a known PTR and CF

For a known PTR, but unknown CF

Let us suppose that your environment is a SNA attached dedicated S/390 DB server executing the same business logic as is found eBusBM. And we assume that PTR is 50 ops / sec, the machine should not run beyond 50 percent of capacity, and the degradation due to SSL is 20 percent. So the PCC is 2.0 (100 / 50 = 2.0), and the SSLC = 1.25 (1 + 0.2 / (1 - 0.2) = 1.25)

First, calculate the added latency to get to the S/390 over SNA. The latency of a SNA S/390 JDBC operation is 1 / 130 = 7.69 msec. And the latency of a TCP/IP to RS/6000 JDBC operation is 1 / 237 = 4.21 msec (see Table 36 on page 319). So, the added latency is 7.69 - 4.21 = 3.48 msec. There are two JDBC connections in eBusBM. Therefore you should double it to 6.96 msec.

Second, calculate the latency and throughput of the prescribed web application. The latency of the eBusBM on AIX is 1 / 62 = 16.13 msec. Then add the latency of different communication paths to get new latency. The new latency is 16.13 + 6.96 = 23.09 msec. Therefore, the throughput is 1 / 23.09 = 43 ops / sec.

Third, calculate the CF of your application. The CF is 62 / 43 = 1.44.

Finally, calculate the APTR:

```
APTR = PTR * CF * PCC * SSLC
     = 50 * 1.44 * 2.0 * 1.25
     = 180
```

Figure 109. APTR for a known PTR, but unknown CF

A high APTR value like this may need a clustered WAS server. In this case you should consider to use an eND (eNetwork Dispatcher). If you need the information of eND, *IBM SecureWay Network Dispatcher Version 2.1 White Paper* is helpful.

For a known CF but an unknown PTR

We assume the PCC is 1.18 (given that the machine should not run beyond 85 percent capacity), the SSLC is 1.25 (degradation due to SSL of 20 percent), and the CF is 1 (application business logic is the same as eBusBM).

Let us suppose that your peak load level is five times larger than the average load (means the PF is 5), you estimate 80 percent of pages are dynamic (means the D is 0.8), the number of visitor sessions per a day is 100,000, and the average number of pages per visitor session is 10 (hence the SC is 10).

The CV can be calculated as:

CV = 100,000 / 24 hours / 60 min / 60 sec = 1.16 ops / sec.

Therefore, the APTR can be estimated as:

```
PTR = CV * SC * D * PF
    = 1.16 * 10 * 0.8 * 5
    = 46.4

APTR = PTR * CF * PCC * SSLC
     = 46.4 * 1 * 1.18 * 1.25
     = 68.4
```

Figure 110. APTR for a known CF but an unknown PTR

Selecting the model

If you need to know which RS/6000 model meets your APTR, you can get the information from the IBM web site. It can be accessed from IBM intranet and it is available for Business Partners only.

- `http://solsrc.rs6000.ibm.com`

7.5.6.5 Performance considerations

The performance of WAS depends on the characteristics of the user applications running on the WAS server. Generally, an application is composed of both static and dynamic pages. The throughput difference between them can be two orders of magnitude.

If SSL encryption is applied to all contents, it results in an additional 20 percent penalty. Although key generation operations for SSL happen on the order of 10s - 100s of milliseconds, over the life of a client session this severe penalty is distributed over all the client interactions and the SSL encryption / decryption dominates the performance impact.

You should consider the tuning of the Web Server when you use WAS. Review your web server documentation for advice. If you use the IBM HTTP Server, more information can be found at:
`http://www-4.ibm.com/software/webservers/httpservers/doc/v136/misc/perf.html`

Enabling security on WAS has a significant impact on performance. You should enable it only it is needed. The default is *disabled*. When using security, the security cache time out can have a performance impact. The

default is 600 seconds. We recommend larger figures for better performance. The typical test at larger figures such as 6000 showed performance improvements up to 40 percent.

7.5.7 Sizing Net.Commerce

This section provides an overview of concepts for sizing Net.Commerce systems based on IBM RS/6000.

Net.Commerce V3.2 and V4.1 have similar core engine and performance characteristics.

7.5.7.1 General considerations

There are a lot of differences between a Net.Commerce site and a typical web site. The Net.Commerce utilizes some technologies to allow a merchant to provide a shopping experience for a user. They are server side Java, C++, EJB, Net.Data, and integration with back end server processes, coupled with DB2 UDB. Because the Net.Commerce uses RDBMS, you should consider using the RDBMS sizing method when you size your Net.Commerce server. For more information about RDBMS, see Section 7.4, “Database sizing” on page 291.

7.5.7.2 Sizing method for Net.Commerce

Let us show how to estimate your workload.

1. Study what the workload of your merchant typically looks like. Categorize it into two parts; browsing and buying transaction. If it is more appropriate to define a new type of workload such as transfer of heavy video clips, determine the weighting factor of that workload.
2. Convert the total workload into number of commands / second by counting the number of Net.Commerce commands that are executed by each transaction type. If you do not have any information to estimate number of Net.Commerce commands, use the values below:
   - A browsing transaction = 3 Net.Commerce commands
   - A buying transaction = 9 Net.Commerce commands
3. Determine the range of business hours when most transactions take place.
4. Calculate the peak time workload (commands / second).

Net.Commerce commands per second

Net.Commerce commands per second is the primary metric used for determining the size of a Net.Commerce site. A Net.Commerce command represents a request from the browser to the web server for a URL, which is

served by Net.Commerce / Net.Commerce Hosting Server (NCHS). Requests for images or other hits triggered by the response from Net.Commerce / NCHS should not be counted as Net.Commerce / NCHS command hits. For example, if your *ProductDisplay* page returns HTML that causes your browser to perform 8 more hits to obtain the needed images, you should only count this interaction as one Net.Commerce Command request.

Performance factors

It is important to understand that the following factors affect performance.

1. Dynamic and static page ratio
   Dynamic pages require substantially more resources than static pages do. Static pages can be ignored in capacity planning if the ratio of dynamic to static requests is fairly high, so we assume there is no request for purely static pages.
2. Average hits / page
   We assume one hit per page. If your site estimate is more hits per page, consider the weighting factor of that workload.
3. Page weight
   We assume 9 KB per page. If you are planning a site with significantly differing characteristics, be sure to compensate for your page size. You should not forget a page includes requests for the initial URL as well as all subsequent requests for images.
4. Database size
   The suggested small and medium configurations were tested using a store with one merchant, 10,000 shoppers, and 2,500 completed orders at the beginning of the test runs. The large configuration was tested using a store with one merchant, 100,000 products, and 25,000 completed orders at the beginning of test runs. You should consider the weighting factor of that workload for your site.
5. Network bandwidth
   Generally, between the web server and database server. Internet communication on average will be significantly slower than a 100 Mbps Ethernet.
6. Browsing and buying ratio
   The ratio of browsing to buying of 95 : 5 is representative of most retail shopping sites. In general, buying commands have lower throughput than browsing commands. If the site you are planning will have a significantly different browser / Buyer Conversion Rate, be sure to compensate when using the listed recommendations.
7. Net.Commerce cache
   We assumed every system was well tuned and set on the Net.Commerce cache. There is a significant performance difference caused by the cache. We recommend that the Net.Commerce cache be turned on.

8. Workload modeling
   The following values for a typical implementation of Net.Commerce / NCHS are based on empirical data from a set of benchmarks and customer test cases.
   - A browsing transaction = 3 Net.Commerce commands
   - A buying transaction = 9 Net.Commerce commands

   The above factors are always susceptible to change according to the design of each transaction. Count the number of Net.Commerce commands for each transaction type. The following lists the Net.Commands used in our sample tests.

   - Browsing transaction:
     - ExecMacro (Home page, Search page)
     - ProductDisplay
     - CategoryDisplay
   - Buying transaction:
     - ExecMacro (Order accept, inquiry)
     - OrderDisplay
     - OrderItemUpdate
     - OrderItemDisplay
     - RegisterForm
     - RegisterNew
     - Logon
     - AddressUpdate
     - OrderProcess

Net.Commerce hosting server

The number of merchants defined in each configuration does not affect performance directly. The number of merchants you can host in a site varies inversely with the number of commands executed per unit of time per merchant; in other words, with the amount of workload. For example, a site that can serve 100 Net.Commerce commands per second can host 500 merchants that each receive (100 / 500) = 0.2 Net.Commerce command requests per second. The same site would roughly be able to host 100 merchants that each received (100 / 100 =) 1 Net.Commerce command request per second.

7.5.7.3 Selecting the hardware model

The information on which hardware model to select can be obtained from the following IBM web sites:

- http://http://www-4.ibm.com/software/webservers/commerce/servers/downloads/config-plan.pdf
- For Business Partners:
  http://solsrc.rs6000.ibm.com

7.5.8 Resources

- *WSAS Implementation Planning* by Ed Merenda
- http://w3.aixncc.uk.ibm.com/
  (IBM internal URL providing online sizing tools.)
- *WebSphere V3 Performance Tuning Guide*, SG24-5657
- IBM SecureWay Network Dispatcher Version 2.1 White Paper
- http://powernet.austin.ibm.com, IBM internal
- http://solsrc.rs6000.ibm.com, registered Solution Source Business Partner only
- *IBM HTTP Server Powered by Apache on RS/6000*, SG24-5132
- http://www-106.ibm.com/software/developerworks/library/scalability/
- *IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher*, SG24-5858

7.6 Lotus Domino Server sizing

This section provides an overview of concepts for sizing Lotus Domino Servers using the IBM RS/6000.

There are several guidelines for sizing a Lotus Domino Release 5 infrastructure on AIX Version 4.3. The information presented in this document is based on *rules of thumb* that provide generic, conservative capacity planning estimates.

Attention

You will find some information such as NotesMark from hardware vendors, but they are inappropriate for sizing your Domino server. Hardware vendors tune their machines for benchmarking, and the maximum number of concurrent active users is inconsistent with reality.

7.6.1 Estimate the workload

The first step is to understand the type of workload and the number of users.

Specifically, the following requirements should be considered:

- Type of server
  - Notes Mail or Notes application server
  - Web / IMAP / POP Mail or Web application server
- Number of registered users
- Number of active users during peak time

No two users perform work in the same manner. For example, there may be casual users and there may be power users. However, across most companies the number and types of users are fairly consistent. Furthermore, the skill level for users is constantly changing, with novice users becoming more advanced as they become familiar with the products.

For these reasons, the two main factors in sizing the a Domino infrastructure is the number of peak concurrent users at any given moment (determines processor/memory requirements) and total number of registered users (determines disk storage requirements).

7.6.2 Processor sizing

The number of processors, and consequently machines, can be determined by dividing the estimated number of peak concurrent users by one of the values listed in the table below. The number of processors per system can impact the number of concurrent users and the number of Domino partitions (individual Domino servers) that should be implemented on a system.

Table 37. Recommended number of concurrent users for Domino R5

Number of processors	Number of partitioned servers	Number of concurrent users		
		R5 Mail	Web Mail	IMAP / POP Mail
1	1	600 - 900	150 - 220	420 - 720
2	1	900 - 1200	220 - 300	630 - 960
4	1	900 - 1200	220 - 300	630 - 960
4	2	1300 - 1500	320 - 370	910 - 1200
4[a]	2	n/a	500	1400

Number of processors	Number of partitioned servers	Number of concurrent users		
		R5 Mail	Web Mail	IMAP / POP Mail
4[a]	1	2000	n/a	n/a
8[a]	2	4000	n/a	n/a
[a] These results only apply to F80, H80, and M80 RS/6000 servers				

Despite the matter of fact that there might be a CPU bound situation on an SMP machine, you may find that the CPU resources are not fully consumed. The reason is that the current implementation of Lotus Domino is not able to efficiently use all the CPU resources for multi processor systems. The Domino R5 architecture has, however, been improved over Release 4 so the application can now scale to four processors instead of two.

In order to avoid the situation described above, the guidelines are:

- No more than four processors per Domino server for Domino R5.
- No more than two Domino servers per three processors.

Some Lotus Domino Server environments require high availability and Internet access, which are not discussed in detail in this redbook. The following might, however, serve you as a reference:

- If high availability is required, the use of products such as Domino Server Clustering or HACMP is recommended. For greater detail refer to the redbook *High Availability and Scalability with Domino Clustering and Partitioning on AIX*, SG24-5163
- If Web access is desired the Internet Cluster Manager (ICM) is recommended to be used. ICM is the task which manages failover and load balancing of Web client requests. For more information about ICM refer to the redbook: *Lotus Domino R5 on IBM RS/6000: Installation, Custumization, and Administration*, SG24-5138.

7.6.3 Memory sizing

As with most sizing methods, there are multiple approaches and recommendations to determine the amount of memory required for Lotus Domino Server.

As a starting point, always refer to the release notes for the version of Domino that you are planning to use. Beyond that, the following recommendations can serve as a guideline.

For small servers (< 400 concurrent users):

- Basic memory for AIX should be sized at 64 MB.
- The minimum Domino Server requirement is 128 MB for Domino and 0.5 MB per concurrent user.

For large servers (> 400 concurrent users):

- Basic memory for AIX should be sized at 64 MB.
- Each Domino partition should be sized at 2 GB.

Table 38. Domino R5 recommended memory size for large servers

Number of processors	Number of partitioned servers	Memory per system
1	1	2 GB
2	1	2 GB
4	1	2 GB
4	2	4 GB
8	2	4 GB

Notes

Sizing a Domino application server depends on how complex the user application is. There is no specific formula to determine the complexity.

R5 Mail is one of the more complex Domino applications. Therefore, it is recommended to use the R5 Mail server as a baseline for sizing in order to get an adequate level of complexity.

Paging space

The general recommendation for sizing the paging space is discussed in Section “Paging spaces” on page 275.

A common practice is to create paging spaces that are at least twice the size of real memory for up to 512 MB real memory.

For memory sizes larger than 512 MB, the creation of paging spaces that are the same size as real memory is recommended.

7.6.4 Disk sizing

The following disk sizing recommendations can be used as a rule of thumb:

- A minimum of 160 MB of disk space is required for the binaries.
- For mail, add an additional 50 - 100 MB of disk space for each registered mail user.

The rest of the database must be sized as appropriate for the individual situation. Keep in mind that the disk space needed for Lotus Notes Server will increase as users become familiar with it, so you should try to have a few slots available for later evolutions for additional disk drives in the future.

As a general recommendation, you should not put any two or more of AIX, Notes binaries, Mail files, or DB files on the same hard disk. If some DB files are very update-intensive, you should locate them on separate disks, no matter how empty these drives might be.

Concerning SCSI buses, initially choose Ultra2 SCSI. The best choice for performance are SSA disks.

The usage of integrated disk storage systems, such as the IBM Enterprise Storage Server (ESS), can also be taken into consideration.

7.6.5 Example

The following is a sizing example for 5,000 registered users and 4,000 simultaneous connecting active users during peak times.

1. Estimate the number of simultaneous connecting users for peak times.
   - R5 Mail users = 3,000
   - Web Mail users =1,000
2. Select the processor by using the recommended number of concurrent Domino R5 Mail users from Table 37 on page 326.

   Because the number of R5 mail users is 3000, a 8-way SMP server (F80, H80, or M80) with two partitions may be chosen as the Domino server.

   For Web Mail, two 4-way SMP models F80, H80, or M80, with two partitions each can serve as Web Mail Domino servers.
3. Estimate the memory using the method described in Section 7.6.3, “Memory sizing” on page 327.

   The memory requirement for the R5 Mail server is 4 GB, plus 64 MB for the operating system.

The memory requirement for each of the Web mail servers is 4 GB, plus 64 MB for the operating system.

4. Estimate the disks.
   As a generous estimate, allow 600 MB per server for the operating system.

   R5 Mail server

   Supposing an average mail box size is 80 MB, the total disk space requirement for the mail boxes is 80 MB * 3000 users = 240 GB + 600 MB for AIX.

   This space can be supplied with either 27 x 9.1 GB disks or 14 x 18.2 GB disks.

   Web Mail server

   Use the same estimates as for R5 mail users. The total amount of disk space can be calculated together, but the disk pools will have to be divided onto two servers.

   If an average mail box size is 80 MB, the total disk space requirement for the mail boxes is 80 MB * 1000 users = 80 GB + 1200 (2 x 600) MB for AIX.

   This means either 5 x 9.1 GB disks for each server or 3 x 18.2 GB disks for each server.

   If AIX LVM mirroring is required, the number of disks has to be doubled. For performance, using 9.1 GB SSA disks is recommended because it helps distribute the workload. For a lower number of disks, consider using 18.2 GB or 36.4 GB SSA disks.

These estimates represent the bare minimum of resources to get. If other applications are being used, such as Lotus Notes databases, you would have to take this into consideration as well.

A Lotus Domino Capacity Sizing tool is available on the IBM intranet. IBM Business Partners can access the tool through the IBM Business Partner web pages.

7.6.6 Conclusion

These sizing recommendations are based on rules of thumb. They are meant to give some guidelines for sizing a Lotus Notes Server R5 on AIX 4.3. We strongly recommend closely monitoring your Notes environment and adjusting the size of memory, disks, and machine models.

7.6.7 Resources

- *Lotus Domino R5 on IBM RS/6000: Installation, Customization, and Administration*, SG24-5138
- *NotesBench Disclosure Report for the IBM RS/6000 Enterprise Server S80 with Lotus Domino R5.0.2 on AIX V4.3.3* from:
  `http://www.notesbench.org`
- `http://www.rs6000.ibm.com/resource/technology/notesSPcfg`
- `http://solsrc.austin.ibm.com`, IBM/Lotus internal use only
- `http://solsrc.rs6000.ibm.com`, Internet access for Business Partners
- *Resource Tuning of Lotus Domino on AIX: Quick Reference* from:
  `http://www.lotus.com/performance`
- *High Availability and Scalability with Domino Clustering and Partitioning on AIX*, SG24-5163

Chapter 8. Performance tools

Due to its UNIX heritage, AIX provides a powerful set of performance-monitoring and tuning tools. The available tools give performance information for different components of the system and various parameters that affect performance. The details of the syntax and functions of these commands are documented in the *Performance Management Guide.* Before using these tools, a few concepts should be clarified.

What is a performance bottleneck?
A performance bottleneck is the slowest performing component in a computer environment that causes a reduction in the total throughput of the system. This can either be a resource on the system, for example CPU, memory, or disk, or it could be the network. In the case of the network, it could be a different system on the network that causes a performance problem on the local system. There is always be a bottleneck because some resource will always be the slowest.

How to determine a performance bottleneck.
When dealing with performance issues, it is important to check the following subsystems:

- CPU
- Memory
- Disk I/O
- Network

It does not matter in which order you investigate these, but it is important that all four subsystems are checked for bottlenecks. Once a bottleneck has been eliminated, another bottleneck may be created in another area of the system; for this reason, you must rechecks all the subsystems several times until all the bottlenecks have been removed or the performance has been resolved.

8.1 AIX performance tools and commands

AIX provides several monitoring tools to determine performance bottlenecks and to tune the system. Not all of these tools are supplied with the AIX Base Operating System. Some of them are part of the AIX Performance Toolbox software, as shown in Table 41 on page 335.

For detailed information on how these commands work, please refer to the *Performance Management Guide.*

Enhancements have been made to the following commands:

- fdpr
- netstat
- no
- nfso
- rmss
- svmon
- vmstat
- vmtune

Refer to Section 8.1.3, “Command descriptions” on page 336 for details.

8.1.1 Commands viewed by filesets

The following tools are supplied on the AIX installation media. The bos.acct and bos.rte filesets in Table 39 allow you to have a first look at where your bottlenecks may be. The other filesets allow you perform a second level performance diagnosis. Some of them allow you to perform performance tuning.

trace, trcrpt, and prof are 64-bit enabled.

Note

It is important to use 64-bit performance monitoring and turning tools on 64-bit systems.

Table 39. Performance commands supplied on AIX installation media

Fileset	Commands
bos.acct	iostat, vmstat, sar, timex
bos.rte	ps, lsps, lsattr, lsdev, lspv, lslv, lsvg, chdev, chlv, mkps, chps, migratepv, reorgvg, nice, renice, wlmstat
bos.sysmgt	trace, trcrpt, pstat
bos.net.client	netstat, nfsstat, ifconfig, no, nfso
bos.net.server	iptrace, ipreport, tcpdump
bos.perf	Performance Diagnotic Tool (PDT)
bos.adt.prof	prof, grof
bos.adt.samples	vmtune, emstat, schedtune

The following tools in Table 40 are supplied as part of the *Performance Agent*. They are installation specific and therefore cannot be copied from one system to another without risking unpredictable results or a system crash.

`filemon`, `fileplace`, `netpmon`, `svmon`, `tprof`, `fdpr`, `stripnm`, `genkld`, `genld`, and `genkex` are all 64-bit enabled. There is no 64-bit support for bf and bfrpt.

Table 40. Command supplied by Performance Agent

Fileset	Commands
perfagent.tools	`filemon, fileplace, netpmon, svmon, tprof, rmss, syscalls, lockstat, fdpr, stem, bf, bfrpt, stripnm, genkld, genld, genkex, topas`

The following tools in Table 41 are supplied as part of *Performance Toolbox for AIX*. This is a Licensed Program Product (LPP) that needs to be ordered from IBM separately.

There is no 64-bit support for Xprofiler

Table 41. Filesets supplied with Performance Toolbox

Fileset	Commands
perfmgr.local (Performance Toolbox local)	`xmperf`
perfmgr.network (Performance Toolbox Network)	`xmperf, chmon`
Tools from the Web	monitor, PV, xgprof, Xprofiler

8.1.2 Commands viewed by system resource

The following commands are useful for determining where bottlenecks are occurring on your system.

Table 42. View by system resources

CPU	MEMORY	I/O	Network
`vmstat, iostat`	`vmstat`	`iostat`	`lsattr`
`ps`	`lsps`	`vmstat`	`netstat`
`sar`	`svmon`	`lsps`	`nfsstat`
`gprof/prof/tprof`	`filemon`	`lsattr`	`netpmon`
`time/timex`	`bf, bfrpt`	`lsdev`	`ifconfig`

CPU	MEMORY	I/O	Network
netpmon	wlmstat	lspv/lslv/lsvg	iptrace/ipreport
wlmstat	topas	fileplace	tcpdump
topas		filemon	
stem		wlmstat	
syscalls			
lockstat			
emstat			
Performance Toolbox	Performance Toolbox	Performance Toolbox	Performance Toolbox
trace, trcrpt, utld	trace, trcrpt	trace, trcrpt	trace, trcrpt
nice/renice	vmtune	vmtune	no
schedtune	chps/mkps	chdev	nfso
bindprocessor	fdpr	migratepv	chdev
chdev	chdev	chlv	ifconfig
pstat	rmss	reordvg	
fdpr	schedtune		

8.1.3 Command descriptions

The following are descriptions of the performance tuning commands.

bf

The bf (bigfoot) command traces the memory use of the applications. With the back-end program bfrpt, it provides a detailed trace, or footprint, of the memory page references for most processes on the system. It captures references to all unpinned pages and many pinned pages.

bindprocessor

The `bindprocessor` command binds or unbinds the kernel threads of a process to a specifics processor, or lists available processors. The *Process* parameter is the process identifier of the process whose threads are to be bound or unbound, and the *ProcessorNum* parameter is the logical processor number of the processor to be used. If the ProcessorNum parameter is omitted, the process is bound to a randomly selected processor.

It is important to understand that a process itself is not bound, but rather its kernel threads are bound. Once kernel threads are bound, they are always scheduled to run on the chosen processor unless they are later unbound. When a new thread is created, it has the same bind properties as its creator. This applies to the initial thread in the new process created by the fork subroutine; the new thread inherits the bind properties of the thread called fork. When the exec subroutine is called, thread properties are left unchanged.

The `-q` flag of the bindprocessor command lists the available logical processor numbers; you can use the logical numbers given as values for the *ProcessorNum* parameter. The `-u` flag unbinds the threads of a process, allowing them to run on any processor.

Note

The bindprocessor command is meant for multiprocessor systems. Although it will also work on uniprocessor systems, binding has no effect on such systems.

You need root authority to bind or unbind threads in processes you do not own.

AIX V4 allows users to bind processes to a specific processor by using the `bindprocessor` command. Although the bindprocessor command is intended for tuning, it is very useful for performance analysis on SMP systems. The bound process will run only on the designated processor. If the process is multi threaded, all the related threads will be bound to the same processor.

Binding is different from partitioning; it is not possible, for example, to dedicate a set of processors to a specific workload and another set of processors to another workload.

It is not possible to bind a thread to a specific processor. You can bind one or several threads within a process at the programming level by using the

bindprocessor(What,Who,Processor) call. The bindprocessor() call must be used in the source code of your program.

chdev
The chdev command changes the characteristics of the device specified with the given device logical name (the -l name flag).The device can be in the Defined, Stopped, or Available state. Some changes may not be allowed when the device is in the Available state. When changing the device characteristics, you can supply the flags either on the command line or from a specified file parameter.

chlv
Changes only the characteristics of a logical volume.

chmon
This command allows collection of vital statistics from a character terminal, and is available as a C sample code with the Performance Toolbox Network LPP. See also monitor.

chps
The chps command changes attributes of a specific paging space.

To change the size of a Network File System (NFS) paging space, the size of the file that resides on the server must first be changed and then the swapon command used to notify the client of the change in size of the paging space.

cpu_state
The cpu_state command shows the number of processors and the current state of each processor within the system. A processor may be enabled, disabled, or unknown.

CURT
CURT supersedes utld with AIX V4.3.3.

As utld, CURT is a tool that takes an AIX trace file as input and produces a number of statistics related to CPU utilization and process/thread activity. It works with both uniprocessor and multiprocessor AIX traces.

Unlike utld, lock statistics have been excluded from CURT and incorporated into SPLAT, which by the time of the production of this redbook was not publically available yet.

CURT can be obtained from the following Web site:

ftp://ftp.software.ibm.com:/aix/tools/perftools

emstat
The command allow you to detect emulated instruction on your system.

fdpr
The `fdpr` command (feedback directed program restructuring) is a performance-tuning utility that may help improve the execution time and the real memory utilization of user-level application programs. The `fdpr` command optimizes the executable image of a program by collecting information on the behavior of the program while it is used for some typical workload. The `fdpr` command then creates a new version of the program that is optimized for that workload. The new program generated by `fdpr` will typically run faster and use less real memory.

`fdpr` has been updated in AIX4.3.2 to provide enhancements to ease debugging optimized executables and to optimize shared libraries.

Attention

The new executable will not be supported by IBM.

You should run a full test cycle for the new executable to guarantee the same functionality. Performance will change when the initial workload, data, or parameters change.

filemon
The `filemon` command collects and presents trace data on the various layers of the file system, and reports the I/O activity on behalf of logical files, virtual memory segments, logical volumes, and physical volumes. It is used to check for an I/O bottleneck on your system.

To provide a more-complete understanding of file system performance for an application, the command monitors file and I/O activity at four levels:

- Logical file system

  The `filemon` command monitors logical I/O operations on logical files. The monitored operations include all *read*, *write*, *open*, and *lseek* system calls, which may or may not result in actual physical I/O, depending on whether or not the files are already buffered in memory. I/O statistics are kept on a per-file basis.

- Virtual memory system

  The `filemon` command monitors physical I/O operations (such as paging) between segments and their images on disk. I/O statistics are kept on a per-segment basis.

- Logical volume

  The `filemon` command monitors I/O operations on logical volumes. I/O statistics are kept on a per-logical-volume basis.

- Physical volumes

  The `filemon` command monitors I/O operations on physical volumes. At this level, physical resource utilizations are obtained. I/O statistics are kept on a per-physical-volume basis.

In its normal mode, the `filemon` command runs in the background while one or more application programs or system commands are being executed and monitored. The filemon command automatically starts and monitors a trace of the program's file system and I/O events in real time.

Like `tprof`, the `filemon` command uses the AIX system trace facility. Currently, the trace facility only supports one output stream. Consequently, only one `filemon` or `trace` process can be active at a time. If another `filemon` or `trace` process (like `tprof`) is already running, the `filemon` command will respond with an error message.

The performance impact of `filemon` depends on how low or high the I/O rate on the system is. The CPU consumption can range from 1 percent at a low I/O rate to 5 percent at a high I/O rate.

fileplace

The `fileplace` command displays the placement of blocks for a specified file within the AIX logical or physical volumes containing the file. It will most likely be used if the `filemon` command displays large seek distances.

Optionally, the `fileplace` command will also display:

- Statistics indicating the degree to which the file is spread within the volume.
- The indirect block addresses for the file.
- The file's placement on physical (as opposed to logical) volume for each of the physical copies of the file.

Most variations of this command use less than 0.3 seconds of CPU time.

Filemon supports large files for 32-bit and 64-bit kernels.

genkex

The `genkex` command extracts the list of kernel extensions currently loaded

onto the system and displays the address, size, and path name for each kernel extension in the list.

For kernel extensions loaded onto the system, the kernel maintains a linked list consisting of data structures called loader entries. A loader entry contains the name of the extension, its starting address, and its size. This information is gathered and reported by the genkex command.

genkld
The `genkld` command extracts the list of shared objects currently loaded onto the system, and displays the address, size, and path name for each object on the list.

For shared objects loaded onto the system, the kernel maintains a linked list consisting of data structures called loader entries. A loader entry contains the name of the object, its starting address, and its size. This information is gathered and reported by the `genkld` command.

genld
The `genld` command collects the list of all processes currently running on the system, and optionally reports the list of loaded objects responding to each process.

gprof
The `gprof` command produces an execution profile of C, Pascal, FORTRAN, and COBOL programs. The effect of called routines is incorporated into the profile of each caller. The gprof command is useful in identifying how a program consumes CPU resource. To find out which functions (routines) in the program are using the CPU, you can profile the program with the gprof command.

The profile data is taken from the call graph profile file (gmon.out by default) created by programs compiled with the cc command using the `-pg` option. The `-pg` option also links in versions of library routines compiled for profiling, and reads the symbol table in the named object file (a.out by default), correlating it with the call graph profile file. If more than one profile file is specified, the `gprof` command output shows the sum of the profile information in the given profile files.

The `-pg` option causes the compiler to insert a call to the mcount subroutine into the object code generated for each recompiled function of your program. During program execution, each time a parent calls a child function the child calls the mcount subroutine to increment a distinct counter for that parent-child pair. Programs not recompiled with the `-pg` option do not have the

mcount subroutine inserted, and therefore keep no record of who called them.

Note: Symbols from C++ object files have their names demangled before they are used.

The `gprof` command produces three items:

- First, a flat profile is produced similar to that provided by the `prof` command. This listing gives total execution times and call counts for each of the functions in the program, sorted by decreasing time. The times are then propagated along the edges of the call graph. Cycles are discovered, and calls into a cycle are made to share the time of the cycle.
- A second listing shows the functions sorted according to the time they represent, including the time of their call-graph descendents. Below each function entry are its (direct) call-graph children, with an indication of how their times are propagated to this function. A similar display above the function shows how the time of the function and the time of its descendents are propagated to its (direct) call-graph parents.
- Cycles are also shown, with an entry for the cycle as a whole and a listing of the members of the cycle and their contributions to the time and call counts of the cycle.

Profiling using the `gprof` command is problematic if your program runs the fork or exec subroutine on multiple, concurrent processes. Profiling is an attribute of the environment of each process, so if you are profiling a process that forks a new process, the child is also profiled. However, both processes write a gmon.out file in the directory from which you run the parent process, overwriting one of them. The tprof command is recommended for multiple-process profiling.

If you must use the `gprof` command, one way around this problem is to call the chdir subroutine to change the current directory of the child process. Then, when the child process exits, its gmon.out file is written to the new directory.

ifconfig

The `ifconfig` command configures or displays network interface parameters for a network using TCP/IP.

iostat

The `iostat` tool reports CPU statistics and input/output statistics for TTY devices, disks, and CD-ROMs. It is used for monitoring system input/output device utilization by observing the time the physical disks are active in

relation to their average transfer rates. The iostat command is useful to determine whether a physical volume is becoming a performance bottleneck and if there is a way to improve the situation.

The generated reports can be used to change system configurations to better-balance the input/output load between physical disks.

Because the CPU utilization statistics are also available with the iostat report, the percentage of time the CPU is in I/O wait can be determined at the same time. For multiprocessor systems, the CPU values are global averages among all processors. Also, the I/O wait state is defined system-wide, and not per processor.

The iostat command adds little overhead to the system. It uses about 20 milliseconds of CPU time for each report generated.

Like vmstat, iostat cannot be used to finally decide whether there is a performance bottleneck. The system administrator will have to use more-complex tools like filemon to identify the source of the slowdown.

ipreport
The ipreport command generates a trace report from the specified trace file created by the iptrace command.

The LogFile parameter specifies the name of the file containing the results of the Internet Protocol trace. This file is created by the iptrace command.

iptrace
The iptrace daemon records Internet packets received from configured interfaces. Command flags provide a filter so that the daemon traces only packets meeting specific criteria. Packets are traced only between the local host on which the iptrace daemon is invoked and the remote host. The LogFile parameter specifies the name of a file to which the results of the iptrace command are sent. To format this file, run the ipreport command.

lockstat
Note: To enable lock statistics collection, the bosboot -L command must be executed.

The lockstat command reports statistics about contention in the operating system among simple and complex kernel locks.

Reports generated by the lockstat command can be used to ensure that system performance is not being reduced by excessive lock contention.

The lockstat command generates a report for each kernel lock that meets all the specified conditions. If no condition values are specified, default conditions are used. The reports give information about the number of lock requests for each lock. A lock request is a lock operation (such as taking or upgrading a lock) which in some cases cannot be satisfied immediately. A lock request that cannot be satisfied at once is said to be blocked. A blocked request will either spin (repeatedly execute instructions that do nothing) or sleep (allowing another thread to execute).

The column headings in the lockstat command listing have the following meanings:

- Subsys
  The subsystem to which the lock belongs.
- Name
  The symbolic name of the lock class.
- Ocn
  The occurrence number of the lock in its class.
- Ref/s
  The reference rate, or number of lock requests per second.
- %Ref
  The reference rate expressed as a percentage of all lock requests.
- %Block
  The ratio of blocking lock requests to total lock requests. A block occurs whenever the lock cannot be taken immediately.
- %Sleep
  The percentage of lock requests that cause the calling thread to sleep.

In AIX, you can use the `lockstat` command to see the use of locks. Only kernel locks can be seen with the `lockstat` command.

The `lockstat` command supports the use of user-supplied lock names in files named */usr/include/sys/lockname_*.h*, where `*` is a wildcard.

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 lock contention.

In AIX V4, this possibility should be investigated with the `lockstat` command.

Look for the following pointers:

- Check `lockstat` output for Ref/s > 10000

- Identify subsystems and lock classes that have a high number of Ref/s

Application problems can only be seen indirectly. If there is lock contention, you must check for bottlenecks caused by 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 lock contention.

lsattr
The `lsattr` command displays information about the attributes of a given device or kind of device. If you do not specify the device logical name (`-l` Name), you must use a combination of one or all of the `-c` Class, `-s` Subclass, and `-t` Type flags to uniquely identify the predefined device.

lsdev
The `lsdev` command displays information about devices in the Device Configuration database. You can display information about all devices in the Customized Devices object class using the `-C` flag. Any combination of the `-c` Class, `-s` Subclass, `-t` Type, `-l` Name, and `-S` State flags selects a subset of the customized devices. You can display information about all devices in the Predefined Devices object class using the `-P` flag. Any combination of the `-c` Class, `-s` Subclass, and `-t` Type flags selects a subset of the predefined devices.

lslv
The `lslv` command displays the characteristics and status of the Logical Volume or lists the logical volume allocation map for the physical partitions on the Physical Volume. The logical volume can be a name or identifier.

lsps
The `lsps` command displays the characteristics of paging spaces, such as the paging space name, physical volume name, volume group name, size, percentage of the paging space used, whether the space is active or inactive, and whether the paging space is set to automatic. The Paging Space parameter specifies the paging space whose characteristics are to be shown.

For NFS paging spaces, the physical volume name and volume group name will be replaced by the host name of the NFS server and the path name of the file that is used for paging.

lspv
The `lspv` command displays information about the physical volume if the specific physical volume name is specified. If you do not add flags to the lspv

command, the default is to print every known physical volume in the system along with its physical disk name, physical volume identifiers (PVIDs), and which volume group (if any) it belongs to.

lsvg
The `lsvg` command displays information about volume groups. If you use the VolumeGroup parameter, only the information for that volume group is displayed. If you do not use the VolumeGroup parameter, a list of the names of all defined volume groups is displayed. When information from the Device Configuration database is unavailable, some of the fields will contain a question mark (?) in place of the missing data. The lsvg command attempts to obtain as much information as possible from the description area when the command is given a logical volume identifier.

migratepv
The `migratepv` command moves allocated physical partitions and the data they contain from the SourcePhysicalVolume to one or more other physical volumes. To limit the transfer to specific physical volumes, use the names of one or more physical volumes in the DestinationPhysicalVolume parameter; otherwise, all the physical volumes in the volume group are available for the transfer. All physical volumes must be within the same volume group. The specified source physical volume cannot be included in the list of DestinationPhysicalVolume parameters.

mkps
The `mkps` command adds additional paging space to the system. Before the paging space can be used it must be activated using the `swapon` command. The VolumeGroup parameter specifies the volume group within which the logical volume for the paging space is to be made. The PhysicalVolume parameter specifies the physical volume of the VolumeGroup on which the logical volume is to be made.

In the second form of the `mkps` command, the ServerHostName parameter specifies the NFS server where the ServerFileName resides. The ServerFileName specifies the file that will be used for the NFS paging of the system. The ServerFileName file must exist and be exported correctly to the client that will use the file for paging.

monitor
This is similar to `chmon`, but publicly available for download on the Web at:
`http://aixpdslib.seas.ucla.edu`

netpmon
The `netpmon` command is used to determine network-related performance

problems. It monitors a trace of system events, reporting on network activity, performance, and network-related CPU usage during the monitored interval. By default, `netpmon` runs in the background while one or more application programs or system commands is being executed and monitored.

The `netpmon` command reports in detail on the following system activities:

- CPU usage
  The `netpmon` command monitors CPU usage by all threads and interrupt handlers. It estimates how much of this usage is due to network-related activities.
- Network device driver I/O
  The `netpmon` command monitors I/O operations through all Ethernet, Token Ring, and fiber distributed data interface (FDDI) network device drivers. In the case of transmission I/O, the command also monitors utilizations, queue lengths, and destination hosts. For received I/O, the command also monitors time in the demux layer.
- Internet Socket calls
- NFS I/O

With a moderate, network-oriented workload, `netpmon` increases the overall CPU utilization by 3-5 percent.

Note

`netpmon` does not support NFS v3.0, and is not fully supported by PCI adapters.

netstat

Traditionally, `netstat` is used for determining network problems rather than for measuring performance. But it is useful in determining the amount of traffic on the network to ascertain whether performance problems are due to congestion.

The `netstat` command displays information regarding traffic on the configured network interfaces. Reports include:

- The address of any protocol control blocks associated with the sockets and the state of all sockets.
- The number of packets received, transmitted, and dropped in the communications subsystem.
- Cumulative statistics for errors, collisions, packets transferred.

- Routes and the status.

Most of the variations of this command use less than 0.2 seconds of CPU time.

Netstat has been updated to support IPv6.

no

Use the `no` command to configure network attributes. The no command sets or displays current network attributes in the kernel. This command only operates on the currently running kernel. The command must be run again after each startup or after the network has been configured. Whether the command sets or displays an attribute is determined by the accompanying flag. The `-o` flag performs both actions. It can either display the value of an attribute or set a new value for an attribute. For a more information on how the network attributes interact with each other, refer to the *AIX Version 4.3 System Management Guide: Communications and Networks*.

Attention: Be careful when you use this command. The `no` command performs no range checking, and therefore it accepts all values for the variables. If used incorrectly, the `no` command can cause your system to become inoperable.

Some network attributes are runtime attributes that can be changed at any time. Others are loadtime attributes that must be set before the netinet kernel extension is loaded and be placed near the top of /etc/rc.net. If your system uses Berkeley-style network configuration, set the attributes near the top of /etc/rc.bsdnet.

New parameters have been added to this command in AIX version 4.3:

- *extendednetstat* enables more extensive statistics for network memory services.

  The command to change the setting is:

  ```
  no -o extendednetstat=<New Value>
  ```

  The extended statistics produce a degradation in system performance. You can enable extendednetstat by setting the value to 1. This must be done early in the boot process by putting the command in one of the /etc/rc files.

- *send_file_duration* specifies the cache validation duration for all the file objects that system call *send_file* accessed in the Network Buffer Cache.

The command to change the setting is:

```
no -o send_file_duration=<New Value>
```

A value of 0 means the cache will not be validated for every access. The default value is 300 (5 minutes).

- *nbc_limit* specifies the total maximum amount of memory that can be used for the Network Buffer Cache.

  The command to change the setting is:

  ```
  no -o nbc_limit=<New Value>
  ```

  The value is express in kilobytes. When the cache gets filled to the amount of memory specified by a *nbc_limit*, old cache data will be overwritten. The default is derived from thewall, and changing the value is effective immediately.

- *nbc_max_cache* specifies the maximum size of the cache object allowed in the Network Buffer Cache (NBC).

  The command to change the setting is:

  ```
  no -o nbc_max_cache=<New Value>
  ```

  Data objects bigger than specified by *nbc_max_cache* will not be put into *nbc_max_cache.* The default value is 131072 (128K).

- *nbc_min_cache* specifies the minimum size of the cache object allowed in the Network Buffer Cache (NBC).

  The command to change the setting is:

  ```
  no -o nbc_min_cache=<New Value>
  ```

  Data objects smaller than specified by *nbc_min_cache* will not be put into *nbc_max_cache.* The default value is 1 (1 byte).

- *ip6forwarding* specifies whether the kernel should forward IPv6 packets.

  The command to change the setting is:

  ```
  no -o ip6forwarding=<New Value>
  ```

  A change in the value is effective immediately. A value of 0 prevents forwarding. A value of 1 forwards IPv6 packets.

- delayack

  The command to change the setting is:

  ```
  no -a delayack=<new value>
  ```

  delayack delays acknowledgements for certain TCP packets and attempts to piggyback them with the next packet sent instead.

- delayackports

  The command to change the setting is:
  `no -a delayackports=<new value>`

  It defines the list of destination ports for which the operation defined by the delayack port option will be performed.

nfso

The `nfso` command configures Network File System (NFS) network variables

nfsstat

The `nfsstat` command displays statistical information about NFS clients or servers and RPC calls. There is a specific parameter for the server information (`nfsstat -s`) and for the client information (`nfsstat -c`).

- NFS Server Information

  The NFS server displays the number of NFS calls received (*calls*) and rejected (*badcalls*), as well as the counts and percentages for the various kinds of calls made.

- NFS Client Information

  The NFS client displays the number of calls sent and rejected, as well as the number of times a client handle was received (*nclget*), the number of times a call had to sleep while awaiting a handle (*nclsleep*), and a count of the various kinds of calls and their respective percentages.

- RPC Statistics

  The `nfsstat` command displays statistical information pertaining to the ability of a client or server to receive calls. Information includes:

  - Total number of RPC calls received or rejected.
  - Number of times no RPC packet was available when trying to receive.
  - Number of packets that were too short or had malformed headers.
  - Total number of RPC calls sent or rejected by a server.
  - Number of times a call had to be transmitted again.
  - Number of times a reply did not match the call.
  - Number of times a call timed out.
  - Number of times a call had to wait on a busy client handle.
  - Number of times authentication information had to be refreshed.

nice

The `nice` command is used to set the *nice* value of a process. If the process is running in the foreground, then the *nice* value is 20; if in the background, then the *nice* value is 24. `nice` can only set the value of the process at creation

time. If you want to change the *nice* value of a running process, then use the `renice` command.

The Command parameter is the name of any executable file on the system. If you do not specify an Increment value the nice command defaults to an increment of 10. You must have root user authority to run a command at a higher priority. The priority of a process is often called its nice value.

The nice value can range from 0 to 39, with 39 being the lowest priority. For example, if a command normally runs at a priority of 20, specifying an increment of 5 runs the command at a lower priority, 25, and the command runs slower. The nice command does not return an error message if you attempt to increase a command's priority without the appropriate authority. Instead, the command's priority is not changed, and the system starts the command as it normally would.

The nice value is used by the system to calculate the current priority of a running process. Use the ps command with the -l flag to view a command's nice value. The nice value appears under the NI heading in the ps command output.

PDT

The Performance Diagnostic Tool (PDT) collects configuration and performance information. It attempts to identify potential problems, both current and future. In assessing the configuration and the historical record of performance measurements, PDT attempts to identify unbalanced use of resources or asymmetrical aspects of configuration or device utilization. In general, if there are several resources of the same type, then performance is improved by meeting the following goals for a balanced use of those resources:

- Comparable numbers of physical volumes (disks) on each disk adapter.
- Paging space distributed across multiple physical volumes.
- Roughly equal measured load on different physical volumes.
- Trends in usage levels that will lead to saturation. Resources have limits to their use. Trends that would attempt to exceed those limits should be detected and reported. A disk drive cannot be utilized more than 100 percent of the time. File and file system sizes cannot exceed the allocated space.
- New consumers of resource-expensive processes that have not been observed before. Trends can indicate a change in the nature of the workload as well as increases in the amount of resource used:

- Number of users logged on.
- Total number of processes.
- CPU idle percentage.
- Inappropriate system parameter value settings that may cause problems.
- Hardware or software errors that may lead to performance problems. The PDT tool checks the hardware and software error logs and reports bad VMM pages.

perfpmr

The `perfpmr` package was developed to ensure that reports of suspected performance problems in AIX were accompanied by enough data to permit problem diagnosis by IBM. This makes the shell scripts that the command `perfpmr` invokes useful to other performance analysts as well.

The perfpmr package runs the following commands in the following sequence:

- `trace`
- `monitor`
- `iptrace`
- `filemon`
- `tprof`

It also collects the following:

- Paging space before and after perfpmr.sh is run
- Process listings
- nfsstat output
- netstat output
- error logs
- system configuration and parameters

The `perfpmr` command generates files containing statistics for each interval (*.int*) and summary (*.sum*) data from the interval files

The perfpmr facility is available from the Web at:
`ftp://ftp.software.ibm.com/aix/tools/perftools/perfpmr/`

Note

- When you submit a data using perfpmr, please download a new perfpmr from the web site just prior to running it, as it can be updated without warning.
- It is useful to load perfagent.tools (see Table 40 on page 335) so perfpmr.sh collects all the performance data required for analysis.

prof

This command displays a profile of CPU usage for each external symbol, (routine) in a specified program. It displays the percentage of execution time spent between the address of the symbol and the address of the next, the number of times that function was called and the average number of milliseconds per call. However, to get a profile at the source statement level, recompilation is required with the `-p` option.

ps

The `ps` command displays statistics and status information about processes in the system, including process or thread ID, I/O activity, and CPU and memory utilization.

The `ps` command can be used to monitor memory use by an individual process. The `ps v [pid]` command provides reports on memory-related statistics for individual processes such as page faults. It also describes the size of a working segment that has been touched, the size of a working segment and code segment in memory, the size of a text segment, the size of a resident set, and the percentage of real memory used by this process.

The `ps` command provides standard output on the current status of active processes. If the `-m` flag is used, it also gives 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.

The CPU time consumed by this command varies with the number of processes to be displayed, but it usually does not exceed 0.3 seconds.

Only a snapshot is provided by `ps`. To gather data over time, use the `tprof` command.

pstat
The `pstat` command is a non-interactive form of the crash command. `pstat` interprets the contents of the various system tables and writes it to standard output. You must have root user or system group authority to run the `pstat` command.

renice
The `renice` command alters the nice value of one or more running processes. The nice value is the decimal value of the system scheduling priority of a process. By default, the processes affected are specified by their process IDs. When you specify a process group, the request applies to all processes in the process group.

The nice value is determined in an implementation-dependent manner. If the requested increment raises or lowers the nice value of the executed utility beyond implementation-dependent limits, the limit whose value was exceeded is used.

If you do not have root user authority, you can only reset the priority of processes you own, and can only set their priorities from 0 to 20, with 20 being the lowest priority. If you have root user authority, you can alter the priority of any process and set the priority to any value from -20 to 20. The specified increment changes the priority of a process in the following ways:

- 1 to 20
  Runs the specified processes slower than the base priority.
- 0
  Sets priority of the specified processes to the base scheduling priority
- -20 to -1
  Runs the specified processes quicker than the base priority.

The `renice` command maps these values to those actually used by the kernel.

reorgvg
The `reorgvg` command reorganizes the placement of allocated physical partitions within the VolumeGroup, according to the allocation characteristics of each logical volume. Use the LogicalVolume parameter to reorganize specific logical volumes; highest priority is given to the first logical volume name in the LogicalVolume parameter list and lowest priority is given to the last logical volume in the parameter list. The volume group must be varied on and have free partitions before you can use the `reorgvg` command.

The relocatable flag of each logical volume must be set to y with the `chlv -r` command for the reorganization to take effect. Otherwise the logical volume is ignored.

Important considerations include:

- The `reorgvg` command does not reorganize the placement of allocated physical partitions for any striped logical volumes.
- At least one free physical partition must exist on the specified volume group for the `reorgvg` command to run successfully.
- To use this command, you must either have root user authority or be a member of the system group.
- If you enter the `reorgvg` command with the volume group name and no other arguments, it will only reorganize the first logical volume in the volume group. The first logical volume is the one listed by the `lsvg -l` VolumeName command.

rmss

The reduced memory system simulator, `rmss`, offers a way to simulate RS/6000 systems with different sizes of real memory that are smaller than those of your actual machine without having to extract and replace memory boards. Moreover, `rmss` provides a facility to run an application over a range of memory sizes, displaying, for each memory size, performance statistics such as the response time of the application and the amount of paging. In short, `rmss` helps uncover about how many megabytes of real memory are needed to run AIX and a given application, and how many users can run this application simultaneously in a machine with X megabytes of real memory. It is important to keep in mind that the memory size simulated by `rmss` is the total size of the machine's real memory, including the memory used by AIX and any other programs that may be running. It is not the amount of memory used specifically by the application itself. Because of the performance degradation it can cause, `rmss` can be used only by root user or a member of the system group. The smallest amount of memory that can be simulated is 8MB.

`rmss` has been updated in AIX version 4.3.2 to utilize new parameters in the `vmtune` command. Updates include:

- More than 80% of memory can now be pinned
- The `-M`, `-f/-F`, `-p/-P` options of vmtune are now read by rmss so they should be set before stealing memory (`rmss -c`) and reset after freeing memory (`rmss -r`).

sar
The `sar` command reports either system-wide (global among all processors) CPU statistics (which are calculated as averages for values expressed as percentages, and as sums otherwise) or it reports statistics for each individual processor. Therefore, this command is particularly important on SMP systems. With its numerous options, `sar` also provides queuing, paging and TTY statistics. Only root users are able to execute the `sar` command.

Note

The `sar` command only reports on local activities.

The `sar` command returns information on:

- The number of kernel processes terminating per second
- The number of times kernel processes could not be created because of enforcement of a process threshold limit
- The number of kernel processes assigned to tasks per second
- The number of IPC message primitives and semaphore primitives
- Queue, paging, CPU, and TTY statistics

schedtune
This command is used to set the parameters for the CPU scheduler and Virtual Memory Manager (VMM). It is implementation specific and resides in the /usr/samples/kernel directory.

Attention

Do not copy this command from another version or release of AIX, as doing so is likely to result in a system crash when the command is run.

SPLAT
SPLAT is one of the tools that superceeded `utld` with AIX V4.3.3. SPLAT (Simple Performance Lock Analysis Tool) is a tool for post-processing AIX trace files to perform analysis of lock activity in the AIX kernel and kernel extensions. At the time of development of this book, SPLAT was not publically available. However, there are plans to release the tool to IBM external users in the future.

You can check the following Web site for availability:

ftp://ftp.software.ibm.com:/aix/tools/perftools

stem
The stem (scanning tunneling encapsulating microscope) command is a tool for inserting instrumentation code (subroutines), either user-supplied or default routines provided with stem. The stem command operates on existing libraries and programs without requiring source code or recompilation of the libraries or programs as long as the program is not stripped.

stripnm
The stripnm command prints the symbol table of a specified object file to standard output.

svmon
The svmon command offers a more in-depth analysis of memory usage. The svmon command displays information about the current state of memory. The displayed information does not constitute a true snapshot of the memory, because the svmon command runs at the user level with interrupts enabled. The svmon command creates several types of reports; global user, process command, workload management class, segment, and detailed segment.

The svmon -G command uses about 3.2 seconds of CPU time. An svmon -P command for a single process takes about 0.7 seconds of CPU time.

A scan of 1GB of memory by svmon takes 4 seconds. On a 32 GB system it would take 2 minutes, so the command cannot be used on a large memory system.

svmon has been updated to provide additional information pertaining to 64-bit processes.

syscalls
The syscalls (system call tracing) command captures system call entry and exit events by individual processes or for all processes on the system. It can also maintain counts for all system calls over long periods of time.

tcpdump
The tcpdump command prints out the headers of packets captured on a network interface that matches the boolean Expression parameter. If no Expression parameter is given, all packets on the network will be dumped. Otherwise, only packets for which the Expression parameter is True will be dumped. Only Ethernet, Fiber Distributed Data Interface (FDDI), token-ring, and loopback interfaces are supported. Access is controlled by the permissions on /dev/bpfO,1,2, and 3.

time
The `time` command prints the elapsed time during the execution of a command, time in the system, and execution time of the time command in seconds to standard error.

> **Note**
>
> Sleep time is not charged to either system or user time.

timex
The `timex` command reports, in seconds, the elapsed time, user time, and system execution time for a command. With specified flags, the `timex` command lists or summarizes process accounting data for a command and all of its children. “Command” can be any executable file on the system. It also reports total system activity during the execution interval. Output is written to standard error. The system uses the /var/adm/pacct file to select process records associated with the command and includes background processes with the same user ID, workstation ID, and execution time window.

topas
This command is a tool that allows collection of vital statistics from a character terminal. `topas` compares to `chmon` and `monitor`, but has much more functionality than those tools.

tprof
The `tprof` tool is a very versatile AIX profiler that provides a detailed profile of CPU usage for every AIX process ID and name. It profiles at the application, routine, and source statement levels. This provides both a global view of the system as a whole and a detailed view of individual programs.

For subprogram level profiling, the `tprof` command can be run without modifying your executable program. You do not have to recompile with special compiler flags or linker options. This means that you can obtain a subprogram profile of any executable module that has already been built as long as it is not stripped. Please note that `tprof` does not work with COBOL or PASCAL programs at the statement level, but it will work with C and FORTRAN programs.

The `tprof` command uses `trace`. Since only one trace (with the same trace hook) can run on an AIX system, you will not be able to use `trace`, `filemon`, or `netpmon` simultaneously with `tprof`.

The tprof command will cause some system overhead, but because it only enables one trace hook, the degradation of the performance of a large compile, for example, would be less than 2 percent.

trace

The AIX trace facility is useful for observing system activity, particularly that of a running device driver. The trace facility captures a sequential flow of time-stamped system events, providing a fine level of detail on system activity. Events are shown in time sequence and in the context of other events. The trace facility is useful for expanding the trace event information to understand which, when, how, and even why the event happened. The operating system is shipped with permanent trace event points. These events provide general visibility to system execution. You can extend the visibility into applications by inserting additional events and providing formatting rules with low overhead. Because of this, the facility is useful as a performance-analysis tool and as a problem-determination tool. The trace facility is more flexible than traditional system-monitor services that access and present statistics maintained by the system. With traditional monitor services, data reduction (conversion of system events to statistics) is largely coupled to the system instrumentation. For example, the system can maintain the minimum, maximum, and average elapsed time observed for runs of a task and permit this information to be extracted.

The trace facility does not strongly couple data reduction to instrumentation, but it does provide a stream of system events. You need not presuppose what statistics are needed. The statistics or data reduction are to a large degree separated from the instrumentation. You can choose to develop the minimum, maximum, and average time for task A from the flow of events. But it is also possible to extract the average time for task A when called by process B, extract the average time for task A when conditions XYZ are met, or even decide that some other task, recognized by a stream of events, is more meaningful to summarize. This flexibility is important for diagnosing performance or functional problems. For example: netpmon uses the trace to report on network activity, including CPU consumption, data rates, and response time. The tprof command uses the trace to report the CPU consumption of kernel services, library subroutines, application program modules, and individual lines of source code in the application program.

trcrpt

The trcrpt command reads the trace log specified by the File parameter, formats the trace entries and writes a report to standard output. The default file from which the system generates a trace report is the /var/adm/ras/trcfile file, but you can specify an alternate File parameter.

utld

`utld` is a post-trace tool that was originally used in the development of AIX. It reports on locks and delays and thread/processor affinity. utld can be obtained from the web:
`ftp://ftp.software.ibm.com:/aix/tools/perftools`

With AIX V4.3.3, `utld` has been replaced by the tools *CURT* and *SPLAT*. SPLAT is still an IBM intenal tool, but CURT can be obtained from the same ftp site as `utld`.

vmstat

The first tool to use is `vmstat`, which provides very quick and compact information about various system resources and their related performance problems. The `vmstat` command reports statistics about the number of kernel threads in the run queue and the wait queue, memory, paging, disks, interrupts, system calls, context switches, and CPU activity. The reported CPU activity is a percentage breakdown of user mode, system mode, idle time and waits for disk I/O. Keep in mind that the CPU statistics on SMP systems are an average of all processors. With `vmstat`, you cannot see per-processor CPU usage on SMP systems.

The `vmstat` command adds little overhead to the system. It uses about 30 milliseconds of CPU time for each report generated.

When diagnosing a performance bottleneck, `vmstat` is only a first step. The system administrator will have to use more complex monitoring tools to check the indications of `vmstat` and to determine whether there are hidden performance bottlenecks.

`vmstat` has been modified to report the amount of virtual memory actually accessed because with Deferred Page Space Allocation (DPSA), the paging space may not get touched. The means that with DPSA on, you could see smaller values for avm, SIZE, and SZ.

There are special considerations about `vmstat` on AIX V4.3.2 and earlier versions, and AIX V4.3.3 and later versions. AIX 4.3.3 contains an enhancement to the method used to compute the percentage of CPU time spent waiting on disk I/O *(wio* time). The method used in AIX 4.3.2 and earlier versions of AIX can give an inflated view of *wio* time on SMPs in some circumstances. The *wio* time is reported by the commands `sar` (%wio), `vmstat` (wa), and `iostat` (%iowait).

Method used in AIX 4.3.2 and earlier AIX versions

At each clock interrupt on each processor (100 times a second in AIX), a determination is made as to which of four categories (usr/sys/wio/idle) to place the last 10 ms of time. If the CPU was busy in usr mode at the time of the clock interrupt, then *usr* gets the clock tick added into its category. If the CPU was busy in kernel mode at the time of the clock interrupt, then the *sys* category gets the tick. If the CPU was NOT busy, then a check is made to see if ANY I/O to disk is in progress. If any disk I/O is in progress, then the *wio* category is incremented. If NO disk I/O is in progress and the CPU is not busy, then the *idl* category gets the tick. The inflated view of *wio* time results from all idle CPUs being categorized as *wio* regardless of the number of threads waiting on I/O. For example, an AIX system with just one thread doing I/O could report over 90 percent *wio* time regardless of the number of CPUs it has.

Method used in AIX 4.3.3 and later

The change in AIX 4.3.3 is to only mark an idle CPU as *wio* if an outstanding I/O was started on that CPU. This method can report much lower *wio* times when just a few threads are doing I/O and the system is otherwise idle. For example, a system with four CPUs and one thread doing I/O will report a maximum of 25 percent *wio* time. A system with 12 CPUs and one thread doing I/O will report a maximum of 8.3 percent 'wio' time.

vmtune
This command allows you to modify the VMM parameters that control the behavior of the memory management subsystem.

`vmtune` has been updated with the following options:

- lrucket (`-l`)
  Specified the number of memory frames per bucket. Frames get assigned to buckets so when the VMM wants more memory, it scans the buckets rather than the frames. Each bucket is scanned twice in succession, once to set the reference bit and once to steal the page. The default is 131072, the equivalent of 512 MB of memory.
- defps (`-d`)
  Prior to AIX4.3.2, the VMM created paging space for a memory page that had been touched, but now with large memory systems, paging using DPSA may be useful. When DPSA is enabled, paging space will be created for the frame only when the frame is paged out.

A new option to vmstat (`-d`) was introduced to switch on (`-d1`) or switch off (`-d0`) DPSA.

- sync_release_ilock (`-s`)
  A non-zero value will cause sync() to flush all I/O to a file without holding the inode lock, and then use the lock to do the commit. The default value (`-s0`) will release the lock only after the commit.
- Options to display the current statistic counters (`-a`), to specify the number of memory pools (`-m`), specify the MP kernel (`-U`), and turn on and off page coloring (`-C`) have also been added.

Attention

Do not copy this command from another version or release of AIX, as doing so is likely to result in a system crash when the command is run.

wlmstat
The `wlmstat` command was introduced with AIX Version 4.3.3 Workload Manager (WLM). This command reports system resource usage statistics (CPU, Memory, and Disk I/O) by WLM superclasses, subclasses, and tiers.

Xprofiler
This is a GUI-based tools that helps you analyze your application's performance. Although it is more useful to analyze parallel applications, serial applications can be analyzed too.

Xprofiler only analyses CPU busy time, not when the CPU is idle or I/O is being performed.

xgprof
This is an extended graphical user interface profiler, and is an enhancement of the `gprof` command, which is the standard AIX graph profiler. `xgprof` uses the same data as `gprof`, and works the same way.

xmperf
This is a graphical user interface for analyzing system resources.

8.1.4 References

- *AIX Commands Reference Manuals*
- *AIX Performance Tuning Guide*, SC23-2365

8.2 Performance Toolbox (PTX) for AIX

Anyone faced with the task of keeping a network of computers well tuned and capable of performing as expected recognizes the need for a comprehensive tool for monitoring and tuning system performance.

The Performance Toolbox (PTX) for AIX is a set of tools to monitor the performance of UNIX platforms. PTX also aids in the tuning of UNIX platforms and offers a wide range of performance tools within a versatile framework for both stand-alone and networked systems.

PTX is designed to properly answer questions about performance such as “How can I tell my system is performing optimally?”, “Does the system meet my performance expectations?”, “Why is the system response time so slow lately?”, “Why do my applications take so much longer to run?” and “Which disk drive is a bottleneck in the I/O workload?”

8.2.1 Performance Toolbox concepts

PTX for AIX is a client/server tool set based on Motif. It monitors local and remote system performance with several graphical windows that are fully user configurable. This includes 2D and 3D views of performance statistics.

PTX for AIX is divided in two components; agent and manager.

The manager has the tools for the key elements of performance monitoring and tuning:

- Monitoring of system statistics
- Analysis of system statistics
- Tuning of the system performance parameters to balance the utilization of fixed system resources

The agent’s part consists of programs that run on the machines you want to monitor. The role of the programs is to get and filter data that will be handled by the manager.

The client/server implementation allows PTX to help monitor and tune the performance of various UNIX systems in a network environment from a single graphics workstation.

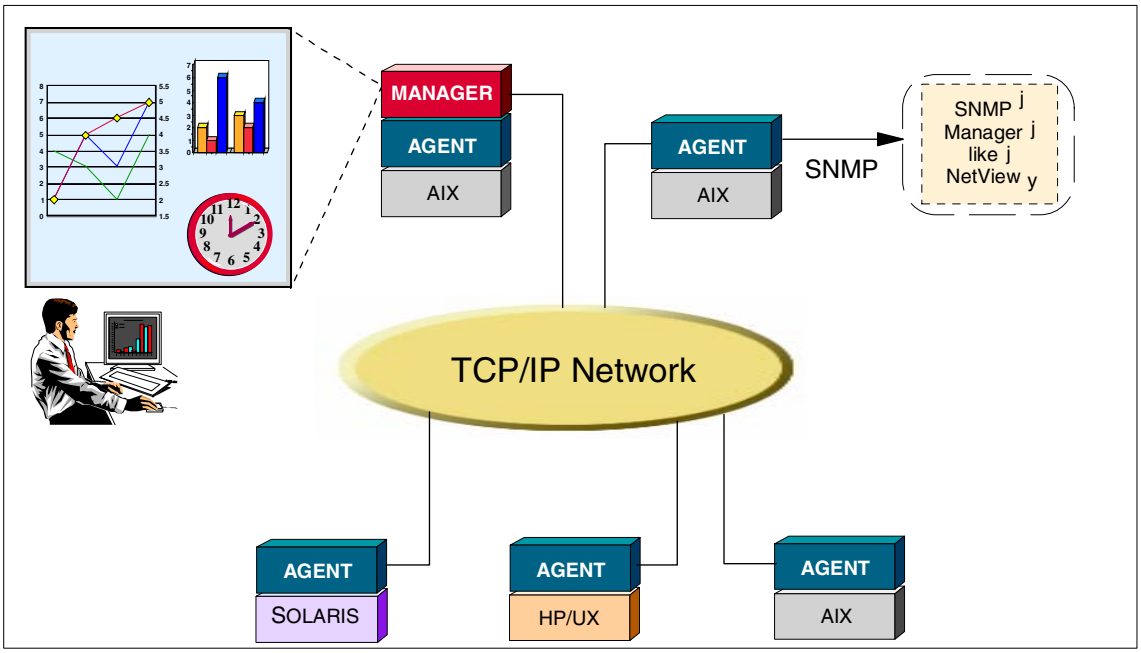


Figure 111. Performance Toolbox Environment

While you may use a graphics display connected to the server, we strongly recommend that you use a separate graphics workstation. This will minimize the impact of running PTX on the system load, and it will give a truer indication of system performance.

Major PTX features are:

- Concurrent monitoring of near-real-time local and remote machine statistics
- Configurable color graphic display format in user-designed monitoring consoles
- Extensive record/playback/analysis for long-term performance analysis
- Data filter/alarm facilities and exception monitoring
- Separately installable manager and agent components that allow a manager to monitor multiple agents, and an agent to supply data to multiple managers
- Agent support on HP-UX Version 9 and Sun Solaris Versions 2.4 and 2.5

- Application programming interfaces (APIs) that allow programmers to access local or remote data as well as to register custom data with the local agent
- Ability to respond to SNMP “get” and “get next” requests, and to send traps to an SNMP manager (AIX agents only)
- Support for RS/6000 SP systems using the Performance Toolbox Parallel Extensions Feature of the Parallel System Support Program (PSSP) for AIX

Performance Toolbox Parallel Extensions (PTPE) is a feature of PSSP 2.2 and is used in combination with PTX to simplify performance analysis and reduce PTX administrative overhead in RS/6000 SP systems by organizing the SP nodes into reporting groups. It provides the utilities to monitor, store, and retrieve performance information collected on RS/6000 SP subsystems such as the high performance switch, virtual shared disk, and LoadLeveler.

8.2.2 Graphical monitoring and analysis issues

Although the information produced by the graphics can be obtained by running individual text based programs, it is useful to depict the results in a graphical manner. This makes analysis easier, and because the graphical representations are dynamic, you can see the state of your system and other remote system that you are monitoring easily.

8.2.2.1 System monitoring

In network client/server applications, the performance of sets of systems working together can be as important as the performance of an individual system. Likewise, the performance of multiple applications working together can be as important as that of an individual application. Therefore, it is very important to be able to get the big picture by graphically viewing many correlated parameters concurrently across multiple nodes in a network. PTX allows a user to concurrently visualize the live (near-real-time) performance characteristics of the clients and server applications across the network.

8.2.2.2 Analysis and control

By providing an umbrella for tools that can be used to analyze performance data and control system resources, the manager program `xmperf` assists the system administrator in keeping track of available tools and in applying them in appropriate ways. This is done through a customizable menu interface. Tools can be added to menus, either with fixed sets of command-line arguments to match specific situations or in a dialog window. The menus of `xmperf` are pre configured to include most of the performance tools shipped as part of the tools option of the agent component.

Features for analyzing a recording of performance data are provided by the azizo program and its support programs. Recordings can be produced from the monitoring programs xmperf and 3dmon during monitoring, or they can be created by the xmservd daemon. The xmservd daemon allows for recording with a minimum of overhead. This makes constant recording possible so that you can analyze performance problems after they have occurred.

Finally, using the agent component filter filtd, you can define conditions that, when met, could trigger any action you deem appropriate, including alerting yourself and/or initiating corrective action without human intervention. This facility is entirely configurable so that alarms and actions can be customized to your installation.

8.2.2.3 Capacity planning

If you can make your system simulate a future load scenario, xmperf can be used to visualize the resulting performance of your system. By simulating the load scenario on systems with more resources, such as more memory or more disks, the result of increasing the resources can be demonstrated.

8.2.2.4 Network operation

The xmservd data-supplier daemon can provide consumers of performance statistics with a stream of data. Frequency and contents of each packet of performance data are determined by the consumer program. Any consumer program can access performance data from the local host and one or more remote hosts. Any data-supplier daemon can supply data to multiple hosts.

8.2.2.5 SNMP interface

By entering a single keyword in a configuration file, the data-supplier daemon can be told to export all its statistics to a local snmpd SNMP agent. Users of an SNMP manager such as IBM NetView see the exported statistical data as an extension of the set of data already available from snmpd.

The SNMP multiplex interface is only available on IBM RS/6000 agents.

8.2.3 Manager

The manager component of the Performance Toolbox for AIX has the following components:

- xmperf

  The main interface program providing graphical display of local and remote performance information in a menu interface to commands of your choice.

- `3dmon`

  A program that can monitor up to 576 statistics simultaneously and display the statistics in a three-dimensional graph.

- `3dplay`

  A program to play back `3dmon` recordings in a `3dmon`-like view.

- `chmon`

  Supplied as an executable as well as in source form. This program allows monitoring of vital statistics from a character terminal.

- `exmon`

  A program that allows monitoring of alarms generated by the `filtd` daemon running on local or remote hosts.

- `azizo`

  A program that allows you to analyze any recording of performance data. It lets you zoom in on sections of the recording and provides graphical as well as tabular views of the entire recording or zoomed-in parts.

- `ptxtab`

  A program that can format recording files for printed output.

- `ptxmerge`

  This program allows you to merge up to 10 recording files into one. For example, you could merge `xmservd` recordings from the client and server sides of an application into one file to better correlate the performance impact of the application on the two sides.

8.2.3.1 The xmperf program

The `xmperf` program is the most comprehensive and largest program in the manager component. It is based on the X Window system and was developed with the OSF/Motif toolkit. The `xmperf` program allows you to define monitoring environments to supervise the performance of the local and remote AIX systems. The initial screen is shown in Figure 112 on page 368.

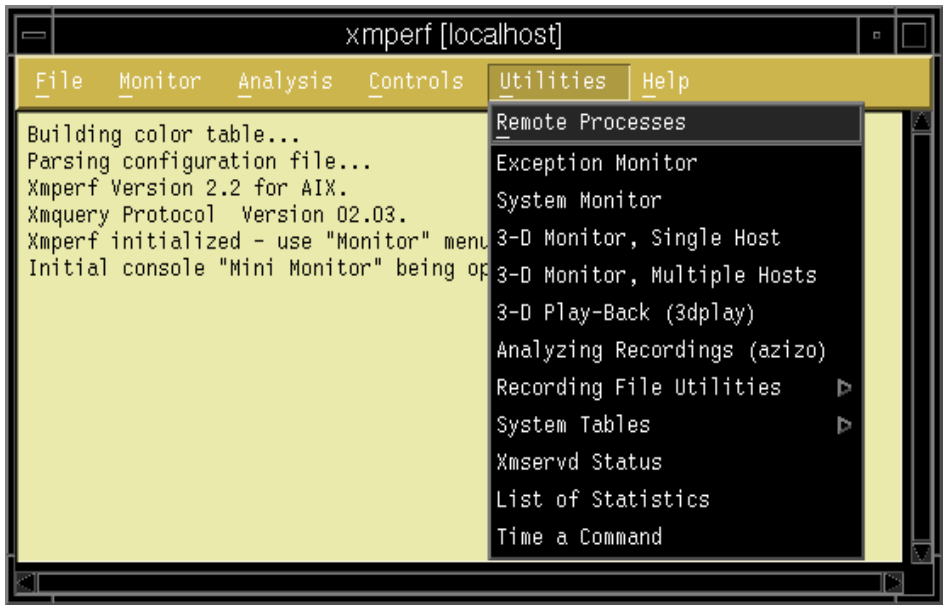


Figure 112. AIX Performance Toolbox Initial Screen

Each monitoring environment consists of a number of consoles displayed as graphical windows. Consoles, in turn, contain one or more instruments, and each instrument can show one or more monitored values (see Figure 113 on page 369).

The following terms are used to refer to the `xmperf` monitoring functions or components:

- A *console* is a graphical window containing instruments that monitor the system. A console can have one or more instruments.
- An *instrument* is a graphical view of monitored values, and each instrument can show one or more monitored values. 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 quantifiable aspect of system performance, including CPU usage for user processes, disk transfer rates, or TCP bytes transmitted.

The `xmperf` command is not hard-coded to monitor a fixed set of resources. It is dynamic in the sense that a system administrator can customize it to focus on the critical resources for each host.

Consoles can record the data they monitor to disk. Such recordings can be played back with `xmperf` and analyzed with the `azizo` program.

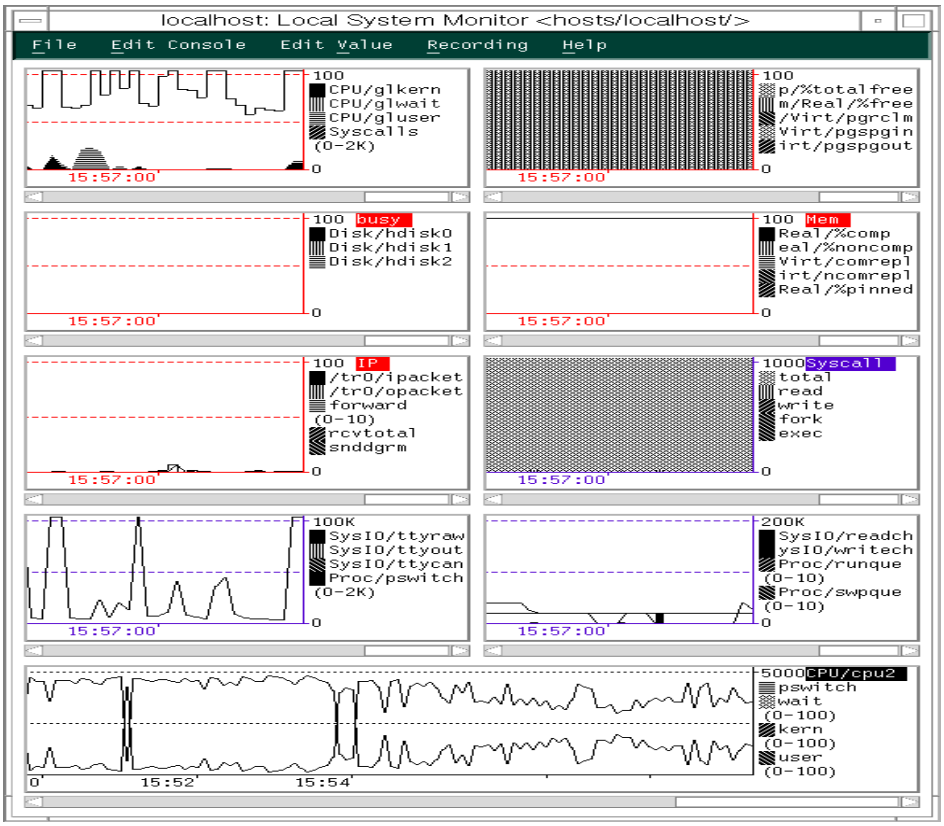


Figure 113. A Users' Console

8.2.3.2 The 3dmon and 3dplay utilities

The `3dmon` program offers a graphical presentation that allows you to survey the same key data on several hosts. The output looks like a chessboard, except that each side may have from one to 24 fields.

If you use `3dmon` so that it will display data from several hosts, it will emit a request at launch time for any available agents on the network, then lets you choose those you want to survey. The `3dmon` program is really helpful for monitoring all the active CPUs in the network.

It can be configured to contact specific hosts - up to 24 at one time - and to display 24 values for a maximum of 576 simultaneous statistics.

With 3dmon, it is possible to record any 3dmon console for future analysis using the 3dplay program; 3dmon allows the user to add notes to a record. With 3dplay you can use seek and speed controls like those found on a conventional VCR.

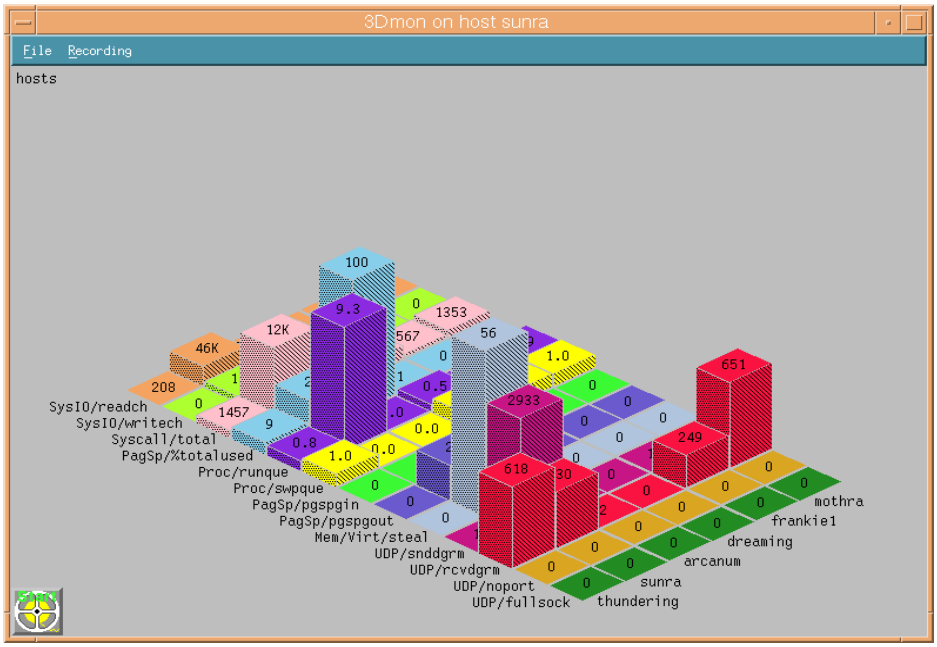


Figure 114. A View of 3dmon

8.2.3.3 The azizo and ptxmerge utilities

The azizo tool is a PTX program that is used to analyze prerecorded files. It can analyze only one file at a time and render a maximum of 256 statistics (see Figure 115 on page 371). If multiple recordings must be analyzed together, the support program ptxmerge can be used to merge multiple recording files into one for simultaneous analysis of statistics from multiple sources.

The azizo tool provides statistics on the file, number of samples, time stamp, and minimum, maximum, average, and standard deviation for each value recorded. With azizo, it is possible to build sub graphs from the original graph by removing values or changing start and/or end times. These subgroups can be saved for further analysis.

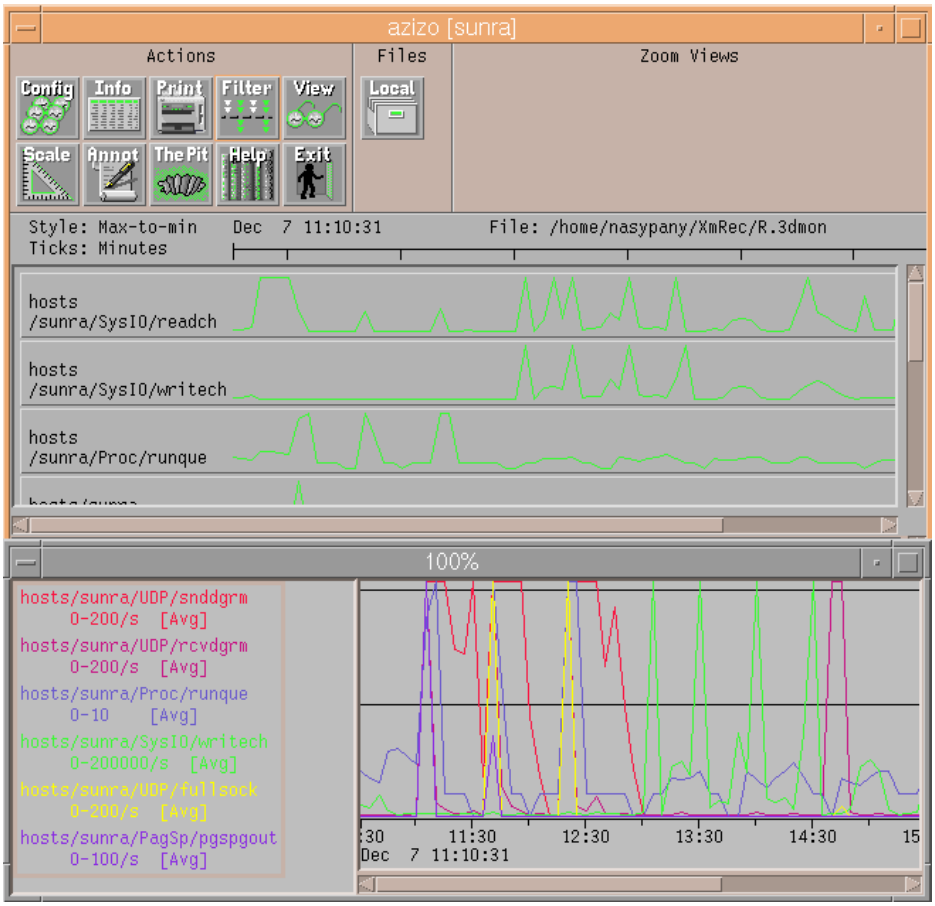


Figure 115. Azizo interface

8.2.3.4 The exmon utility

The exception monitor program, exmon, is a tool to manage exceptions. Exceptions are numbered from 0 to 10 - they are packets transmitted from the agents on the networks for consumers who request them. The exmon program is one of these consumers.

The exmon program is designed to provide a convenient facility for monitoring exceptions as they are detected on remote hosts. It does so by allowing its user to register subscriptions for exceptions packets from all or selected hosts in the network and to monitor the exception status in a graphical window.

8.2.3.5 The chmon program

The chmon program allows you to display data on a text-based screen for a given host. Default data are CPU, memory, disks, and system calls activities. It also provides the data consumed by the most active processes on the machine.

The chmon program is a sample of using one of the two APIs provided with PTX, the *Rsi* API. It can be modified to display requested values in any useful format. A sample of a chmon screen is shown in the Figure 116.

```
Data Consumer API        Remote Monitor for host       Mon Apr 10 13:54:46 2000
CHMON Sample Program        *** localhost***           Interval:     5  seconds

% CPU                                                EVENTS/QUEUES    FILE/TTY
Kernel     0.8   |                                |  Pswitch   1069   Readch     805
1ser       0.2   |                                |  Syscall   2850   Writech   2 16
Wait       0.0   |                                |  Reads        7   Rawin        0
Idle      98.9   |############################    |  Writes       0   Ttyout    2 16
                                                     Forks        0   Igets        0
PAGING counts    PAGING SPACE    REAL MEM 511MB      Execs        0   Namei        0
Faults       0   % Used   0.0    % Comp     0.0     Runqueue   0.0   Dirblk       0
Steals       0   % Free 116.2    % Noncomp200.0     Swapqueue  1.0
Reclaim      0   Size,MB    88   % Client   0.0

PAGING page/s    DISK      Read    Write      %     NETWORK    Read    Write
Pgspin       0   ACTIVITY KB/sec  KB/sec  Busy     ACTIVITY  KB/sec  KB/sec
Pgspout      0   hdisk2       0.0     0.0   0.0     lo0         0.3     0.3
Pagein       0   hdisk3       0.0     0.0   0.0     tr0         0.0     0.0
Pageout      0   hdisk0       0.0     0.0   0.0
Sios         0   hdisk1       0.0     0.0   0.0
                 cd0          0.0     0.0   0.0
```

Figure 116. Chmon Illustration

8.2.4 Agent

The agent component is a collection of programs that make it possible for a host to act as a provider of performance statistics across a network or locally. The key program is the daemon xmservd. It supplies the statistics for the monitoring environment through an API called System Performance Measurement Interface (SPMI). The SPMI implementation allows one agent to supply data to many managers, and it allows one manager to request data from many agents. The SPMI interface can be used for any dynamic data-supplier program to export its data.

The xmservd daemon can also record pre-configured sets of statistics on a local system for 24x7 monitoring. These recordings can then be post-processed by the command utilities and azizo. Additionally, the agent can record statistics by activity, rather than by metric name. This is controlled by a user-defined set of criteria applied to each statistic. All performance-related tools already available in AIX can be accessed through this interface. In addition, the ability to record load scenarios and play them

back in graphical windows at any desired speed offers new ways of analyzing performance problems.

Another agent program is `filtd`. This daemon is in charge of filtering data by processing all previously defined expressions that define new statistics. This feature allows you to easily combine existing “raw” statistics into new statistics that make more sense in the monitoring environment. The `filtd` also supplies `exmon` with the exceptions that occurred in the monitored system, and it allows you to define alarms that trigger actions.

The xmservd daemon also acts as a supplier of performance statistics to Simple Network Management Protocol (SNMP) managers. This feature is only available in AIX systems.

8.2.5 Monitoring an SMP with the performance toolbox

PTX can be used to monitor an SMP system.

In order to monitor an SMP with Performance Toolbox, you need a graphical display. While you may use a graphical display connected to some SMPs, we again strongly recommend that you use a separate graphics workstation. This will minimize the impact of running PTX on the system load, and it will give a truer indication of system performance.

PTX provides predefined consoles for monitoring a single system. All you need is to create your own console using the tools provided in the main window of `xmperf`.

You can select a value and 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.

Select values you want to monitor for processor 0. You can then edit your console and add a new local instrument for processor 1, and so on. Figure 117 on page 374 shows an example of a customized SMP console.

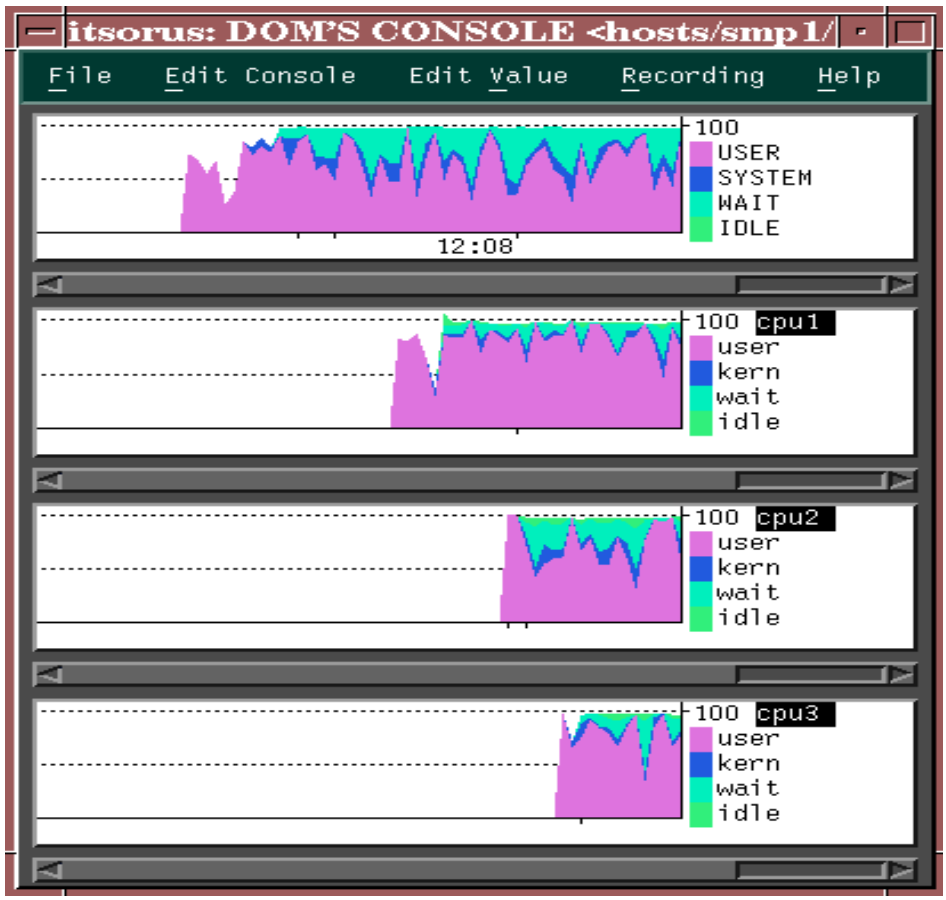


Figure 117. SMP Console Example

8.2.5.1 Monitoring an SMP with 3dmon

The `3dmon` program 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 3-D Monitor, Single Host and 3-D Monitor, Multiple Hosts submenus. Choosing Local Processors (CPUs) will give you a screen where you can choose which CPUs you want to monitor, and when complete, your screen is displayed. The performance values you will be monitoring are user, kern, wait, pswitch, syscall, read, write, fork, exec, readch, writech, iget, namei, and dirblk.

The 3dmon monitor may also be invoked from the command line by typing:

```
# 3dmon -h <hostname>
```

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:

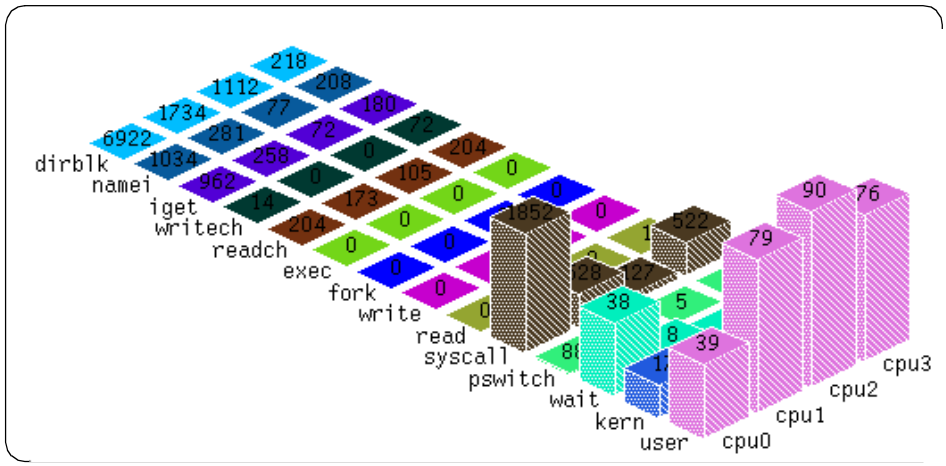


Figure 118. 3dmon Output on a 4-Way SMP

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. Special notices

This publication is intended to help system operators, system administrators, and sales representatives understand the factors that determine the performance of applications running under AIX V4 on the IBM RS/6000 and IBM @server pSeries platforms. The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM RS/6000, IBM @server pSeries, and AIX. See the PUBLICATIONS section of the IBM Programming Announcement for IBM RS/6000, IBM @server pSeries, and AIX for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer

responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

This document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM ®
e (logo)® 
Lotus Notes
Redbooks Logo
SP2
Ultrastar
Wave
XT
Domino
Lotus
Redbooks
SP
System/390
Visualization Data Explorer
Websphere

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli

Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

SPEC and the “performance chart” SPEC logo are registered trademarks of the Standard Performance Evaluation Corporation.

Other company, product, and service names may be trademarks or service marks of others.

Appendix B. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

B.1 IBM Redbooks publications

See individual chapters for related redbooks.

B.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

B.3 Other resources

See individual chapters for related publications.

B.4 Referenced Web sites

See individual chapters for related Web sites.

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Abbreviations and acronyms

ABI	Application Binary Interface
ACID	Atomicity Consistency Isolation Durability
ACL	Access Control List
AFS	Andrew File System
AGP	Accelerated Graphics Port
AIX	Advanced Interactive Executive
ANSI	American National Standards Institute
API	Application Program Interface
ARB	Architecture Review Board
ARP	Address Resolution Protocol
ASCII	American National Standard Code for Information Interchange
AST	Automatic Summary Table
AT&T	American Telephone & Telegraph
ATM	Asynchronous Transfer Mode
B2C	Business to Consumer
bf	Bigfoot
BI	Business Intelligence
BIU	Bus Interface Unit
BPU	Branch Processor Unit
bps	Bits per second
BTAC	Branch Target Address Cache
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CATIA	Computer-graphics Aided Three-dimensional Interactive Application
CCR	Condition-Code Register
CDE	Common Desktop Environment
CD-ROM	Compact Disk Read Only Memory
CGI	Common Gateway Interface
CICS	Customer Information Control System
CIFS	Common Internet File System
CISC	Complex Instruction Set Computer
CMOS	Complimentary Metal Oxide Semiconductor
COFF	Common Object File Format
CPI	Cycles Per Instruction
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CTS	Clear tO Send
CWS	Control Workstation
DB	Database
DBMS	Database Management System

DCD	Data Carrier Detect
DCE	Data Communication Equipment
DCU	Data Cache Unit
DMA	Direct Memory Access
DSR	Data Set Ready
DSS	Decision Support System
DTE	Data Terminal Equipment
DTR	Data Terminal Ready
EAS	Effective Application Speed-up
ECC	Error Correction Code
EISA	Extended Industry Standard Architecture
ESCON	Enterprise Systems Connection (architecture)
ESP	Engineering Support Processor
ESS	Enterprise Storage Server
FC-AL	Fibre Channel Arbitrated Loop
FCRSS	Fibre Channel RAID Storage Server
FDDI	Fiber Distributed Data Interface
fdpr	Feedback directed program restructuring
FFM	Fraction in Faster Mode
FPR	Floating-Point Register
FPU	Floating-Point Unit
FTP	File Transfer Protocol
FXU	Fixed-Point Unit
F/W	Fast/Wide
Gb	Gigabit
GB	Gigabyte
GFLOPS	Giga Floating-point Operations Per Second
GIS	Geographic Information System
GPC	Graphics Performance Characterization
GPR	General Purpose Register
GUI	Graphical User Interface
HACMP	High Availability Cluster Multi-Processing
HACWS	High Availability Control Workstation
HIPPI	High Performance Parallel Interface
HPC	High Performance Computing
HP/UX	Hewlett-Packard Co./UNIX
HSD	Hashed Shared Disk
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
ICU	Instruction Cache Unit
IEEE	Institute of Electrical and Electronics Engineers
IMAP	Internet Message Access Protocol
I/O	Input/Output
IP	Internet Protocol
IPC	Instructions Per Cycle
IPC	Interprocess Communication

IPCMOS Interlocked Pipelined CMOS

ISA Industry Standard Architecture

ISDN Integrated-Services Digital Network

ISO International Organization for Standardization

ISP Internet Service Provider

IT Information Technology

ITSO International Technical Support Organization

IU Integer Unit

JBOD Just a Bunch Of Disks

JFS Journaled File System

JPEG Joint Photographic Experts Group

JVM Java Virtual Machine

Kb Kilobit

KB Kilobyte

km Kilometer

LAN Local Area Network

LANE LAN Emulation

LDAP Lightweight Directory Access Protocol

LPAR Logical Partitioning

LPP Licensed Program Product

LRU Least Recently Used

LU Logical Unit

LV Logical Volume

LVD Low Voltage Differential

LVM Logical Volume Manager

MAU Multistation Access Unit

Mb Megabit

MB Megabyte

MCM Multi Chip Module

MESI Modified, Exclusive, Shared, and Invalid

MFLOPS Millions of Floating-point Operations Per Second

MHz Megahertz

MIPS Millions of Instructions Per Second

MMU Memory Management Unit

MP Multiprocessor

MPEG Moving Pictures Experts Group

MPI Message Passing Interface

MPL Message Passing Library

MPP Massively Parallel Processor

ms Milliseconds

MTU Maximum Transmission Unit

MUSPPA Multiple User Space Processors Per Adapter

MWC Mirrored Write Consistency

N/A Not Applicable

N/A Not Available

NBC Network Buffer Cache

NC Network Computer

NetBIOS Network Basic Input Output System

NFS Network File System

NFSO Network File System Options

NIC Network Interface Card

ns Nanoseconds

NUMA Non-Uniform Memory Access

OLAP Online Analytical Processing

OLTP On-line Transaction Processing

OPC OpenGL Performance Characterization

OpenGL Open 3D Graphics Library interface

OPS Oracle Parallel Server

OS Operating System

OSF Open Software Foundation Inc.

P2SC POWER2 Super Chip

PB Petabyte

PC Personal Computer

PCI Peripheral Component Interconnect

PDT Performance Diagnostic Tool

PHIGS Programmers Hierarchical Interactive Graphics Standard

PLB Picture Level Benchmark program

PLL Phase Locked Loop

POP3 Post Office Protocol 3

POWER Performance Optimization With Enhanced RISC

PP Physical Partition

PSSP Parallel System Support Programs

PTF Program Temporary Fix

PTPE Performance Toolbox Parallel Extensions

PTX Performance Toolbox

PV Physical Volume

PVM Parallel Virtual Machine

PVMe Parallel Virtual Machine extended

QAS Quick Arbitration Select

RAID RedundanT Array Of Independent Disks

RAM Random Access Memory

RAN Remote Asynchronous Node

RARP Reverse Address Resolution Protocol

RDBMS Relational Database Management System

RISC Reduced Instruction Set Computer

rmss Reduced Memory System Simulator

RPC Remote Procedure Call

rpm Revolutions per minute

RS RISC System

Rsi Remote Statistics Interface

RTS Ready To Send

RVSD Recoverable Virtual Shared Disk

RxD Receive Data

SAK System Assurance Kernel

SAP Systems, Applications, Products in data processing

SCSI	Small Computer System Interface
SCSI SE	SCSI Single Ended
SCSI-2 F/W	SCSI-2 Fast/Wide
SDM	System Development Multitasking
SFM	Speed-up of the Faster Mode
SFS	System File Server
SIG	Special Interest Group
SMB	Server Message Block
SMIT	System Management Interface Tool
SMP	Symmetrical Multiprocessor
SMTP	Simple Mail Transfer Protocol
SNA	System Network Architecture
SNMP	Simple Network Management Protocol
SOI	Silicon On Insulator
SP	Single Precision
SPARC	Scalable Processor ARChitecture
SPEC	Systems Performance Evaluation Corporation
SPMI	System Performance Measurement Interface
SPOF	Single Point Of Failure
SPOT	Shared Product Object Tree
SQL	Structured Query Language
SRAM	Static RAM
SSA	Serial Storage Architecture
SSL	Secure Socket Layer
stem	Scanning Tunneling Encapsulating Microscope
STP	Shielded Twisted Pair
T1	Transmission rate: 1.544 Mb/s
T3	Transmission rate: 44.736 Mb/s
TB	Terabyte
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TLB	Translation Lookaside Buffer
TP	Transaction Processing
TPC	Transaction Processing Council
TPM	Transactions Per Minute
TPP	Toward peak performance
TPS	Transactions Per Second
TTY	Teletypewriter
TxD	Transmission Data
UDB	Universal Database
UDP	User Datagram Protocol
UNI	User Network Interface
UP	Uniprocessor
URL	Universal Resource Locator
UTP	Unshielded Twisted Pair
VCI	Virtual Circuit Identifier
VCR	Video Cassette Recorder

VMM	Virtual Memory Manager
VPI	Virtual Path Identifier
VSD	Virtual Shared Disk
WAN	Wide Area Network
WAS	WebSphere Application Server
WLM	AIX Workload Manager
X	X Window System
XCOFF	Extended Common Object File Format
XDM	X Display Manager
XDR	External Data Representation
XOFF	Transmitter off
XON	Transmitter on
XPC	X Performance Characterization
XPS	Extended Parallel Server

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 845 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-4810-01 Understanding IBM @server pSeries Performance and Sizing
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

Understanding IBM @server pSeries Performance and Sizing

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Understanding IBM @server pSeries Performance and Sizing

Comprehend IBM RS/6000 and IBM @server pSeries hardware architectures

Get an overview of current industry benchmarks

Understand how to size your system

This redbook is an update to the successful first edition of Understanding IBM RS/6000 Performance and Sizing, SG24-4810 that was published in 1997. It gives a broad overview of IBM RS/6000 and IBM @server pSeries performance and sizing.

Contained in this redbook is a close-up, performance related view of the different hardware architectures IBM offers in its RS/6000 and @server pSeries systems, including system, processor, memory, storage, and network architectures. One chapter is dedicated to general sizing rules for a number of environments such as database sizing, IBM HTTP server sizing, Net.Commerce sizing, and Lotus Domino sizing. The reader will also find a description of the industry benchmarks that are performed on IBM systems as well as an overview of AIX performance tools.

This redbook is intended to be a handbook for anyone who wants to gain a deeper understanding of IBM RS/6000 and IBM @server pSeries architectures from a performance perspective.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-4810-01 ISBN 073841915X