Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
FAX 415 969-9131

For U.S. Sales Office locations, call:
800 821-4643
In California:
800 821-4642

Australia: (02) 413 2666
Belgium: 32-2-759 5925
Canada: 416 477-6745
Finland: 358-0-502 27 00
France: (1) 30 67 50 00
Germany: (0) 89-46 00 8-0
Hong Kong: 852 802 4188
Italy: 039 60551
Japan: (03) 221-7021
Korea: 822-563-8700
Latin America: 415 688-9464
The Netherlands: 033 501234
New Zealand: (04) 499 2344
Nordic Countries: +46 (0) 8 623 90 00
PRC: 861-831-5568
Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (1) 825 71 11
Taiwan: 2-514-0567
UK: 0276 20444

Elsewhere in the world,
call Corporate Headquarters:
415 960-1300
Intercontinental Sales: 415 688-9000

Conclusion

Sun provides a comprehensive system solution for computational needs. From the hardware design emphasizing modularity and performance to the multi-threaded Solaris 2 kernel, to the multi-processor aware tools and compilers, the SPARCserver 1000 provides solutions to computationally intensive problems previously provided by only specialized and costly systems. No matter whether the solution calls for an array processor (SIMD), a fully symmetric multi-processor (SMP) or loosely coupled, cooperative systems (clusters), the SPARCserver 1000 is the right choice. Sun has made it possible to have superior computational performance at low cost and has provided a scalable, upgradeable architecture protecting one's investment.

The added value of the Solaris operating system and its tools does not take away from any of the other SVR4 benefits, such as standards (API, ABI), portability across hardware, interoperability across different vendor platforms, and scalability. All Catalyst applications run unchanged on the SPARCcenter and SPARCserver systems with its multi-threaded Solaris OS offering an easy upgrade path as these applications become parallelized themselves. Sun Microsystems is delivering the power for applications of today and tomorrow.

result of the shared memory capabilities of threads and the unpredictable execution sequence of threads. As a result, tools are needed to address these new concepts in software development.

SunPro's approach is to first extend existing SPARCworks tools to understand threads. SPARCworks Debugger MT is discussed below. Additionally, SunPro will provide tools to help developers in parallelizing code and detecting deadlocks and data races. Without tools these problems can be very difficult to find and reproduce.

SPARCworks Debugger MT

SPARCworks Debugger MT, an add-on to SunPro's current SPARCworks Debugger, allows developers to view, set breakpoints and navigate parallel execution streams -- threads. The debugger supports multi-threading of applications written in C, C++, or FORTRAN.

MT compatible language libraries will also be provided for C, C++ and FORTRAN. These runtime libraries will execute correctly in an MT environment.

SunPro provides developers with a range of options for getting the maximum performance out of an application. Developers can use the compiler for automatic parallelization, use compilation analysis to add directives, or include direct calls to the user level threads library. They can also combine the options all within one application. This gives software developers a very flexible range of choices when optimizing applications, a balance can be made between the amount of effort invested versus the amount of performance gained.

supercomputers, is an integrated compilation system that provides direct translation from source code to optimized code that balances the effectiveness of parallelization and other optimizations.

As FORTRAN MP performs the automatic loop parallelization in the optimizer portion of the compiler, it can automatically detect both loop level parallelism and instruction level parallelism per processor. FORTRAN MP performs automatic restructuring of code; these are automatic transformations that expose higher degrees of loop level parallelism.

The parallelization analysis report can be used to identify loops that were not automatically parallelized by FORTRAN MP. Directives, via comment statements, can then be placed in the code[1] to tell the compiler that it is all legal to parallelize a loop. This provides a high degree of loop parallelization allowable within an application.

FORTRAN MP utilizes a micro-tasking library which runs on top of the user level threads library. The micro-tasking library manages the execution of multiple loop iterations on an MP system. It takes a parallelized loop and spreads it across multiple threads. Each thread may in turn execute on a separate CPU. The number of threads allocated for execution of a parallelized program is specified by the user by setting an environment variable. The compiler does not assume any particular number of CPUs and generates parallel code that will be able to run on any number of threads (even one). The developer does not need to recompile if the number of CPUs changes.

SPARCompiler FORTRAN MP makes parallelizing an application simple by taking the source code “as is” and automatically creating a multi-threaded (MT) application. Directives to the compiler allow further tuning of the application for multi-processor systems. This enables software developers to get maximum application performance from multi-threading and multi-processing technology.

MT Tools

To gain even more performance from an application a developer may program directly to the threads library. This is where the function level parallelism of an application must be stated explicitly by the user. Multi-threaded development introduces new challenges, such as non-determinism, deadlocks and data races. These challenges are a

1. Explicit parallelization

This type of parallelism cannot be found automatically by the compiler, as can loop level parallelism. To exploit function level parallelism it is necessary to program directly to the Solaris 2 user level threads library.

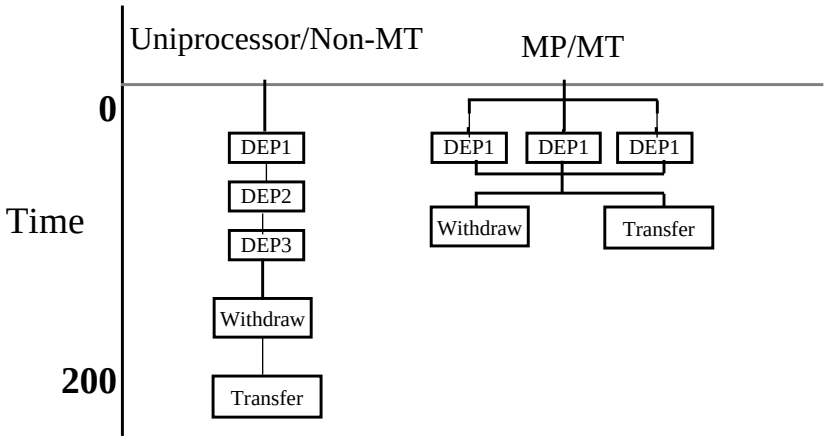

Figure 6 Function Level Parallelism

SunPro's Compiler and Tool Support

Offering two levels of parallelism to applications programmers also requires providing tools for each level. SunPro, the software development business unit of Sun Microsystems, is providing compilers and tools to address both levels.

SPARCompiler FORTRAN MP

SunPro's SPARCompiler FORTRAN MP provides automatic parallelization by breaking up loops within an application into independent tasks that can be run in parallel on multiple processors. In addition, "directives" are provided, which are commands that direct the compiler to perform specific parallelization, and parallelization analysis reports that identify areas of the application where further parallelization can be performed. A key feature, previously found only on

This type of parallelism can be found automatically by the compiler as loops that can be split up across processors. This type of parallelism is called implicit parallelism. Explicit directives (explicit parallelism) are also provided to further optimize an application.

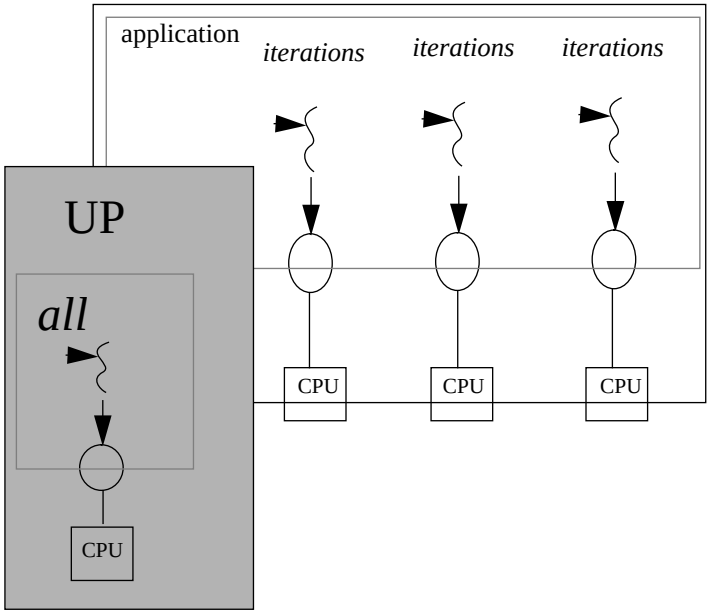

Figure 5 Compiler loop recognition

Function Level Parallelism

Applications suitable for function level parallelism have a large amount of different, but independent tasks. These logical tasks can execute simultaneously.

Referring back to the teller example. The customer now wants to make deposits into three accounts, withdraw money from one account and transfer money between two of the accounts. All accounts are empty when starting. Again, if one teller is working on all the transactions, the sequential sequence will include deposit the money, withdraw and transfer. With multiple tellers (processors) the 3 deposits can happen simultaneously, followed by the transfer between accounts and the withdrawal. Since the accounts were empty to begin with you don’t want the fourth teller starting the withdrawal before the deposits are made. With function level programming the programmer has to think about these synchronizations

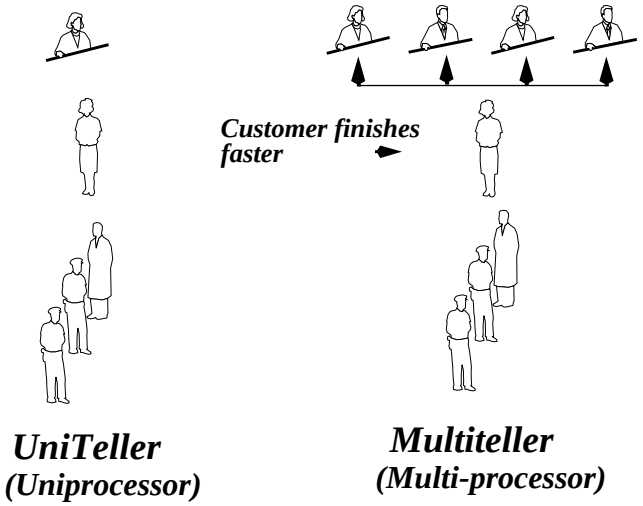

Figure 4 Bank Teller Example

This customer wants to deposit the payroll checks for 100 employees in the company -- same operation, different data. If the customer has one teller (refer to Figure-4) working on this transaction, it will occur sequentially, one deposit after another, until the transaction is completed. Let's assume it takes 2 minutes to complete one deposit.

Time to completion = 2 * 100

It will take 200 minutes or 3 hours to complete.

Referring to the right side of Figure-4 there are multiple tellers (multiple processors). Now the customer has 4 tellers to work on 100 deposits. The first tellers takes 25, the second teller takes 25, etc.

Time to completion = (2*25)

It will take 50 minutes, or less than 1 hour to complete. All the processors can work in parallel on this transaction.

The Solaris kernel can thus execute different components -- file system, process management, streams, -- in parallel across different CPUs in one system. As a result, throughput of Solaris SMP systems increases proportionally with the number of CPUs.

Developing Apps. for Multi-threading and Multi-processing

Completing the total solution for developers of compute intensive applications are tools which provide a flexible range of options for creating parallel applications -- breaking up an application into independent tasks that can be run in parallel on multiple processors. These tools enable software developers to get maximum application performance from multi-threading and multi-processing by combining ease of use and the opportunity to balance programming effort with resulting performance.

Two Parallel Programming Models

Applications tend to lend themselves to parallelization in two distinct ways: loop level parallelism and function level parallelism.

Loop Level Parallelism

Applications suitable for loop level parallelism are those in which there are areas where the same operation is being performed repeatedly on different data. These operations are found within loops.

An example of loop level parallelism using bank tellers follows.

other with several synchronization primitives such as, mutex locks, semaphores, condition variables and reader writer locks. Threads can also synchronize access between processes sharing the same structures.

The user threads library manages the interaction of threads with the kernel. Solaris has a unique two level threads implementation that makes it possible to write applications with thousands of threads with each thread being very light. While the application developer writes application to the threads interface, the threads library uses the lower layer of Light Weight Processes, or LWPs to interact with the kernel. The two level model is designed to give the application developers the freedom to create as many threads as they wish without making the application excessively heavy and slow.[1] The thread context is kept in user space and the kernel does not know about user level threads. When one of the threads in a process is ready to execute, it is scheduled by the threads library on top of an LWP. LWPs are heavier objects whose context is kept in the kernel. The LWP enters the kernel and performs the task of the scheduled user thread. In most applications the number of LWPs or the number of the required number of concurrent accesses into the kernel will be lower than the total number of threads. For instance, each button in a GUI could be a thread but only a few of them are pressed or activated at the same time. The two level model thus ensures that the application only uses as many system resources as needed at any given time and the system is not burdened with managing threads that are not active.

Solaris threads track the POSIX standard 1003.4a draft quite closely and once the standard is set, Solaris will provide a fully compliant interface.

Symmetric Multi-processing

The Solaris 2 kernel incorporates many of the same performance enhancing techniques to optimize itself for symmetric multi-processing. Specifically, the Solaris 2 kernel has been restructured around threads. Threads are used for most asynchronous processing, including interrupts. The resulting kernel is fully preemptible and capable of real time response. Solaris provides a robust base for highly concurrent, responsive operation.

Solaris 2 kernel threads are in themselves light weight with a small stack. Switching between kernel threads is relatively inexpensive. Kernel threads are fully preemptible and may be run from any of the scheduling classes in the system, including the real time (fixed priority) class.

1. It also gives developers the ability to write optimized applications without knowing the number of processors on the target machine.

SunSoft Solaris Features

While the SPARCserver 1000 offers many benefits, such as power, scalability, extensibility, it is the SunSoft Solaris 2 operating system that enables these benefits. Solaris 2 has been enhanced and highly optimized to increase the performance of compute intensive applications. In addition, it offers features that allow application developers to optimize their code to take advantage of Sun's high performance systems.

SunSoft's Solaris 2 is a fully preemptible, fully multi-threaded, symmetric multi-processing (SMP) kernel that now supports user level multi-threading. The power and performance of Solaris is scalable from small uniprocessor systems to large multi-processor systems.

Multi-threading

Solaris has always provided multi-tasking, or the ability to execute many processes simultaneously. There is no logical limit to the number of processes that can execute at once, and many instances of the same program can be executing at one time. This design allows for a time sharing of the system resources.

User level multi-threading extends the multi-tasking capability significantly further by allowing developers to break up a single application into multiple threads that can be acted upon at the same time. User level threads will significantly increase the performance, throughput and responsiveness of applications by allowing them to exploit the parallelism of the hardware. An application can now have multiple threads scheduled on multiple processors.

Multi-threaded servers will also now become much more responsive as they independently handle requests from all clients. A slow client or a large request therefore will not block other requestors.

Solaris multi-threading is thus key to exploiting the benefits of the advanced hardware by building parallel applications. The user thread library that is included in Solaris 2.2 has many advanced features that make it easy to program to this interface. Solaris threads are very lightweight with respect to system resources required. Creating and synchronizing with threads is significantly faster and simpler than dealing with the fork system call and synchronizing between processes.

Threads share all the address space and most of the data for a process. They are however completely independent of each other and execute code, take page faults and make systems calls independently of each other. Threads can synchronize with each

The SBus complies with the SBus Specification Revision B.0, including all burst sizes (including 64 byte) and parity support. The SBus is rated at a theoretical peak of 80 MB/sec, but in practice, utilizing stream mode DVMA each SBus can sustain 55MB/sec write and 49 MB/sec read rates.

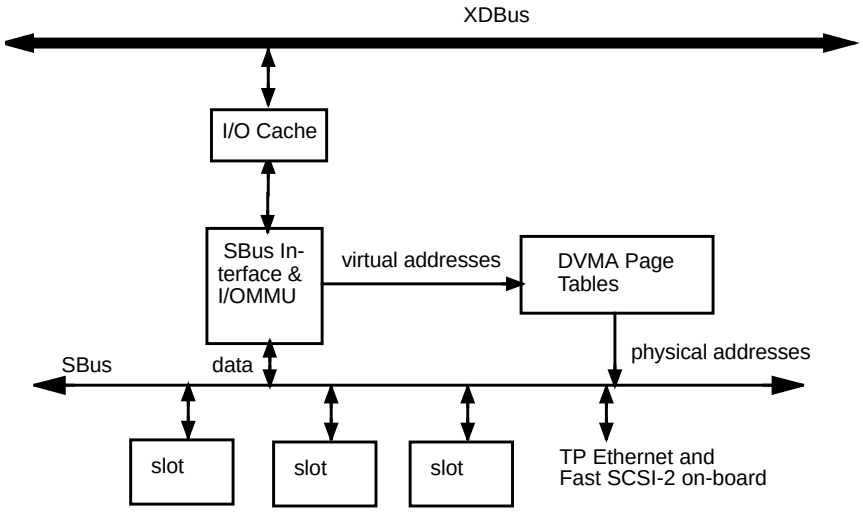

Figure 3 SPARCserver 1000 I/O subsystem

System Bus Architecture

The SPARCserver 1000 provides a high speed system bus known as the XDBus. The XDBus is a packet switched bus with a peak rate of 320 MB/sec, but more importantly, sustainable throughput of 250 MB/sec. The XDBus is used to connect the SuperSPARC modules to the memory subsystem, the I/O subsystem, and each other.

As with all bus designs, only one requester or servicer can be using the bus at a time. Therefore, it is important to minimize the amount of time in which a device is in control of the bus and to eliminate non-optimal use. For example, in the case of a "circuit switched" bus, the requester arbitrates for the bus, places the target address on the bus and then holds the bus while the request is being serviced. This is similar to placing a phone call. In the case of a long distance call, you pay for the entire length of the call even though you may not be talking (using) the phone line (bus) the entire time.

The unique advantage of the XDBus is its "packet switched" design. A packet switched bus permits substantially greater overall throughput than comparable circuit switched buses by separating bus requests from their corresponding replies. A requester arbitrates for the bus, sends a request packet that specifies the target address and operation then releases the bus to make it available for other activity. While the request is being serviced, the bus is free to perform other activities. All packets on the bus are tagged so that a request can be associated with its reply.

I/O Subsystem

The SPARCserver 1000 provides up to four[1] independent high performance I/O units. The units provide for high throughput as well as low latency by using optimized block transfer modes, a non-blocking I/O cache, and a low contention SBus mechanism.

Each of the four units provides a complete SBus peripheral expansion bus with three slots. Included on each unit is an on-board Ethernet and Fast SCSI port. Each SBus is connected to the backplane by an SBus Interface (SBI) ASIC and includes an MMU dedicated to I/O processing, and an I/O cache.

1. There is one I/O unit per system board.

Memory Subsystem

Multiple processors increase the likelihood of memory contention as there are more processors accessing memory. The SPARCserver 1000 has many features designed to eliminate memory contention. Each of these features speeds up the processing by reducing the likelihood that a processor will need to wait for one of its supporting functional components.

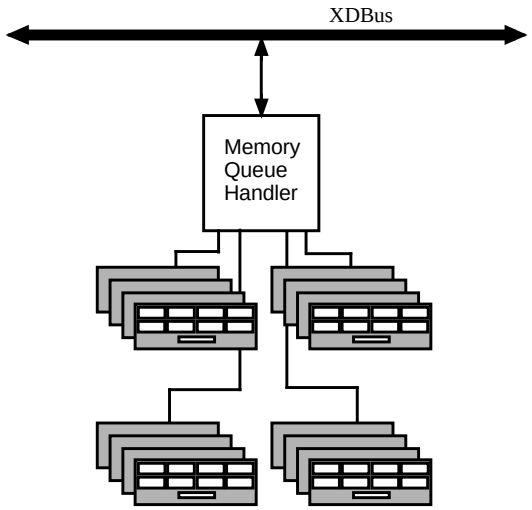

Figure 2 SPARCserver 1000 memory subsystem

A SPARCserver 1000 can be configured with up to 2GB of main memory, using 16 Mbit DRAM chips. High performance is ensured by offering extensive interleaving and a high performance SIMM organization. The memory system has also been designed to transparently include non-volatile SIMMs to effectively accelerate synchronous disk transfers.

Individual DRAMs are not able to provide data on a continuous basis. After each access the chip must spend time recovering before permitting the next access. Interleaved memory helps reduce the cost of memory access by permitting multiple memory components to operate in parallel. The SPARCserver 1000 supports up to 4-way interleaving. This means that 4 separate banks of memory can supply data in parallel. The memory system is organized so that each bit of a 64 bit word is on a different DRAM, thus with ECC the system can tolerate the loss of an entire DRAM.

Features

Features of the SPARCserver 1000 include:

- One to eight SuperSPARC CPUs, each with 1 MB of SuperCache
- 32 MB to 2GB RAM, with NVRAM support
- Up to 8.5 GB internal disk, 100GB external disk
- Internal CDROM and 4mm DAT
- One to four SBuses, three to twelve SBus slots
- Up to four on-board SCSI-2 and TP Ethernet interfaces

Upgradeable Superscalar Processors

Like other Sun systems, the SPARCserver 1000 systems use processor modules that are user upgradeable. Processor modules are designed for easy upgrades, since processor technology advances more rapidly than other system technologies. The SPARCserver 1000 systems are designed to accommodate several generations of processor modules.

The SPARCserver 1000 incorporates up to eight 50 MHz SuperSPARC CPUs. Each CPU is a 3-way superscalar implementation of the SPARC Version 8 architecture. The 3.1 million transistor processor contains an integer unit with two ALUs, an FPU, a branch processor and a SPARC reference MMU. It has a 20KB, 5-way set associative instruction cache and a 16 KB, 4-way set associative data cache. The split cache (or Harvard) architecture allows simultaneous access to data and instructions. Up to four instructions can be fetched per cycle via an internal 128 bit bus. Each system board[1] on the SPARCserver 1000 can contain one or two SuperSPARC modules. Each SuperSPARC module contains a SuperSPARC processor, SuperCache cache controller, and 1 MB of unified (instruction and data) secondary cache.

As a result, each superscalar SPARC chip is capable of completing three instructions each clock cycle when executing from the internal cache and when the instruction stream has been properly scheduled. In this way, the SPARC superscalar processors are able to achieve the goal of a high performance compute server, getting more work done faster, by executing multiple instructions per clock cycle.

1. The SPARCserver 1000 can be configured with 1-4 system boards.

SPARCserver 1000Hardware Design

In order for a computer system to benefit from an SMP architecture, it needs to reduce the potential for bottlenecks due to the increased number of processors requesting information. In other words, the architecture must be balanced. As more processors are added, other aspects of the system must also be able to expand to accommodate the additions and avoid bottlenecks. These counterparts include the memory subsystem, bus, peripherals, networking capabilities and supporting software. The logical organization of the SPARCserver 1000 system board follows.

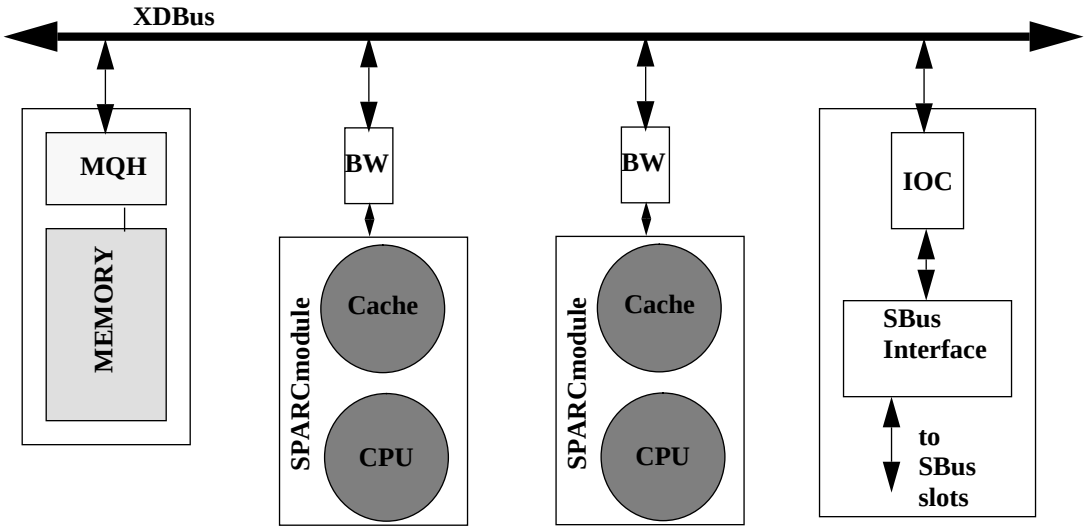

Figure 1 The logical organization of the SPARCserver 1000 CPU board. Note all functional units are connected to the XDBus backplane. The memory queue handler MQH controls access to main memory. The bus watcher (BW) maintains cache consistency. The I/O controller (IOC) provides the buffered interface between the XDBus and the I/O subsystem.

technology. Single chip microprocessor performance is closing the gap with custom designed processors. Combining these processors in a multi-processing configuration results in performance that often exceeds that of vector and massively parallel machines.

The delivery of cost effective compute power, however, entails much more than just fast CPUs. It requires careful attention to modularity and system balance.

Modular Design

A modular designed machine has two primary advantages. First, customers need only enough hardware to get started. As demand grows, additional components can be added to the system to expand its capabilities. For example, a multi-processor machine can start with two or four processors and then configure additional processors as demand requires.

A substantial cost of any computer system is in its chassis and peripherals. Therefore a modular, upgradeable machine lets customers protect their investment. When faster processors or higher capacity memory becomes available, these components can be easily be upgraded in the current system, rather than replacing the entire machine.

Balanced System

Another element of a compute server is a need to be “balanced.” A balanced compute server goes beyond raw compute power. As processor clock speeds increase data is processed faster. This results in the need to have a faster supporting structure so that these fast processors do not end up waiting for their counterparts, such as memory, bus, disks, and networks. These supporting subsystems must have enough capacity and must be available without bottlenecks.

SPARCserver 1000 Compute

Introduction

Some computer tasks require an extraordinary amount of computing power. In today's computing landscape there are few computers distinguished by their ability to deliver exceptional computational power. These are usually expensive and therefore scarce resources. Typically these computers are servers shared by many people in the enterprise.

Sun foresaw the increased demands that are currently being placed on computer systems. There are applications today that need power beyond what single CPU systems can provide, including ECAD, MCAD, databases, and scientific and financial analysis. Sun has made a strategic push to deliver multi-processing (MP) systems, both on the desktop and in server environments. Perhaps more importantly, future applications such as complex systems modeling, object oriented applications, and multimedia, require the power of multi-processing systems.

The Sun SPARCserver 1000 provides cost effective multi-processing through modularity, hardware design features and innovative software tools. Sun has achieved this by taking advantage of the SPARC architecture, by careful attention to system design that benefits multi-processing, and by focusing on application speedup. As a result Sun's MP systems lead the industry in delivered price performance in the compute marketplace.

Requirements of a Compute Server

A compute server has one primary requirement: deliver cost effective compute power -- to take a computationally intensive task and get it done more quickly. This is done by improving the throughput of the tasks and reducing latency. The combined result is getting more work done in a shorter time.

The second part of this requirement is to provide this improved throughput at lower cost. To date, typical compute engines have been composed of expensive, customized components. To provide cost effective computing, the vendor must leverage today's evolving technologies: VLSI, integration of components, and manufacturing

Abstract

Many computer tasks require an extraordinary amount of processing power. This paper discusses computing requirements and solutions in the context of the SPARCserver 1000. Topics covered include the requirements of a compute server, a functional overview of the SS1000 hardware design, operating system features of interest, and the development of applications for multithreading and multiprocessing.

Contents

Please
Recycle

SPARCserver 1000 Compute White Paper

Technical Product Marketing

Sun Microsystems Computer Corporation
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 8xx-xxxx-xx
Revision X, May 1993