SPARCserver™ MP Systems New Technology for Flexibility, Scalability, and Growth

Technical White Paper

Sun Microsystems
Computer Corporation
A Sun Microsystems, Inc. Business

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Contents

Introduction

With the introduction of the SPARCserver 600MP series in 1991, Sun Microsystems® brought multiprocessing capabilities to its family of midrange database, file, and compute servers, extending the product line with new flexibility, scalability and expandability. Multiprocessing is a key technology in Sun's server strategy for the '90s. With multiprocessing, Sun™ can provide customers with high application performance, without compromising SPARC binary compatibility for thousands of existing applications.

The SPARCserver 600MP series also offers an innovative modular systems architecture that ensures a unique balance of CPU and I/O scalability. With the addition of SPARCserver 10, the same modular technology is introduced in a desktop server for the workgroup.

Utilizing a design with modular CPUs on a single system board, Sun has been able to reduce cost, provide easy upgradability and ensure a growth path to future technology. Combined these features enable Sun to offer customers unprecedented performance, flexibility, and investment protection more cost effectively than any other multiprocessing systems.

This paper examines how Sun implemented modular multiprocessing and how it affects end users, system administrators, and software developers perspectives. It also covers several other features of the SPARCserver 600MP and SPARCserver 10 systems designed to optimize performance for real-world applications.

Why Multiprocessing? 1

Sun's hallmark has been its commitment to high-performance systems, a recent example being its innovation of the SPARC® RISC architecture. With the SPARC scalable architecture, customers have seen processing power increase with each new implementation, and future designs will continue to provide higher performance. Another way to increase available processing power is to incorporate more than one CPU in the system, an approach called multiprocessing (MP). Multiprocessing has several benefits:

- Independent tasks can be handled by separate processors, increasing job throughput, the number of transactions, and/or the number of simultaneous users.
- Combining CPUs in one system lowers the overall cost of computing by sharing system resources like memory, disk, and network interfaces.
- Providing a high-speed interconnect between these multiple CPUs also provides the ability to achieve better coordination and faster interaction between related tasks.
- Depending on the type of application and tools available, a single large job can be divided into several smaller tasks that can run simultaneously for faster application time-to-completion.

Sun has introduced multiprocessing products for several important reasons. Multiprocessing technology is starting to move out of specialized high-end applications. This is indicated by two recent trends: the inclusion of MP in UNIX® standards, and the development of open UNIX database products that take advantage of MP. Sun's products are increasingly being used in the commercial computing environments and larger technical departments, characterized by many users and large client/server networks. Extending the product line with MP systems is important for providing the flexible range of solutions and configurations customers are seeking to replace existing large, proprietary, centralized equipment. Tighter integration of CPU and memory technologies makes MP implementations more affordable.

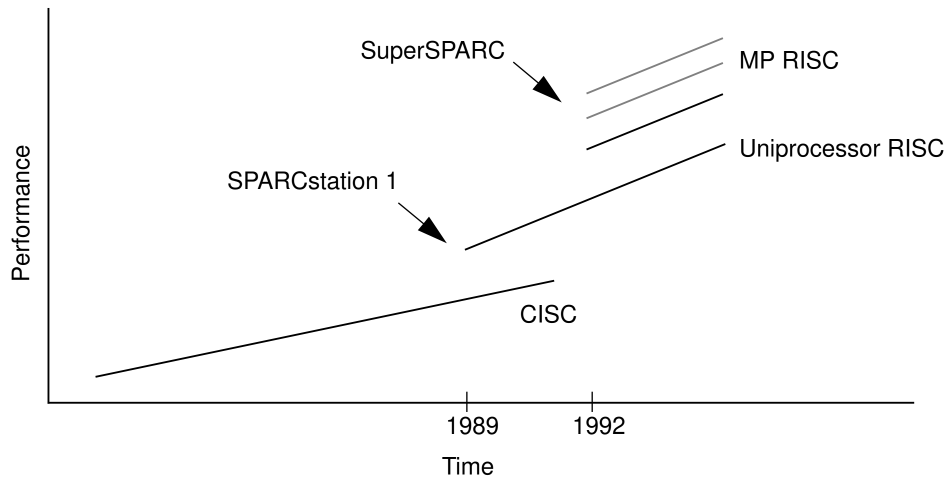


Figure 1-1 System performance in the future will require a multiprocessing RISC architecture

Even with the advent of the latest high-performance superscalar RISC chips, such as Sun's SuperSPARC™ microprocessor, customers are demanding system performance increases greater than chip technology alone can provide. Another advantage of multiprocessing is that it offers Sun's customers the opportunity to rise above the single-chip power curve and achieve greater throughput improvement for their server applications.

Sun's objectives for multiprocessing are focused on delivering these benefits and a migration path that will enable Sun customers to protect their current investment in applications.

Sun's first MP implementation, Solaris® 1.0.1, provides binary compatibility with existing applications, so every MP system running multiple tasks can realize the immediate benefits of finishing more work in a given period of time. Sun's Solaris 2.0 software, based on the System V Release 4 operating system, will improve the underlying implementation by offering a multithreaded kernel to increase both user tasks and system tasks. Solaris 2.0 software will provide standard interfaces for multiprocessing so that software developers can achieve portability for MP applications. The Solaris 2.0 operating environment is source code compatible with the Solaris 1.0.1 environment, easing application migration costs to SVR4.

Sun is also working on a full software development environment for parallel programming, with multitasking compilers to automatically parallelize existing source code, and parallel libraries and tools to support more sophisticated optimizations. The net result will be more applications using affordable multiprocessing to support larger problems and handle more users to deliver solutions faster.

Besides Sun's efforts to provide complete multiprocessing environments, customers can count on a wide range of applications and development tools from Sun partners. A key to Sun's multiprocessing strategy is the modularity that reduces cost and consequently drives increased unit volumes. Sun has rapidly become the volume leader in UNIX MP systems, further motivating independent software vendors (ISVs) to develop software specifically to exploit Sun's multiprocessing systems.

Multiprocessing Hardware Implementation 1

The key to Sun's multiprocessing picture is the tightly coupled, shared-memory model shown in Figure 2-1. Processors are tightly coupled by a high-speed system bus, called the MBus, and share the same image of memory, although each processor has a high speed cache where recently accessed data and instructions are stored. A single copy of the operating system coordinates the activities of all processors, scheduling jobs and coordinating access to common resources. All CPUs have equal access to system services, such as I/O and networking, unlike some designs where specific resources are attached to a specific CPU. The overall objective is a general-purpose system with sufficient compute and I/O resources to improve performance for a wide range of applications.

.

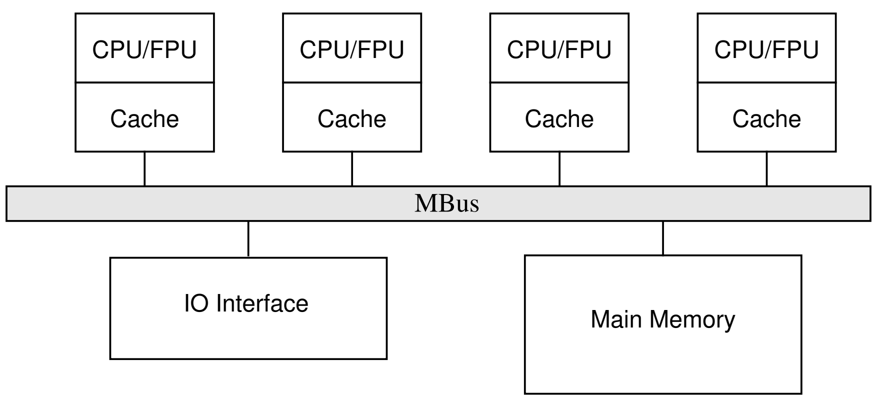
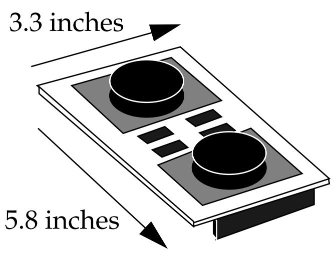


Figure 1-1 SPARCserver 600MP System Shared Memory Model

SPARC Modules and the MBus

With processing power increasing at a rapid pace, one of Sun's primary implementation goals for its future systems is to separate CPU technology from the remainder of the product design. In the SPARCserver 600MP and the SPARCserver 10 systems design, the standard MBus connector provides the interface between two separate boards. The main system functions and I/O are on the system board, while CPUs are located on modular daughtercards, called SPARC modules. Providing a standard interface between CPUs and the rest of the system enables modular upgrading to future processor implementations. It also provides a convenient way to add more processors to a basic system, providing inexpensive scalability to increase system performance as application needs grow. Adding or changing processors is as easy as installing a SPARC module. As future CPUs become available, customers can swap SPARC modules to achieve higher performance. With a stable system platform, both Sun and the customer benefit from a much faster development process and improved time-to-market.



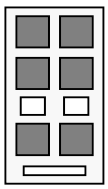
SPARC modules are small daughtercards that plug directly on the main system board

With the flexibility to choose the number of CPUs, and scalability to add power with more or faster CPUs, the SPARCserver 600MP and SPARCserver 10 systems offer the industry's best growth path and investment protection.

The MBus specification for this standard CPU interconnect is widely available through SPARC International to both component and system developers. With a potential product base larger than just Sun, CPU suppliers have greater incentive to develop new SPARC modules for the MBus. Also, they can spread the development costs over a larger number of units, with both Sun and its customers benefitting from lower product costs and wider choice of available processors.

SPARC modules are the size of an SBus card (3.3 inches x 5.8 inches), and attach to the MBus connectors on the system board in much the same way as SBus cards. A processor core consists of an integer unit (IU), a floating-point unit (FPU), a memory management unit (MMU), a cache controller and MBus interface unit (CC/IU) and cache memory. This functionality can be integrated into varying numbers of chips. A single SPARC module can support up to two processor cores for a maximum of four CPUs on a single system board. There are two version of SPARC modules that can be installed into the SPARCserver 600MP series. The CY6002 Cypress/Ross module and the SuperSPARC with SuperCache™ (external cache) module, shown in Figure 2-2. In the

SPARCserver 10 system, there are also two versions of SPARC modules available: the SuperSPARC microprocessor and the SuperSPARC microprocessor with the SuperCache (external cache) controller.

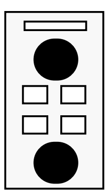


Cypress/Ross module

The two processors found on the CY6002 SPARC module offer the following features:

- 64 Kbyte direct mapped virtually addressed instruction/data cache
- SPARC reference MMU with 64 entry fully associative translation lookaside buffer (TLB)
- 32-byte store buffer and 32 byte read buffer

The SuperSPARC module combines the SuperSPARC microprocessor and the SuperCache controller onto a single module. The SuperSPARC processor offers the following features:



SuperSPARC module

- A single chip with integer, floating point, memory management and cache
- Superscalar pipeline with up to three instructions per clock cycle
- Internal 20-Kbyte 5-way set associative physically addressed instruction cache
- Internal 16-Kbyte 4-way set associative physically addressed data cache
- SPARC reference MMU with 64 entry fully associative translation lookaside buffer (TLB) for hardware page-table walking
- Support for 65,536 contexts
- Internal 64-byte store buffer
- Single/double precision on chip FPU (which also does integer multiply and divide)

The SuperCache external cache controller, offers a 1 megabyte direct mapped physically addressed instruction/data cache.

The SuperSPARC processor and SuperCache controller run asynchronously with the MBus. So, as faster SPARC chips become available, an upgrade is as simple as swapping modules. The SuperSPARC module provides a minimum of twice the performance, on CPU test suites and application throughput, as the CY6002.

Advanced CPU architectures, such as the SPARCserver 600MP series with SPARC modules, will share many of the same performance features. For example, both of the above SPARC modules have integrated instruction and data caches that accelerate the speed of the CPU by storing the most recently referenced instructions or data in fast private memory. Another important feature for CPUs to share is a standard memory management scheme such as the SPARC Reference MMU (SRMMU). This assures compatibility and portability of application binaries. Store buffers are also important to maintaining high throughput by temporarily holding the data, until it can be written out to external cache or main memory. This enables the execution unit to continue on to the next operation immediately rather than waiting many clock cycles for access to a slower resource.

Other features found in the SuperSPARC architecture provide for even faster operation such as a superscalar pipeline. This means that there are multiple arithmetic logic units (ALUs) enabling more than one instruction to be executed during any given clock period. Also new to the SPARCserver 600MP series with the SuperSPARC processor is on-chip set associative caches, which speed up execution further by using faster on-chip cache memory efficiently.

With the use of advanced packaging and IC processing technologies even denser faster modules will be possible. For example, a dual-processor the SuperSPARC with SuperCache module will be offered where the two SuperSPARC processors, each with 1 megabyte of external cache, fit into the standard 3.3-by-5.8-inch MBus form factor. The smaller interconnect dimensions and loads will further facilitate CPU operation at higher clock rates.

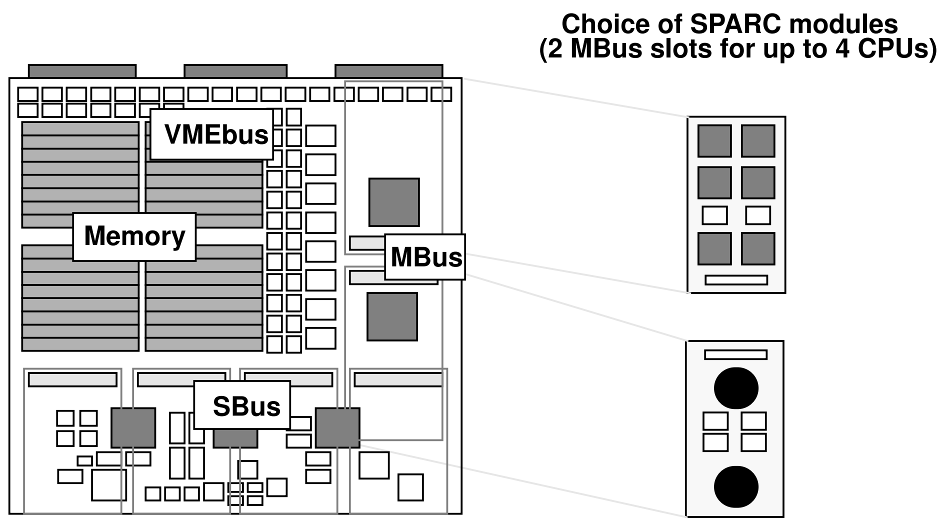


Figure 1-2 SPARCserver 600MP System Board and SPARC Modules

Cache Coherency

When there are multiple local caches in a multiprocessing system, some mechanism must ensure that local copies of data remain consistent. To maintain this data consistency, all caches in the system participate in a multiprocessor *cache coherence* (consistency) protocol. The SPARC MBus provides transactions that make it possible to implement such coherence in a simple way. DVMA is also able to participate in the coherence protocols, eliminating the need for processor cache-flushing around I/O operations.

Because cache coherency is supported both in the MBus specification and in the cache control logic, there is little overhead and little impact on the performance of individual CPUs. With caches snooping the MBus for coherent transactions, and intervening to maintain consistency, copyback (or writeback) operation is possible. The copyback cache only issues stores to memory when flushing or reallocating a modified cache line, reducing MBus occupancy significantly compared to a write-through cache. This causes writes to main memory each time writes to the cache occur, putting substantial and unnecessary loads on the bus. All CPU-to-CPU and CPU-to-memory transactions occur on the MBus, which is less expensive and complex than the specialized synchronization buses used on older MP implementations. Appendix A covers MBus cache coherency in greater detail.

Memory Management & DVMA in an MP Architecture

Some system facilities can be shared with multiple processors in a system, while others need to be replicated per processor. An example of replication is the memory management unit (MMU). Older Sun systems contained a single MMU consisting of several levels of static RAM tables indexed by a context number and by some bits of the virtual address. This older MMU was shared between the processor and DVMA.

Multiprocessor servers place different requirements on memory management. More contexts are necessary than the traditional 8-64 supported in previous designs. More pages should be mapped at any time in order to reduce the frequency of page faults. Also, the MMU should be close to the processor and cache for speed reasons. This means that the MMUs must be on the module, and therefore must be physically small and lower in cost than an array of fast, expensive static RAM chips.

These needs are met by the SPARC Reference MMU (SRMMU). SRMMU is based on tables in memory that contain the multiple-level mappings. Each cache/MMU controller contains a context register and a base address register that points to the first level of tables in memory. A set of translation lookaside buffers (TLBs) that contain the set of recently used translations is also integrated. TLB misses are serviced by hardware, meaning that fewer traps into the operating system are needed to manage mappings once a page has been faulted in by the memory management code. Stale mappings get purged from the TLBs through use of MMU *flush* commands from the operating system.

The SPARC Reference MMU implementations used on the SPARC modules allow up to a maximum of 4096 contexts for the CY6002 and up to 65,535 context for the SuperSPARC design. See Appendix B for a more detailed description of its organization.

The SRMMU in each processor is not accessible from DVMA, so another memory management unit is implemented to translate DVMA addresses to system physical addresses. This unit is the I/O memory management unit (IOMMU). The IOMMU is very similar to an SRMMU, except that there are no contexts, only one level of table, and no “referenced” or “modified” status per page. The IOMMU contains 16 TLB entries with least-recently-used (LRU) replacement. It provides up to 2 Gbytes of virtual address space to device drivers, extending the limit of 1megabyte that was imposed in the older, shared-MMU systems.

DVMA through the IOMMU participates in system coherence with hardware support, so no processor cache-flushing is needed around I/O. DVMA activity can come from SBus devices (including on-board SBus devices such as SCSI and Ethernet) and also from VME on the SPARCserver 600MP systems. A DVMA address can come from an SBus master or a VMEbus master. DVMA can access main memory or any slave devices on the SBus. The only limitation is that DVMA cannot access devices on the VMEbus (although VMEbus masters can access VMEbus slaves since the CPU does not participate in those accesses). See Figure 2-3 for the different memory access modes supported.

Slave \ Master	CPU	SBus	VME (600MP only)
Memory	✓	✓	✓
SBus	✓	✓	✓
VME (600MP only)	✓		✓

Figure 1-3 Memory Access Modes Supported

Interrupts from I/O devices can be steered to any processor in the system, meaning that the same hardware can support either a symmetric or asymmetric multiprocessing operating system.

Multiprocessing Software Implementation 3

The operating system releases for the SPARCserver 600MP series were designed to provide the simplest transition for customers moving to multiprocessor computers. The initial operating system release for the SPARCserver 600MP system product is the Solaris 1.0.1 operating environment (SunOS™ release 4.1.2 software) which provides binary compatibility for all "well-behaved" SunOS 4.X software-based applications as well as a minimum of new administrative procedures so that the SPARCserver 600MP systems can be integrated into existing networks in a few hours.

Solaris 2.0 system software combines the benefits of symmetric multiprocessing with those of an operating system based on the industry standard: the System V Release 4 (SVR4) UNIX operating system.

Figure 3-1 illustrates shared memory multiprocessing with Solaris 1.0.1 or 1.1 system software. One Solaris kernel image controls the entire system, with each CPU running separate UNIX processes. All processes share a common pool of physical main memory managed by the operating system virtual memory system. Individual CPUs are not generally visible to one another or to the processes executing on a different CPU. Any process can execute on any CPU, and the Solaris kernel is free to execute on any CPU. Processes typically migrate from CPU to available CPU many times during their execution lifetimes. A single scheduler schedules the UNIX processes to run on any of the available CPUs. For speed and flexibility, interrupts are mapped to and serviced by whichever CPU is currently executing kernel level code.

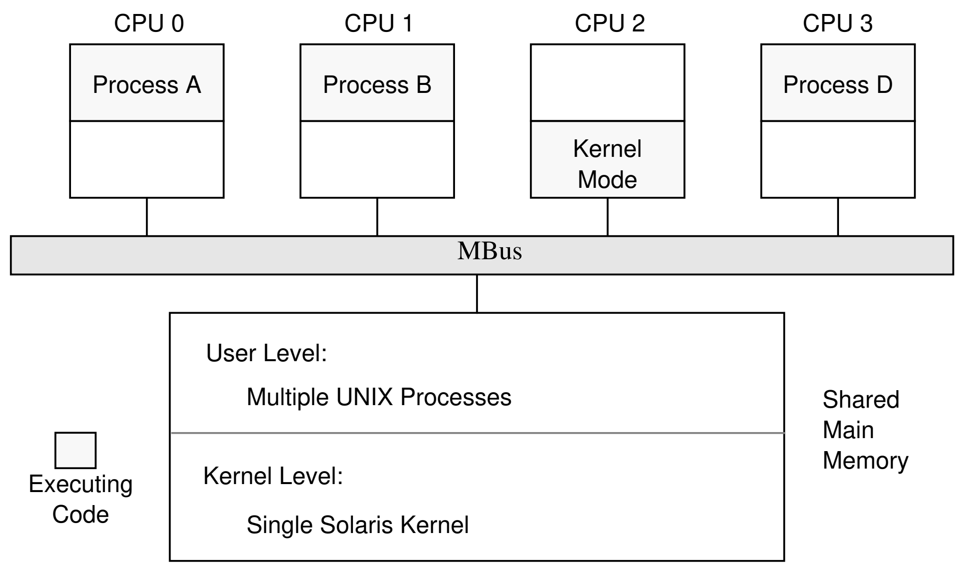


Figure 3-1 Solaris versions 1.0.1 and 1.1 Multiprocessing

When multiple CPUs need to access a shared data structure in an orderly fashion, software primitives called *locks* are used to synchronize access to shared data, particularly the kernel data structures. These locks are often referred to as mutual exclusion locks, or *mutexes*. Solaris versions 1.0.1 and 1.1 software use a small number of kernel mutex locks for its operation; only one CPU at a time is executing kernel-level code. In contrast, Solaris 2.0 software uses a larger number of locks and is able to support more threads of execution within the kernel.

The Solaris 2.0 operating environment has a multithreaded kernel for fully symmetric multiprocessing (SMP) support. The difference between Solaris 1.0.1 and Solaris 1.1 software, and the SMP implementation of Solaris 2.0 is that the kernel can execute on more than one CPU at a time. Multithreaded kernel support with the Solaris 2.0 software provides better I/O and NFS performance than Solaris versions 1.0.1 or 1.1 software because of the ability to execute multiple concurrent device drivers and NFS daemons. See Figure 3-2 for an illustration of symmetric multiprocessing with Solaris 2.0 operating environment.

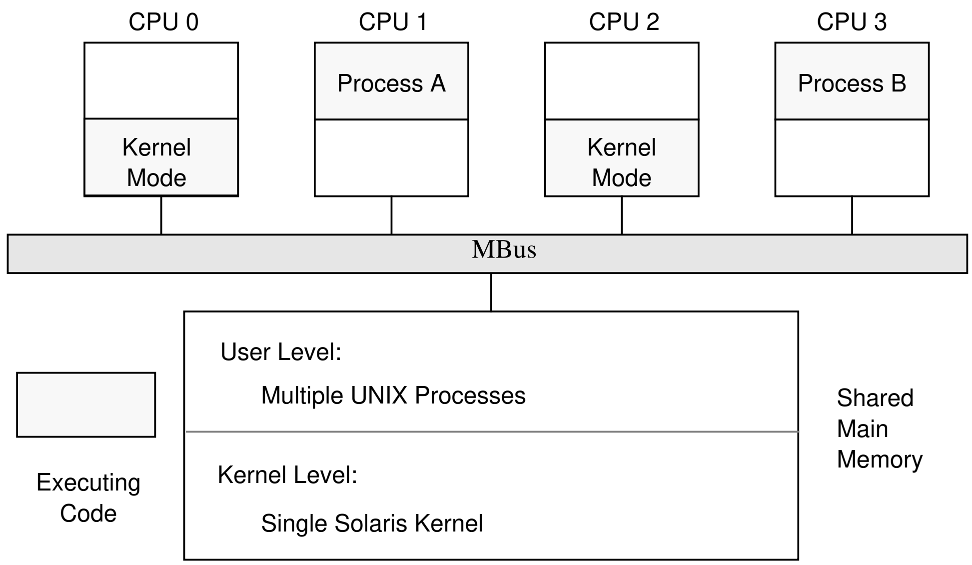


Figure 3-2 Solaris 2.0 Symmetric Multiprocessing

Solaris versions 1.0.1 and 1.1 system software are binary compatible with all existing "well behaved" SunOS 4.X system software based applications. They are also binary compatible with all existing SunOS 4.1 software-based device drivers that restrict themselves to published SunOS interfaces. The system was designed so that the same interfaces provide the same facilities with the same semantics of interaction between the portions of the driver or between multiple drivers as well as between the driver and the system kernel. Binary compatibility for existing applications and third-party peripherals further simplifies the task of integrating the SPARCserver 600MP or SPARCserver 10 into production operations.

The Solaris 2.0 operating environment provides customers with the performance increases of fully symmetric multiprocessing in a standards-based operating system — simply by installing the new software. Solaris 2.0 software-based releases will combine both an innovative multithreading architecture for advanced application development and the proposed POSIX 1003.4 multiprocessing libraries for application portability. Sun's future technology direction for providing a parallelized software development environment will be based on Solaris 2.0 system software. The SPARCserver 600MP and SPARCserver 10 systems were designed to efficiently support either Solaris 1.0.1, 1.1, or Solaris 2.0 software.

Scheduling

The scheduling of jobs is straightforward, using a small extension to the traditional SunOS timesharing scheduler. Scheduling minimizes idle CPU cycles. The Solaris software maintains a single run queue for the entire system. The highest priority runnable process for the entire system is selected from the run queue and started on the available processor.

Processes are free to migrate from one CPU to another during execution. Because of the shared memory and cache coherency support in hardware, the cost of a granting a time quantum to a process on a new CPU is essentially the same as that of continuing it on the last CPU on which it ran. In this way, available CPU cycles can be used as processes become available. The Solaris 2.0 software also has a fixed priority scheduler that provides for real-time UNIX applications.

The traditional UNIX utility, `nice(1)`, can increase the priority of processes to grant them additional CPU time.

Tools for Monitoring Multiprocessing

The Solaris shared memory multiprocessing model simplifies the job of administering and monitoring the MP SPARCserver systems. Because one shared memory image of the Solaris kernel is running on all system CPUs, the theory of tuning Solaris system software for performance is unchanged from previous releases and there are no new MP-specific tuning operations.

All the standard Sun operating system utilities and monitoring tools are included in the Solaris releases on the SPARCserver 600MP and SPARCserver 10 systems. This reduces the learning time for administrators working with MP.

Two utilities are provided in Solaris system software in the `/usr/kvm` directory. The `mps(1)` utility prints the status of every process and the last CPU on which it ran. The `mpstat(1)` utility prints the percent of CPU utilization for every configured CPU in the system.

Performance Enhancements for Server Applications 4

A good server requires more than just plenty of processing power: high I/O bandwidth, overall system throughput, and sufficient connectivity are key elements in server environments. The SPARCserver 600MP and SPARCserver 10 products were designed with careful attention to balancing speed and throughput.

SBus as Primary I/O Bus

An onboard SBus enables the user to take advantage of high-performance and low-cost peripherals from Sun and from third parties. There are a total of four SBus slots on both the SPARCserver 600MP series and the SPARCserver 10. On the SPARCserver 600MP, three SBus slots are available in the absence of a second MBus module. With 32-byte activity, the SBus provides sustained performance, assuming an ideal master, of 53 Mbytes per second writes to memory, and 30 Mbytes per second reads from memory.

VME and I/O Cache Improvements on the SPARCserver 600MP

VMEbus support provides a bridge to older configurations and to widely available VME peripherals. There are up to 11 VMEbus slots, depending on which model of the 600MP family you have. The VMEbus channel provides a 32-Kbyte I/O cache (IOC) to accelerate sequential activity between VMEbus masters and system memory. This IOC provides VME masters with an 8-Mbyte window of virtual address space with which to read and write memory, and enough buffers to support up to 1024 simultaneous transfers in progress. The

I/O system is tuned to provide optimal performance for masters that do 32-byte burst accesses. IOC activity matches that burst size by buffering VME writes into 32-byte packets, and doing 32-byte reads on behalf of multiple sequential VME reads. VMEbus activity through the IOC will provide a sustained 16-Mbyte/sec writes and 13-Mbyte/sec reads. Both numbers assume an ideal master.

Optimizing Bus Transactions

With many masters sharing the same buses, wait-states on a bus can degrade overall performance. The the SPARCserver 600MP and SPARCserver 10 design minimizes bus occupancy through the use of store buffers (see Figure 4-1) and in some cases through split transactions. Store buffers can be found on each module. They are also provided in the memory controller (two 32-byte swing buffers) and between the MBus and the I/O section of the machine, as well as on the processor-to-VMEbus interface. DVMA utilizes dual swing buffers as well as stores to memory. Store buffers is to accept a store operation, free up the master and the intervening bus, and then carry that operation to the next stage of bus or device. All store buffers in this design are strongly ordered and provide the capability for software synchronization.

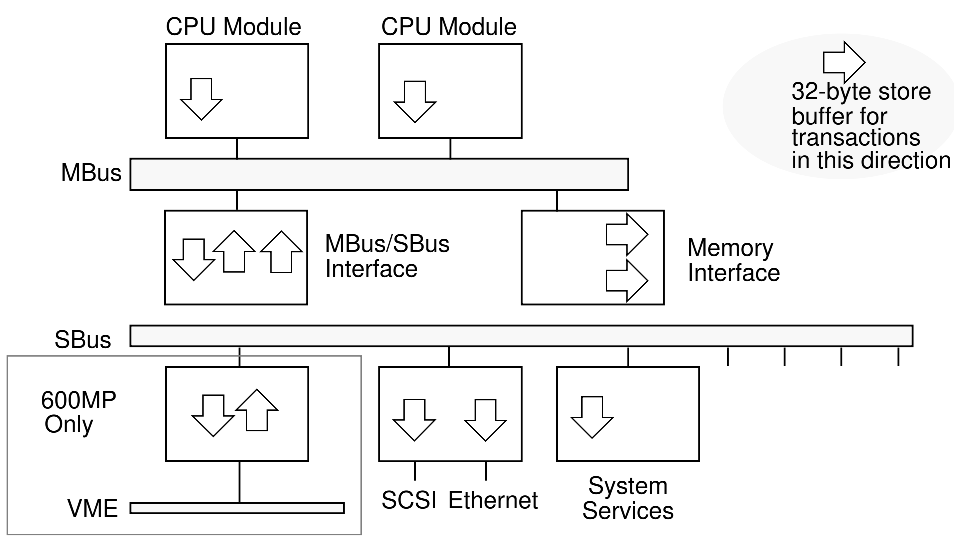


Figure 4-1 System Block Diagram with Store Buffers

Connectivity

Standard features on every SPARCserver 600MP system are an onboard Ethernet and SCSI-2 (10 Mbytes per second) channels and two synchronous serial ports. Optional features include additional buffered Ethernet and SCSI channels, additional serial and parallel ports, FDDI, Token Ring, HSI, and a wide range of IBM, DEC, and OSI connectivity products.

This wide range of options extends the configuration flexibility of the SPARCserver 600MP series by supporting a variety of network topologies and fitting into environments with existing equipment.

Summary 5

The the SPARCserver 600MP series and the SPARCserver 10 offer new technologies for higher application performance while protecting investments in existing hardware and software. Multiprocessing, and easy upgrades to future SPARC CPU technology, provide scalability and growth for Sun's mid-range and high-end servers. Hardware features like the SPARC Reference MMU, the separate I/O MMU, Sun's I/O cache, and store buffers increase application throughput in a wide variety of server environments. With both SBus and VMEbus for I/O expansion, Sun customers benefit from new low-cost, high-performance SBus options while maintaining compatibility with previous investments in VME technology. And Sun's MP software strategy, with binary compatibility in the Solaris 1.0 and Solaris 1.1 environments, and sophisticated multithreading in the Solaris 2.0 operating environment, provides the right flexibility for protecting existing investments and taking advantage of innovation.

The new technology in SPARCserver MP systems provide the flexibility, scalability, and growth to employ client-server solutions throughout the enterprise.

SPARCserver Family Overview 6

The SPARCserver family provides a choice of servers that combine an innovative multiprocessing design, flexible I/O, disk and memory expansion, network connectivity, into a powerful lineup of binary compatible server products.

The SPARCserver 10 system ideal as a file, print, or applications server providing multiprocessing capability to the workgroup or connectivity to the PC LAN.

The SPARCserver 630MP general-purpose system is well suited to a broad range of workgroup server applications with more room for expansion and growth than a desktop server. This system provides an excellent database management system for small departments, for both higher transaction performance and a lower price than traditional entry-level super-minicomputers.

SPARCserver 670MP system is Sun's most expandable office-environment server. It offers the balanced performance benefits of multiprocessing, with more memory and expandability than the SPARCserver 630MP. Appropriate in a wide variety of roles, from dedicated compute server to departmental database system, the SPARCserver 670MP system blends the best features from Sun's mid-range and high-end servers.

The SPARCserver 690MP system is Sun's premier server, providing high system throughput, fast I/O performance, enhanced unattended backup, greater expandability, and the storage capacity to support larger environments with centralized database, fileserver, compute, or multiuser system requirements. It also has unique availability features. The SPARCserver 690MP system more closely meets application needs that once required mainframe and super-minicomputers, yet it does so at significantly lower prices.

Cache Coherency A

MBus specifies two levels of operation. Level-1 MBus is a simple protocol that enables *read* and *write* operations of sizes ranging from 1-128 bytes. Level-2 MBus adds four other transactions; *coherent read*, *coherent read and invalidate*, *coherent write and invalidate*, and *coherent invalidate*. With this simple set of transactions caches can maintain a consistent (or *coherent*) image of memory at all times.

Coherence among multiple caches is possible because of cache block ownership. Copyback caches are used in order to minimize bus traffic, so memory is allowed to contain *stale*, or out-of-date, data. If a line has been written to, the processor/cache that has the most recent copy is considered to be the *owner* of that block. This means that if any other processor/cache (or DVMA) wants to read that data, the data must be supplied by the owner rather than by memory. A read of cacheable data is done with a *coherent read* operation. Whenever any coherent operation goes by on the bus, each participating cache monitors the transaction type and address. This is known as *snooping*. If a cache recognizes that it is the owner of the requested block (a *snoop hit*), it will abort the memory access and provide the up-to-date copy of the block instead. The term for this is *cache intervention*.

One complication in this scheme is having a virtual cache rather than a physical cache, since two different virtual addresses could map to the same physical address. In this processor implementation, all aliases map to the same cache line to ensure that the consistency scheme works properly.

If more than one cache has a copy of the same block, that block is considered to be *shared*. If only one cache has a copy then the block is *exclusive*. In order to maintain a coherent memory image, the various copies of a *shared* block in all of the caches must match at all times. In the general case of a multiple cache environment, when one processor/cache modifies a block it can either alert all other caches to update their copy (*write broadcast*) or to discard their stale copy (*write invalidate*). MBus implements a *write invalidate* protocol. Any time a write is done to a *shared* line, the cache will issue a *coherent invalidate* transaction on the MBus. The other caches that have a copy of that block will discard it.

A line will be marked as *shared* in all caches that have a copy if it was provided by cache intervention. It will also be marked as shared if a snooping cache sees that it has a copy but is not the owner when another processor/cache allocates this block. In both cases the status is signaled to the receiving cache by all caches that have a copy.

Each cache line has five states: invalid, shared clean, exclusive clean, shared modified, and exclusive modified (see Figure A-1). The latter two are the states that indicate that this cache *owns* this block. Only one cache at any time will have ownership of a particular block. When a processor/cache wants to store to a line that is *invalid*, it will issue a *coherent read and invalidate* operation; this gets the most recent copy of this block from whoever owns it (either memory or some cache) and instructs all caches (including the supplier of the data) to discard their copy of the block. A write to a *shared* line will simply generate a *coherent invalidate* on the MBus. Obtaining a copy of a block and assigning it to a cache line is called *allocation* of the line. If the line was already in use for a different block, then that block gets *displaced*.

The *owner* of a block is responsible for writing the valid information back to memory. This will occur either when a *modified* block is displaced (that is, when the cache allocates that line to a new block) or when the operating system explicitly flushes that cache line. This *copyback* operation is done with a simple *write* on the MBus. A *coherent write and invalidate* is used only for 32-byte writes to cacheable space (that is from DVMA) or for coherent write-through caches, which Sun does not use.

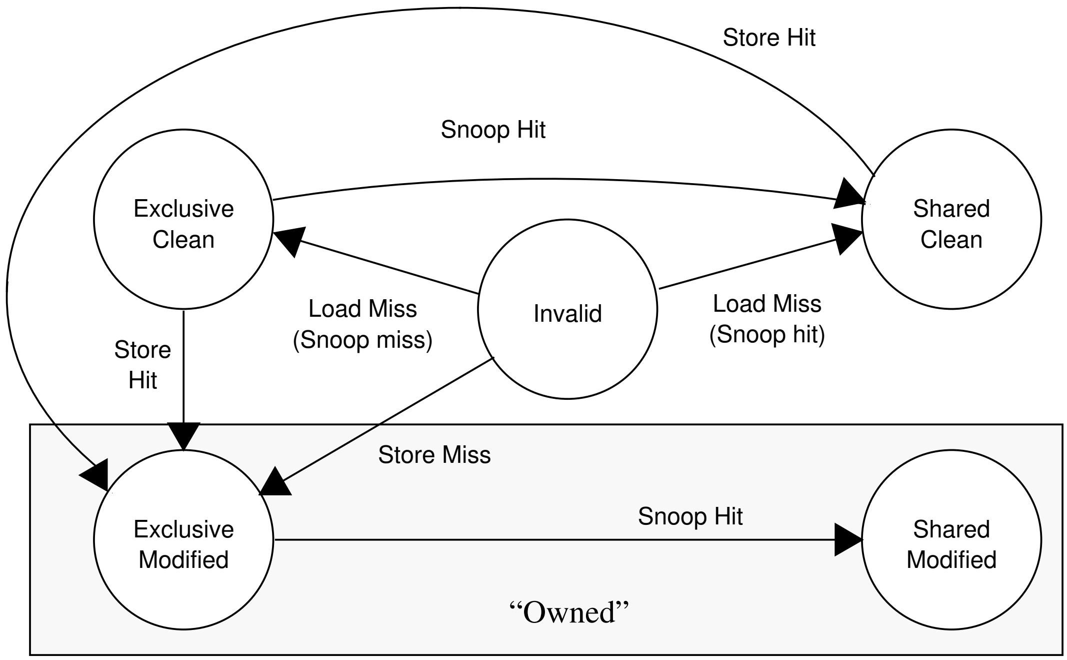


Figure A-1 Cache Coherency State Diagram

Simultaneous stores to the same block are sorted out by bus arbitration. A block may not be updated in a cache until that cache has exclusive ownership of that block. An atomic load-store sequence is treated in the same way as a store. All of this coherence protocol is transparent to software.

SPARC Reference Memory Management Unit B

SRMMU has four levels of tables, corresponding to *context*, *region*, *segment*, and *page* in the traditional Sun MMU design (see Figure B-1). Entries in these tables contain either a page table pointer (PTP) or a page table entry (PTE). A PTP points to the next level of table; a PTE contains a physical address, protection information, and other page status. A PTE can be found at any level in the translation process; if found at the *page* level it maps a 4 Kbytes page. A PTE found at the *segment*, *region*, or *context* level maps a page of 256 Kbytes, 16 Mbytes, or 4 Gbytes respectively. When servicing a TLB miss, the MMU controller does a *table walk* to find the translation; if no valid translation is found, a page fault occurs.

Currently only 4-Kbyte pages are used by SunOS system software, but as the kernel is tuned it will be able to take advantage of larger page groupings (as segments or regions) to minimize TLB misses. Each processor operates in its own context, but the set of contexts is shared among all processors in the system.

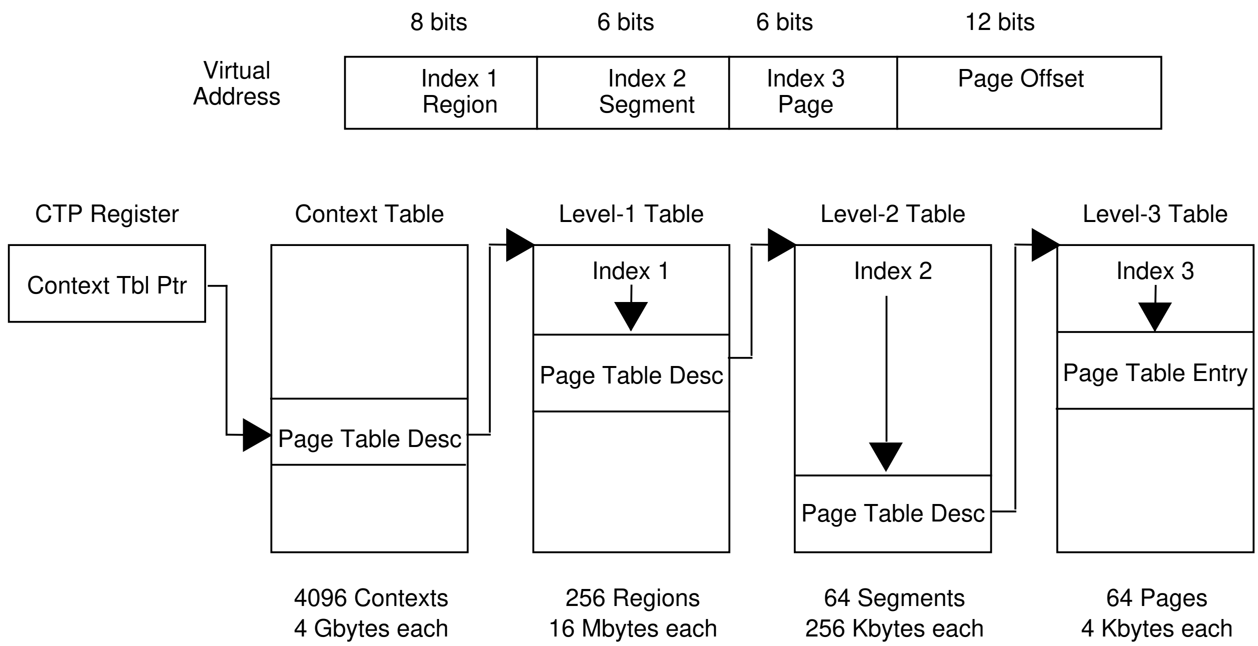


Figure B-1 SPARC Reference MMU Table Structure

Sun Microsystems
Computer Corporation
A Sun Microsystems, Inc. Business

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

Australia: (02) 413 2666
Belgium: +32 2 759 38 11
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) 3221-7021
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: (01) 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

Printed in USA

4/92 FE345-0/30K