CHAPTER 1

Introduction to Sun Fire Systems

Most companies want the best solution for their needs, especially when they are purchasing a computer system. However, designing a reliable system that performs well takes careful consideration and planning.

Consider a widely-used analogy—automobiles. A vast range of different types of vehicles, including sedans, sports cars, convertibles, sport-utility vehicles, trucks, and hybrids is available. Each of these types of vehicles has advantages and disadvantages. When trying to decide what type of vehicle to purchase, you must consider your needs. How many people must it be able to carry? How much power does it need? How much cargo must it hold? Is size a concern? And so forth.

You would probably be making a mistake if you purchased a two-seater sports car when you had a family of five that regularly attended soccer games. You would probably be making a similar mistake if you bought a sedan with the intent of using it to haul your trailer and dirt bikes out to the desert on the weekends. For a purchase to be worthwhile, you must ensure that it properly fits your specific requirements.

This analogy may seem trite, but many people do not put nearly as much thought into purchasing a high-end server as they do a car. Instead, a system often is purchased based on its processor speed or size alone. Little, if any, consideration is given to concerns such as how to layout the I/O, how much memory to get, and how much expansion capacity is really needed.

As with the car analogy, you must ensure that the design of your server configuration meets your company's needs. This design process requires time and effort, but it pays off in better system reliability, availability, serviceability (RAS) and performance.

This chapter summarizes the RAS and performance features of the Sun Fire system hardware. The definitions of the RAS features are:

- *Reliability*—The ability of the system to run without interruption, to continue to operate when correctable errors are detected, and to prevent data corruption.
- *Availability*—The percentage of time the customer's system is able to do productive work. The ability to always recover after a failure by testing and bypassing failed components.

- *Serviceability*—The system ensures that repair time (downtime) is minimized.

This chapter covers these topics in two sections—RAS and Performance.

RAS

The RAS goals for the Sun Fire system are to protect the integrity of the customer's data and to maximize availability. The focus is on three areas:

- Problem detection and isolation—knowing what went wrong and ensuring that the problem is not propagated
- Tolerance and recovery—absorbing abnormal system behavior and fixing or dynamically circumventing it
- Redundancy—replicating critical components

To ensure data integrity at the hardware level, all data is error correction code (ECC) protected, and address and control buses are protected by parity checks. These checks ensure the containment of errors.

For tolerance of errors, resilience capabilities are designed into the Sun Fire system to ensure that the system continues to operate, even in a degraded mode. The Sun Fire system can function with one or more processors disabled. In recovering from a problem, the system is checked quickly to determine the fault and to ensure minimum downtime. To reduce downtime, redundant hardware can be configured into the system.

Reliability

Sun Fire systems have five categories of reliability capabilities:

- Reducing the probability of errors
- Detecting and correcting errors using error correction code (ECC)
- Detecting uncorrectable errors with ECC and parity checking
- Redundant power and cooling
- Environmental sensing

Availability

Availability is the ability of a system to be continually accessible and useful to the customer. Sun Fire systems have many features that contribute to this quality, including the ability to:

- Test, identify, and de-configure failed components following a system interrupt
- Configure and boot a usable configuration with a subset of the original configuration
- Change the configuration without interrupts using dynamic reconfiguration (DR).

For higher levels of availability Sun Fire systems can be clustered.

Serviceability

To reduce repair time, the Sun Fire systems are designed with a number of maintenance capabilities and aids. These are used by the Sun Fire system administrator and by the service provider.

Failing components are listed in the failure logs in such a way that the field-replaceable unit (FRU) is clearly identified. You can remove and replace most system components in a properly configured system during system operation without scheduled downtime. If properly configured, CPU/Memory boards, I/O boards, I/O controllers, fans and power supplies can all be replaced while the system is running.

Sun Fire RAS Features

CPU and System Interconnect

The most important reliability feature in any system is the protection of data integrity.

The UltraSPARC® III processor has parity protection for all major internal caches and ECC protection for transactions to and from the external caches.

The data interconnect has ECC and parity protection throughout the system. The address interconnect has parity protection.

CPU Error Protection

The CPU corrects errors detected in the internal data and instruction cache SRAMs. When an internal error is detected, the CPU invalidates the cache and retries the data or instruction load.

Two SRAM modules per CPU reside on the CPU/Memory system board. These modules contain the external cache (Ecache). The CPU corrects all parity errors detected during data cache or instruction cache by invalidating and flushing the cache line. The hardware corrects all single bit errors detected by ECC during data transfers on the fly. ECC also detects and reports any uncorrectable (multibit) errors to the Solaris OE.

System Interconnect Error Protection

The system has three types of error protection—data interconnect, address interconnect, and error isolation.

Data Interconnect

The system protects all data interconnect pathways by using ECC and parity protection. It generates ECC and parity bits for all data blocks sourced by processors and PCI I/O controllers (system devices). All data switches in the path for each transfer check ECC and parity. The receiving system device checks and corrects ECC.

Address Interconnect

The system has parity protection for all address interconnect pathways and checks parity between all system devices and address repeaters. On the Sun Fire ™15K/12K systems there is also ECC protection on the address and address response crossbars for transactions across the centerplane.

Error Isolation

Because each of the data switches in the path for every data transfer checks ECC, the source of ECC errors can be identified in most cases. However, some types of ECC errors are difficult to isolate. For ECC errors, such as a CPU writing bad ECC into memory, finding the source is difficult because multiple system devices may read and report the bad ECC. In such cases, the ECC error usually can be isolated to a dual CPU data switch or its pair of processors. This is an improvement over the previous architecture in which it was more difficult to isolate these types of ECC errors.

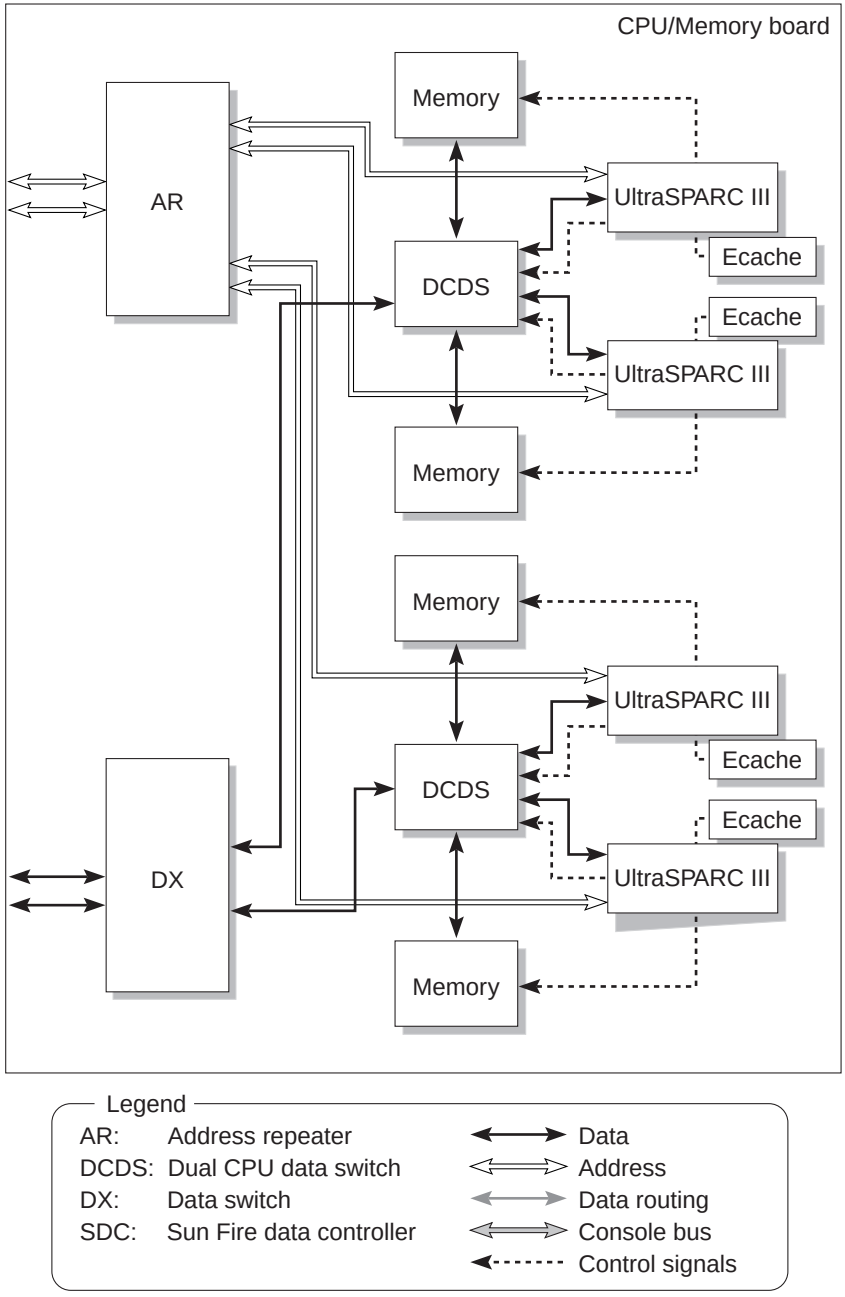


FIGURE 1-1 CPU Board Block Diagram

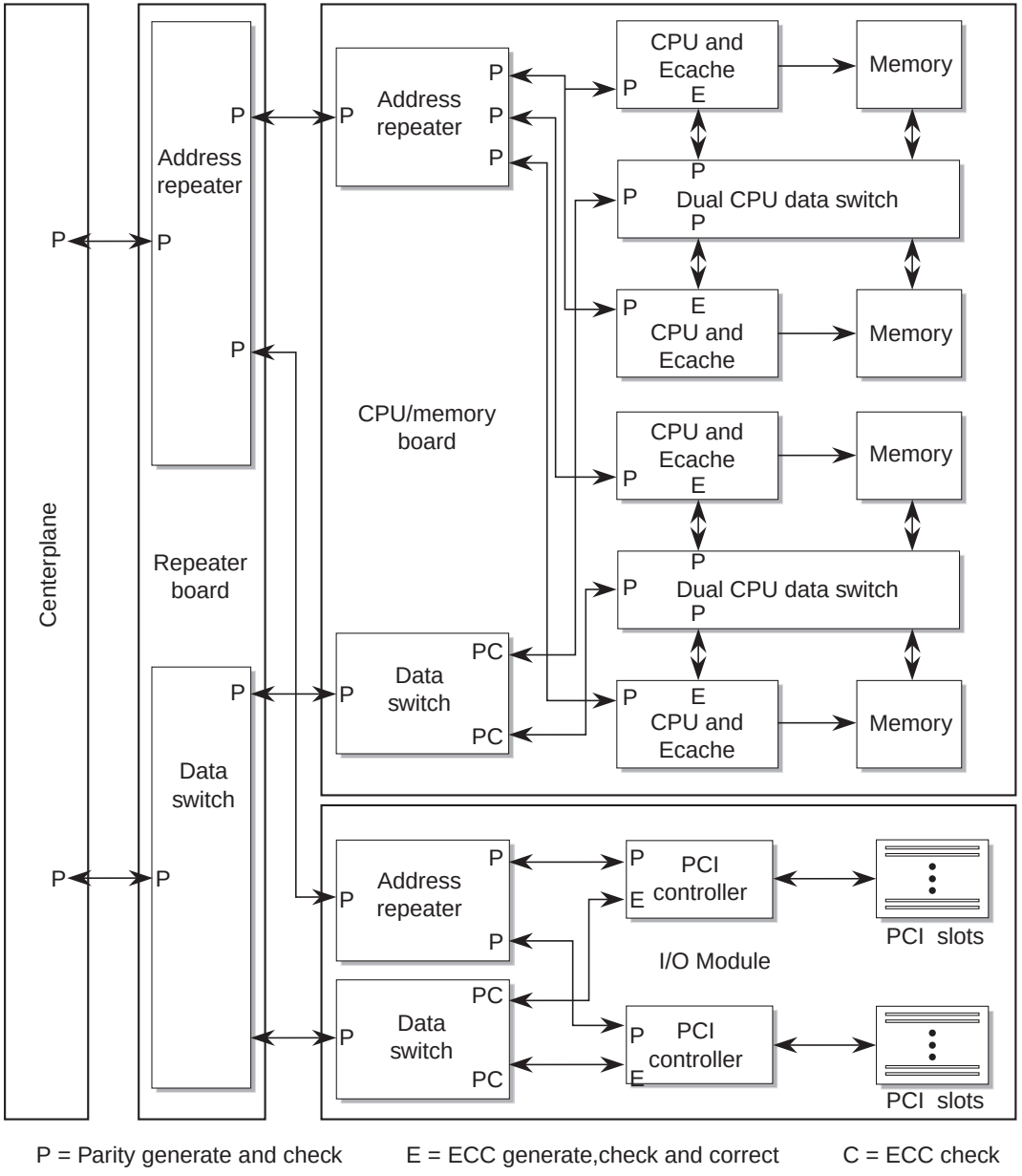
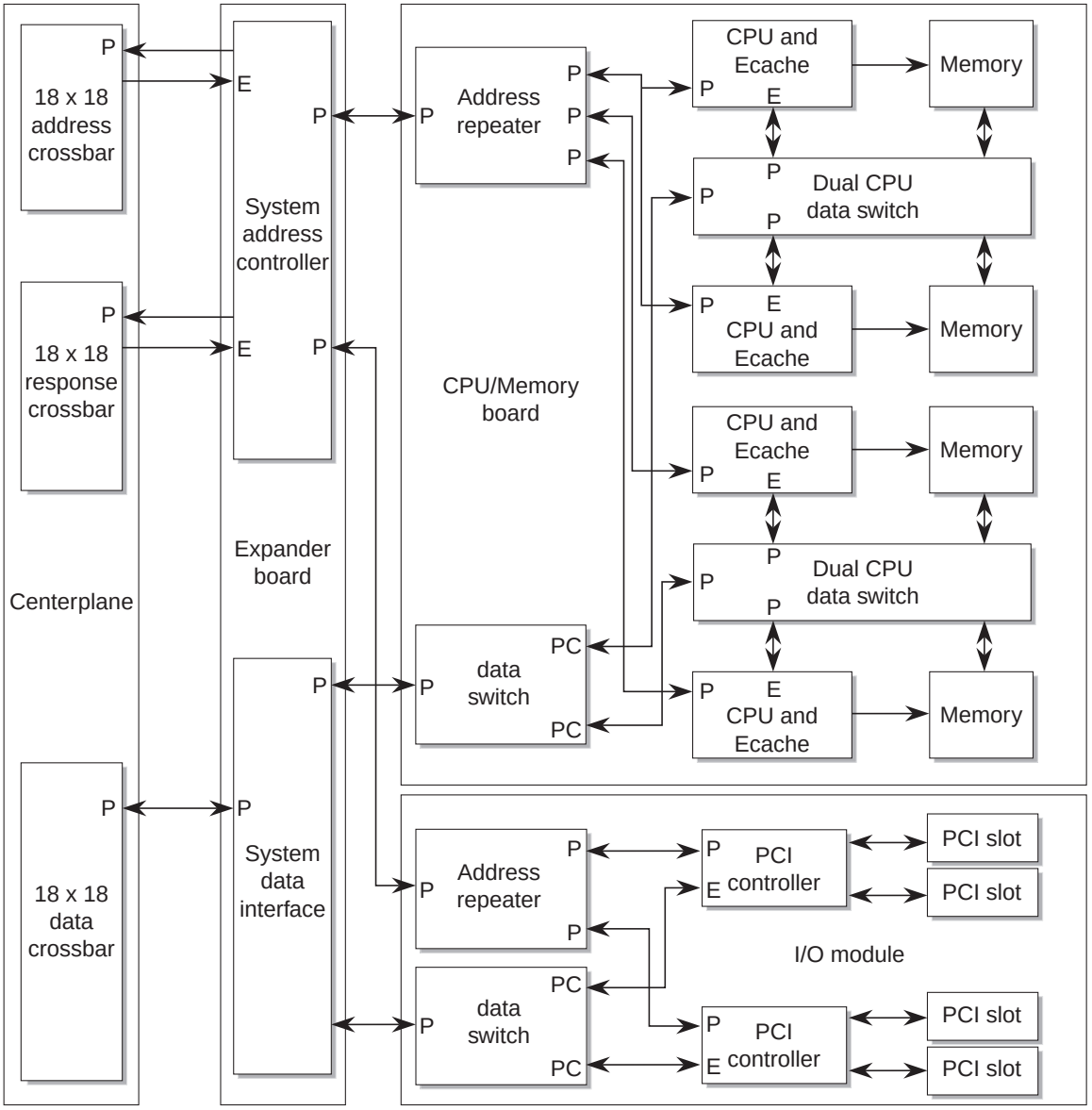


FIGURE 1-2 Sun Fire 6800/4810/4800/3800 Systems Interconnection Diagram



P = Parity generate and check E = ECC generate,check and correct C = ECC check

FIGURE 1-3 Sun Fire 15K / 12K Systems Interconnection Diagram

System Controller

A small system called the system controller (SC) manages the Sun Fire systems. The SC is responsible for all of the functions required to test, configure, and boot domains. It supplies all system clocks and virtual TOD clocks for the domains and monitors power and environmental status. It also provides domain control with virtual key switches and console connections for each domain.

Testing and Configuration

Upon power on or a reset, the SC runs a self test called SCPOST. It then starts the platform management software.

The SC configures and coordinates the initialization, testing, and boot processes. After a system failure, and when the virtual key switch of a domain is turned on, the SC powers on the system components associated with the domain and runs SPOST for the domain. SPOST controls the running of LPOST and IPOST. LPOST tests the CPU/Memory boards, I/O boards, and the system interconnect. IPOST tests the PCI controllers. The SC then configures the domain based on the components that pass the tests and starts the boot sequence.

Environmental Monitoring

The SC monitors the following conditions:

- Voltage, current, and temperatures for power supplies
- Voltage and temperatures for all system boards and processors
- Temperatures of ASICS
- Fan status and speed

If safe thresholds are exceeded, the SC shuts down components to prevent damage to the system.

System Administration and Maintenance

The SC provides access for platform and domain administration. Access to the SC is by the included RS-232 serial connection or by network connection.

Tasks performed at the platform level are:

- SC setup and configuration
- Allocation of system resources for domains
- Domain creation

- Status display of all domains
- Power control for all system components
- Logical enable and disable of components
- Individual CPU/Memory board tests
- Component and environmental status display
- Platform error message administration
- Platform password setup (Sun Fire 6800/4810/4800/*3*800 systems)
- Platform and Domain Security configuration (Sun Fire 15K/12K systems)

Tasks performed at the domain level are:

- Power and boot control
- Domain status display
- Logical domain component enable and disable
- Individual CPU/Memory board tests
- Domain error message administration
- Domain password setup (Sun Fire 6800/4810/4800/*3*800 systems)

Redundant System Components

All systems in the Sun Fire system product line can be configured with redundant components. The ability to run with a subset of configured components increases availability of the system. As long as one processor with memory and one I/O module are functional, the system can run. If the system is configured with redundant connections to storage and network, the access can be maintained using these alternate paths.

Redundant components include:

- CPU/Memory boards
- I/O modules
- PCI cards
- System Controller boards
- Repeater boards (Sun Fire 6800/4810/4800/*3*800 systems)
- Sun™ Fireplane interconnect (Sun Fire 15K/12K systems)
- Fan trays
- Power supplies

CPU/Memory Boards

Depending on the model, Sun Fire systems can support up to 18 CPU/Memory boards. Each board contains two or four CPUs and is capable of running independently or together with other boards in a larger domain.

I/O Modules

Depending on the model, Sun Fire systems can support up to 18 I/O modules:

- Sun Fire 15K system supports up to 18 I/O modules, with four PCI slots each.
- Sun Fire 12K system can have up to nine I/O modules with four PCI slots each.
- Sun Fire 6800 system can be configured with a combination of four I/O modules with eight PCI or four Compact PCI slots each.
- Sun Fire 4810/4800 can be configured with two I/O Modules with eight PCI or four Compact PCI slots each.
- Sun Fire 3800 system is configured with two I/O modules with six Compact PCI slots each.

TABLE 1-1 summarizes these configurations.

TABLE 1-1 I/O Module Configurations

System	I/O Modules	PCI Slots	Compact PCI Slots
Sun Fire 15K	18	4 each	0
Sun Fire 12K	9	4 each	0
Sun Fire 6800	4	8 each	or 4 each
Sun Fire 4810/4800	2	8 each	or 4 each
Sun Fire 3800	2	0	6 each

PCI Cards

Redundant PCI cards can be configured to provide alternate paths to all peripheral connections.

System Controller Boards

With two SC boards configured in the system, a failure of the primary SC does not cause a domain interrupt. The system clocks, virtual TOD clocks, and all other SC functions fail over to the secondary SC without causing a domain failure.

Repeater Boards

The Sun Fire 6800/4810/4800 systems contain pairs of Repeater boards. The Repeater boards contain data and address repeater switches. The Sun Fire 6800 contains two pairs of Repeater boards. If one board fails, the system can run in degraded mode on the other pair of boards. The Sun Fire 4810/4800 system is configured with one pair of Repeater boards. If one board fails, the system can run in degraded mode on the other board. The Repeater boards can be replaced while the system is running. Although the Sun Fire 3800 system has the same ability to run in degraded mode in the case of a failure, the functionality of the Repeater boards is built into the centerplane and cannot be replaced without an interrupt of the system. TABLE 1-2 summarizes these configurations.

TABLE 1-2 Repeater Board Configurations

System	Repeater boards
Sun Fire 3800	Functionality is built into the centerplane
Sun Fire 4810/4800	1 pair
Sun Fire 6800	2 pairs

Sun implements the Sun Fire 15K/12K interconnect differently than the midrange systems. The interconnect consists of three independent 18 way crossbars—one for data, one for addresses, and one for address responses. Some of the interconnect is made of multiple ASICs that reside on the centerplane that cannot be replaced while the system is running. If one of the crossbars fails, the system can run in degraded mode while the other crossbars continue with full bandwidth. Depending on the failure, the degraded crossbar may be configured to affect a single domain.

Fan Trays

All Sun Fire systems can be configured with redundant fan trays. All fan trays can be replaced while the system is running.

Power Input and Supplies

The Sun Fire 6800/4810/4800/3800 systems can be configured with up to four separate AC input connections. The systems and peripherals can be connected to two different AC input power grids using two Redundant Transfer Units (RTU), each with two redundant transfer switches (RTS).

The Sun Fire 15K/12K systems are configured with six dual AC-DC power supplies, each with two AC input connections. The power supplies convert the AC voltage to 48 VDC. The 48 VDC is supplied to all of the system modules. Each system module contains its own on-board DC-DC converter to supply the lower DC voltages needed by the logic components local to each module. A failure of a DC-DC converter only affects that board.

Domains and Partitions

The definitions of these features are:

Domain—The ability to create logically independent multiple sections within a partition, with each domain running its own operating system. The Sun Fire 6800 system can have up to four domains. The Sun Fire 4810/4800/3800 systems can each have up to two domains. Each instance of the Solaris Operating Environment (Solaris OE) runs in its own domain. Domains do not depend on each other and do not interact with each other.

- *Partition*—A partition differs from a domain in the level of isolation. The Repeater boards are logically isolated from each other so the system functions as two separate servers. On a Sun Fire 6800 system, segments can be configured to reside completely within a single internal Sun Fire 6800 power grid. The Sun Fire 15K/12K systems interconnect does not use Repeater boards and does not support multiple partitions.

A Sun Fire system can be logically divided into multiple domains. Since each domain is comprised of one or more system boards, a domain can have anywhere from two to 106 processors (on the Sun Fire 15K system). Each Sun Fire system has at least one domain to support the main functionality of the system.

Additional domains can be used for:

- Testing new applications
- Operating system updates
- Configuring several domains to support separate departments

Each domain runs its own instance of the operating system and has its own peripherals and network connections. Domains can be configured without interrupting the operation of other domains on the same system.

While production work continues on the remaining (and usually larger) domain, there is not any adverse interaction between any of the domains. You can gain confidence in the stability of applications or upgrades without disturbing production work. When the testing work is complete, the system can be rejoined logically without rebooting (there are no physical changes when you use domains). Thus, if problems occur, the rest of your system is not affected.

The Sun Fire 15K/12K systems can be configured with up to 18/9 domains. The Sun Fire 15K/12K systems do not use partitions. The Expander boards are responsible for domain separation.

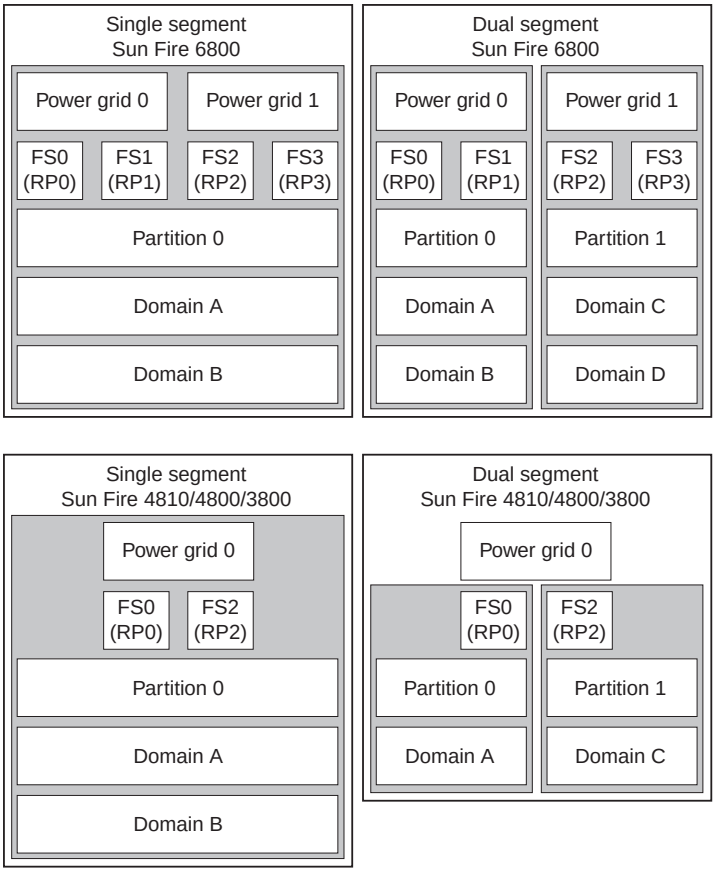


FIGURE 1-4 Sun Fire 6800/4810/4800/3800 Domain and Partition Allocations

Mechanical Serviceability

Connectors are keyed so that boards cannot be installed upside down. Special tools are *not* required to access the inside of the system. This is because all voltages within the cabinet are considered extra-low voltages (ELVs) as defined by applicable safety agencies.

No jumpers are required for configuration of the Sun Fire system. This makes it much easier to install new and/or upgraded system components. There are no slot dependencies other than the special slots required for the SC and Repeater boards.

The Sun Fire system cooling-system design consists of redundant, hot-swappable modules. Standard proven parts and components are used wherever possible. Sun designs the field-replaceable units (FRUs) and subassemblies for quick and easy replacement with minimal use of tools required.

Performance

This section describes the performance features of the Sun Fire system hardware.

UltraSPARC III Processor

The UltraSPARC III is a high-performance implementation of the 64-bit SPARC® V9 architecture.

Features of the UltraSPARC III CPU include:

- Full binary compatibility with all UltraSPARC CPU applications
- VIS instruction set for better 2D and 3D graphic processing
- 4-way superscalar
- 14-stage, non-stalling pipeline
- Integrated memory controller
- L1 caches: integrated instruction (32 kilobytes) and data (64 kilobytes)
- L2 cache: 8 megabytes
- MP scalability: Over 1000 CPUs per system
- System bus: Sun Fireplane interconnect at 150 MHz
- Error checking and correction (ECC) in external cache

TABLE 1-3 lists the UltraSPARC III CPU performance improvements over the UltraSPARC II CPU performance.

TABLE 1-3 UltraSPARC III CPU Performance Improvements

Item	Improvement
Clock frequency	Approximately 2X
Data cache size	4X
Instruction cache size	2X
Memory bandwidth	2 to 3X
External cache bandwidth	2X

CPU/Memory Board

The CPU/Memory board is common to all Sun Fire mid-range and high-end systems. The CPU/Memory board holds up to four processors. Each processor has an associated memory subsystem of eight DIMMs, so memory bandwidth and capacity both scale as processors are added. The memory capacity of the board is 32 gigabytes using 1-gigabyte DIMMs. The maximum memory bandwidth on a board is 9.6 gigabytes per second. The CPU/Memory board has a 4.8 gigabyte per second connection to the rest of the system.

I/O Modules

Each Sun Fire system I/O module contains two PCI controllers. Each controller provides one 66 MHz PCI bus, and one 33 MHz PCI bus. Each PCI bus contains two or more slots for PCI cards. A Sun Fire I/O Module has a 2.4 gigabyte per second connection to the rest of the system.

System Interconnect

All Sun Fire systems use the Sun Fireplane interconnect architecture, which is the coherent shared-memory protocol used by the UltraSPARC III/IV processor generation. Sun Microsystems uses an improved system interconnect with each new processor generation to keep system performance scaling with CPU performance.

The Sun Fireplane architecture is an improvement over the previous generation Ultra Port Architecture (UPA). The system clock rate is increased by fifty percent from 100 megahertz to 150 megahertz. The snoops-per-clock is doubled from one half to one. Taken together, these improvements triple the snooping bandwidth to 150 million addresses per second. The maximum data bandwidth for the Sun Fire 6800 and smaller systems is 9.6 gigabytes per second—triple that of the previous generation Sun Enterprise™ 6500/5500/4500/3500 systems.

The Sun Fireplane architecture also adds a new layer of point-to-point directory-coherency protocol, for use in systems that require more bandwidth than a single snooping bus can provide coherency for. This facility allows coherency to be maintained between multiple snooping buses, and is used in the Sun Fire 15K/12K systems. TABLE 1-4 lists the Sun Fire system interconnect specifications.

TABLE 1-4 Sun Fire System Interconnect Specifications

	Sun Fire 3800 system	**Sun Fire 4810 / 4800 systems**	**Sun Fire 6800 system**	**Sun Fire 15K/12K systems**
System clock	150 MHz			
Coherency protocol	Snooping			Snooping on each board set, directory across centerplane
Address interconnect	1 snooping bus			18 snooping buses, 18x18 global address crossbar, 18x18 global response crossbar
CPU/Memory board internal bisection bandwidth	4.8 Gbytes/sec			
CPU/Memory board external bandwidth	4.8 Gbytes/sec			
I/O board external bandwidth	2.4 Gbytes/sec			
Inter-board data interconnect	4 x4 crossbar	5 x 5 crossbar	10 x10 crossbar	18 3x3 crossbars, 18x18 global crossbar
Same-board bandwidth	9.6 Gbytes/sec			121 Gbytes/sec
Different-board bandwidth	4.8 Gbytes/sec	9.6 Gbytes/sec		43/21.6 Gbytes/sec

Centerplane Implementation

The centerplane implementation depends upon system size. The Sun Fire 6800 and 4810/4800 systems have passive centerplanes, with switch ASICs located on the Repeater boards. The Sun Fire 6800 system uses four Repeater boards to implement a 10x10 data crossbar that connects the CPU/Memory boards and the I/O modules together. The Sun Fire 4810/4800 systems use two Repeater boards to implement a 5x5 data crossbar. The Sun Fire 3800 system uses an active centerplane. The Repeater board functionality is included on the centerplane to implement a 4x4 data crossbar.

The Sun Fire 15K/12K systems use an Expander board to implement a 3x3 switch between a CPU/Memory board, an I/O module, and the centerplane port.

The Sun Fire 15K/12K system has three 18x18 crossbars on the active centerplane to provide connections between the Expander boards. The three crossbars are separate busses for address, response, and data transfers. This method keeps address traffic from interfering with data traffic. The peak Sun Fire 15K system centerplane bandwidth is 43 gigabits per second and 21.6 gigabits per second for the Sun Fire 12K system.

System Configurations

TABLE 1-5 lists the Sun Fire system maximum configurations.

TABLE 1-5 Sun Fire System Maximum Configurations

	Sun Fire 3800 system	Sun Fire 4800 system	Sun Fire 4810 system	Sun Fire 6800 system	Sun Fire 12K system	Sun Fire 15K system
CPU/Memory boards	2	3		6	9	18
Processors	8	12		24	38/52[1]	72/106[1]
Number of DIMMs	64	96		192	288	576
Memory capacity (with 1 Gbyte DIMMs)	64 Gbytes	96 Gbytes		192 Gbytes	288 Gbytes	576 Gbytes
Centerplane	Active	Passive			Active	
Repeater boards	0	2		4	NA	
Expander boards	NA				9	18
Domains	2	2		4	9	18
I/O/MaxCPU Modules	2	2		4	9	18

TABLE 1-5 Sun Fire System Maximum Configurations *(Continued)*

	Sun Fire 3800 system	Sun Fire 4800 system	Sun Fire 4810 system	Sun Fire 6800 system	Sun Fire 12K system	Sun Fire 15K system
PCI card types	hot-swap Compact-PCI	PCI and hot-swap CompactPCI			hot-swap PCI	
PCI slots per assembly	6	8 per PCI, 4 per cPCI			4	
Max total PCI slot	12	16		32	36	72
Bulk power supplies	2	3		6		6
Power requirements	100–120 VAC or 200–240 VAC	200–240 VAC				
System Controller boards	2					
Redundant cooling	Yes					
Redundant AC input	Yes					
Enclosure	Rackmount	Deskside or Rackmount	Rackmount	Sun Fire 6800 cabinet	Sun Fire 12K cabinet	Sun Fire 15K cabinet

1. Maximum CPU count is attained by putting MaxCPU boards in the I/O slots.

System Board Designations and Locations

TABLE 1-6 through TABLE 1-15 list the system board designations and locations.

TABLE 1-6 System Board Abbreviations

Abbreviation	System Board	Sun Fire System
IB[6-9]	I/O Module	6800/4810/4800/3800
FP	Filler Panel	All
IO[0-17]	I/O Board	15K/12K
MaxCPU[0-17]	Dual Processor Board	15K/12K
SB[0-17]	CPU/Memory Board	All
SC[0-1]	System Controller	All
SCPER[0-1]	System Controller Peripheral Board	15K/12K

TABLE 1-7 Sun Fire 6800 System Board Locations

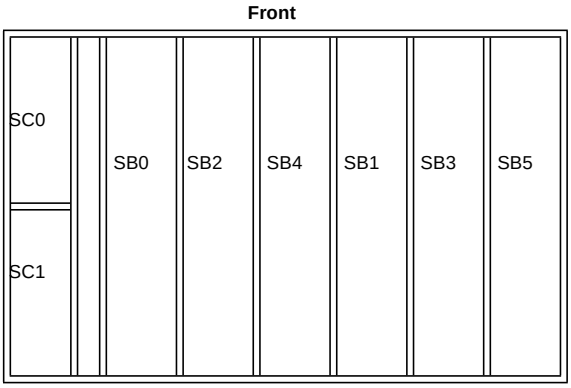


TABLE 1-8 Sun Fire 6800 System Board Locations

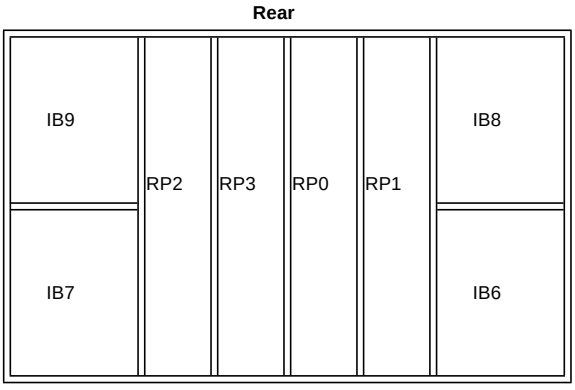


TABLE 1-9 Sun Fire 4810 System Board Locations

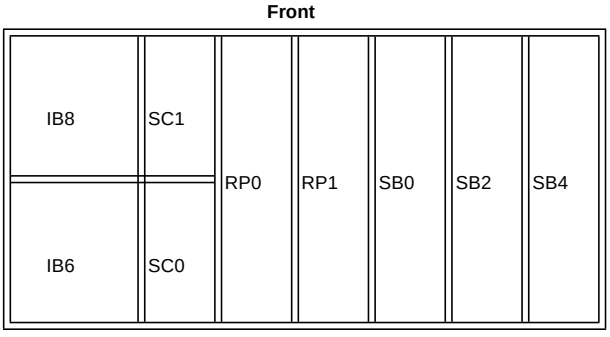


TABLE 1-10 Sun Fire 4800 System Board Locations

Rear

IB8	SC1	RP0	RP1	SB0	SB2	SB4
IB6	SC0					

TABLE 1-11 Sun Fire 3800 System Board Locations

Front

SC1	
SB2	
SC0	
SB0	
IB6	IB8

TABLE 1-12 Sun Fire 15K System Board Locations

Front

SB8	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0	SC0
IO8 / Max CPU	IO7 / Max CPU	IO6 / Max CPU	IO5 / Max CPU	IO4 / Max CPU	IO3 / Max CPU	IO2 / Max CPU	IO1 / Max CPU	IO0 / Max CPU	SCPER0

TABLE 1-13 Sun Fire 15K System Board Locations

Rear

SB17	SB16	SB15	SB14	SB13	SB12	SB11	SB10	SB9	SC1
IO17 / Max CPU	IO16 / Max CPU	IO15 / Max CPU	IO14 / Max CPU	IO13 / Max CPU	IO12 / Max CPU	IO11 / Max CPU	IO10 / Max CPU	IO9 / Max CPU	SCPER1

TABLE 1-14 Sun Fire 12K System Board Locations

Front

SB8	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0	SC0
IO8 / Max CPU	IO7 / Max CPU	IO6 / Max CPU	IO5 / Max CPU	IO4 / Max CPU	IO3 / Max CPU	IO2 / Max CPU	IO1 / Max CPU	IO0 / Max CPU	SCPER0

TABLE 1-15 Sun Fire 12K System Board Locations

Rear

FP	FP	FP	FP	FP	FP	FP	FP	FP	SC1
FP	FP	FP	FP	FP	FP	FP	FP	FP	SCPER1

System Dimensions and Footprints

FIGURE 1-5 shows the dimensions and footprints of the Sun Fire 4810/4800/3800 systems. FIGURE 1-6 shows the dimensions and footprints of the Sun Fire 15K/6800 systems.

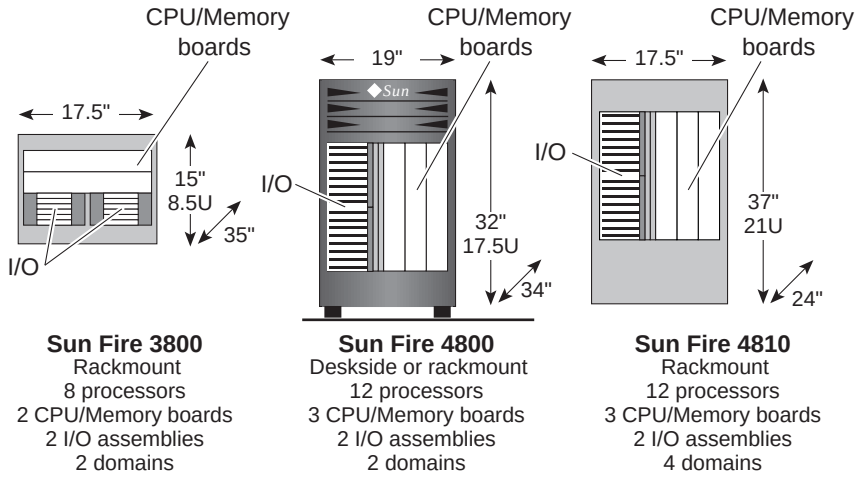


FIGURE 1-5 Sun Fire 4810/4800/3800 Systems Footprints and Dimensions

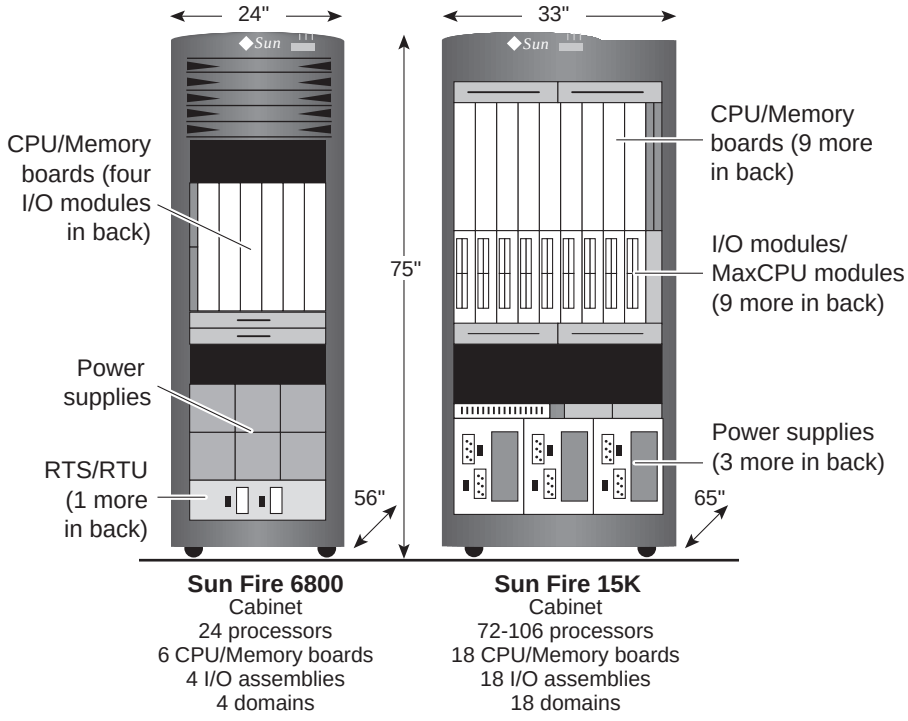


FIGURE 1-6 Sun Fire 15K/6800 Systems Footprints and Dimensions

CHAPTER 2

Determining Your Needs

This chapter considers the different types of questions you should ask when designing a server. It also points out a number of concerns that are often overlooked, but which have a major influence on the design of a server.

The main topics this chapter covers are:

- Creating a New System
- Upgrading an Existing System
- Statement of Requirements

Creating a New System

Usually the most logical place to start is with the simple question:

What is the purpose of this server?

When designing a server, more often than not it is for a specific purpose. That is, you need a new database server, web server, or NFS server, and so on. Rarely do organizations buy servers without a specific need, or merely with the intent of having extra computing power around “just in case.” Even if your organization fits into the latter category, you still should consider what the future use of such a system may be, or it is unlikely to fit into your infrastructure when you decide to begin production.

When thinking about the preceding question, you should be as specific as possible. An answer such as “database server” really does little good. You should also consider the following:

- What type of database?
- Are you going to run it in parallel?
- Are you going to use file system data segments or raw disk?
- How many users do you expect to be using this system concurrently?

- How will the users be connecting to it?
- Will the users be running mostly interactive programs or batch processing jobs?

Note that a system does not necessarily have only one purpose—it may fulfill multiple roles in your organization's infrastructure. It is not uncommon to have a system that is both an NFS server and an intranet web server, for example. Or, perhaps a system is designed to be an end-user system, where users can log in and compile programs, but it must also be shared between several departments with different computing needs. In any case, you should consider each of the proposed roles for the system to determine your organization's requirements.

You should develop a complete specification for your server, with a list of the requirements that it must fulfill. The more critical your server is going to be, the more essential it is that you create an accurate specification. The following sections examine how to develop a detailed specification.

Process Overview

A simple step-by-step approach to designing server specifications is best. This approach involves three steps to create a statement of your requirements:

1. Answering six important questions
2. Looking at often overlooked details
3. Considering potential expansion

After you complete this process, you should have a simple but accurate specification for your server. This specification should consist of your *requirements*, not your proposed design. There should be no mention of CPUs, memory, or redundant power in your answers, but rather descriptions of your needs and the proposed use of the system.

Six Questions to Ask

In practice, the big question, in the preceding questions, is actually composed of six smaller, more specific questions. These are the questions that you should always ask about any system you are designing.

1. What types of applications will be it running?
2. How much data storage is required?
3. How will the system be connected to the network?
4. What type of user traffic do you expect?

5. What are the uptime requirements of the system?

6. How much money can you realistically spend?

While everyone would like to have the fastest, most reliable system made, in reality there are always trade-offs. Since money is a limited resource, often you must make decisions as to whether extra redundancy or better performance is desired. Other factors, such as service contracts and software licenses, should also be considered in the total cost of the system. Thus it is important to carefully answer each of the preceding questions, so you have a clear idea of the requirements for each system you are designing.

To ensure that you have a complete picture of your needs, you should be as specific as possible when answering each of these questions. Taking the first question as an example, the answer:

> A relational database

Is not as useful as:

> An Oracle Database

Though they are the same number of words, the second description dramatically changes the type of system you design. Even better would be:

> Oracle Parallel Database version X.Y. There will be two instances, each requiring a system global area (SGA) of approximately Z. These databases will store all of the organization's employee information, including all payroll records.

By simply expanding the answers to each of the six questions, you will get a comprehensive view of your system requirements. And, as you will see later, your answers to these questions actually map well to the different components of the Sun Fire system, giving you a step-by-step approach to determining how to properly size your system.

Being honest and realistic when answering these questions is important. Do you really need 99.999 percent uptime on a given system? Getting an "extra 9" in reliability may require a substantial increase in the money you have to spend for redundant components and I/O paths. For some systems this is a worthwhile investment, while for others your money may be better spent on extra memory.

So, when designing a system, we recommend that you always answer each of the six questions. In doing so, you will be able to turn the less tangible task of "Design a new NFS server" into a concrete, well-specified set of requirements. This method will also help you to avoid overlooking details that could impact the effectiveness of your server(s).

Often-Overlooked Details

Overlooked details can increase your resource requirements substantially. These include:

- Backups
- System monitoring software
- Multiple network connections

Each of these items can be a substantial hit on the resources of a system, depending on how you use each one. For example, if your monitoring solution consists of a home-grown shell script that pings the machine once an hour, it is probably not going to impact the system. However, it is usually desirable to have a robust monitoring package, especially for production servers. Any monitoring package you get that looks at disk usage, uptime, load, processes, and so forth, can require a good amount of computing power for all of those system calls. The shorter the monitoring interval, the more resources required. You should take this into account when creating your list of requirements and provide additional resources that can be used by the system to support your monitoring package.

Backups are interesting because they almost always tend to catch designers completely by surprise. If you intend to build a server with large amounts of disk, you are probably going to want to back up that data, so you are going to need enough computing and network power for the backups. Otherwise, all of the other programs for which you bought the system will not be able to run while the backups are running your system. Unfortunately, this fact is often missed until the system arrives and the system administrator says, “This system needs to be backed up.”

To avoid such oversights, make a special note if you intend to support backups and take this into account later when you are designing the system.

Planning for Future Expansion

Now that you have considered the first two major topics, you should account for future growth. This often involves some guesswork.

When trying to account for growth, consider issues such as the following:

- What are the future hiring plans of your organization?
- Will this system be serving people outside your organization, such as subscription users? If so, what is the target growth for this user base?
- Can data be retired from the system after a period of time? Or must it remain indefinitely once placed on the system?
- Are there budgeted plans for more system purchases in the future?

Answering these questions should help you make an educated guess at how much future growth you can anticipate. If you do not have enough information to go on, a good rule of thumb is to use a value of 20 percent growth a year.

Once you have a rough idea, take whatever percentage growth rate you anticipate, and multiply it by the answers you came up with to the six questions. So, if you plan on having 10 users now, and anticipate a 25 percent growth rate, this means you will be adding about two to three users a year. If your organization is expanding rapidly, you may be expecting a 50 percent or higher growth rate each quarter.

You should always account for some growth, since, unless your organization is shrinking (in which case you are probably not buying new equipment), growth is inevitable. Also, despite careful planning, you may find your system to be undersized once you put it into production. You definitely do not want to have your brand new server maxed out as soon as it is out of the box.

Upgrading an Existing System

If you have a Sun Fire system with room to expand, upgrading it can make a lot of sense. However, upgrading can be confusing, since it requires a thorough analysis of your current limitations, and careful decisions about what to upgrade.

To simplify the task of upgrading a system, use the following process:

1. First create a new system following the steps outlined previously. This will help you clarify the purpose of the system, as well as its requirements.
2. Identify the shortcomings of the current system from a business standpoint. State these in terms of their impact on the organization, such as “The personnel database is very slow when more than 20 users are accessing it.”
3. Use these two items, along with the analysis methods in Chapter 3 to decide what to upgrade.

Do not make mistakes such as just buying more processors because your system seems slow.

Statement of Requirements

At this point, you should have a good set of answers to the preceding questions. Make sure you did not miss anything major. Because Sun Fire servers are expandable, do not worry if you have some unknowns in your answers, especially in areas such as anticipated growth. You can always add more capacity if you need it, assuming you have a well-designed server. Just make sure you did not miss anything important, like a database instance or huge set of file systems, that could impact the fundamental requirements of the system.

To help formalize this, use your notes to fill in TABLE 2-1, which will then comprise your *statement of requirements*, to which the next chapter refers.

TABLE 2-1 Statement of Requirements Worksheet

Item	Description

CHAPTER 3

Designing Your System

Now that you have completed your statement of requirements you can work on the first half of designing a Sun Fire system—designing the logical server. By the end of this chapter, you will have completed a logical design containing a list of how many of each of the components you need, and a listing of your RAS requirements. Then, you can apply this configuration in Chapter 4 when you choose the physical system in which to place your design.

Systems design is done in this somewhat “backwards” manner for two important reasons:

- To make sure your requirements are clearly stated and met.
- *Multiple servers can be located inside one physical chassis* because Sun Fire systems support domains.

Following this process will also help ensure that you purchase a system with enough room for future expansion.

This chapter covers the following topics, which describe the logical design process:

- Understanding a Running System
- Design Rules of Thumb
- Analyzing an Existing System
- Designing for RAS
- A Logical Design Specification

Understanding a Running System

This section reviews the basics of a computer system. While this is likely all "refresher" material, many people misunderstand the real roles of computer components somewhat. As such, an analogy to a reception desk is used to help

better illustrate the different role each major component plays. In this analogy we follow a receptionist answering various types of incoming calls to show how a computer manages the requests it receives.

Every computer system has three main components that can be configured:

1. I/O devices
2. CPUs
3. Memory

Of course, a Sun Fire system has many other components too, including repeater boards, the Fireplane, and so on. However, in the Sun Fire system (as with most computer systems), these are part of the fundamental architecture of the machine and cannot be configured by the customer. This fact means that to design your system, you should pay close attention to the decisions you make regarding CPUs, memory, and I/O because these decisions will directly affect the effectiveness of your design.

Notice the use of the term CPUs. Because the Sun Fire system board is sold with a minimum of two processors, it is not possible to buy a single-CPU Sun Fire system. All Sun Fires are multiprocessor systems.

I/O Devices

The Sun Fire system uses the PCI bus for all I/O. The I/O is what allows you to do anything productive with the system. Without I/O, you would have no keyboard, no network connection, no disks, and so forth.

Understanding the impact I/O has on the system is important. When something has to be done with I/O, an interrupt is generated. The CPUs must handle this interrupt. Frequently, I/O is the single-biggest resource sink on a system. This fact is especially true when you have multiple types of I/O running heavy loads concurrently, which generates a large number of interrupt requests.

For example, consider a backend database server that is front-ended by a dozen or more concurrent web servers. When a web server needs some dynamic data, it has to make a request via the network to the server, which then must do the appropriate database selects and retrieve the data from its local disk, finally shuffling the reply back across the network to the web server that requested it. This can result in a number of I/O interrupts, as the system must handle all of the network packets as well as all of the disk seeks to get the database information off disk.

When you multiply one request times a dozen or more web servers, each request times a dozen or more clients, you can see that the database server could easily become swamped with I/O interrupts, which excludes the computing power needed to run the operating system, manage memory, and run the database itself.

To tie everything together, think of I/O as each individual phone call received by a receptionist. Each phone call generates an interrupt that the receptionist must handle. Depending on the request, it may result in a lot of data transfer (talking) back to the caller. More calls generate more interrupts. Eventually, the phone system (server) hits a limit either in the amount of concurrent requests that it can handle (memory), the speed with which the requests can be fulfilled (by the CPUs), or how fast the caller and the CPUs can communicate (I/O speed).

CPUs

The CPU is actually responsible for much more than computation. Anything that puts a load on the system, including databases, web servers, email, NFS, NTP, and general network and user traffic, requires a lot of CPU power. The CPU does not do as much *thinking* as it does *handling*. Any time the system must do anything, it must ask the CPU, which has to prioritize the task, schedule it, and allocate resources for it, and do so in a way that allows all the other multitude of things going on to continue running too.

In this way, the CPU can be thought of as a busy receptionist. The receptionist has a number of standard routines. These may include forwarding calls to employees, taking messages, setting up appointments, and even providing direct responses to simple requests, “What is your address?” When an incoming phone call is received, the receptionist executes the proper routine, and completes the request if possible. If the request cannot be fulfilled in a reasonable amount of time, the receptionist may have to place the caller on hold temporarily to handle some other tasks and free up some time.

In some cases, the receptionist may receive a request that is too complex to be handled by standard routines. For example, the receptionist may receive a call that the boss is running late, and that several meetings need to be rescheduled. Here, the receptionist must do some thinking to determine which meetings can be moved to when. At the same time, the receptionist must still pay attention to other incoming calls, to ensure an important request is not missed.

If things get too busy for one receptionist to handle, you may need two or more receptionists. Some callers may even get frustrated and hang up. Even for those that do get through, there will likely not be enough time to properly answer their queries.

So, it is important to consider not only the difficulty of each request, but the volume too. In our analogy, each incoming request requires a certain baseline of time to handle properly. Typically, the receptionist will have to press a button to pick up the appropriate line, answer the call with a greeting, listen and analyze the request, then prioritize it and complete it appropriately. Even if a request consists of nothing more than “Is Mr. Johnson in?”, it still takes a certain amount of time to fulfill the request.

Memory

In the Sun Fire system, the system memory is dynamic random access memory (DRAM). The system uses memory to store things that it is using actively such as the operating system, programs, and their data.

When asked to execute a program, the system must allocate space in memory to hold an image of the program and its associated data. This space can grow or shrink as the program runs, since its resource requirements may change. In reality, most applications grow over time because they do a poor job of cleaning up after themselves.

When a system is under a very heavy load, it may run out of room in memory to hold all the information it needs. In this case, it uses predetermined disk space, known as *swap space*, to temporarily store lesser-used things from memory temporarily to make room for other things. This is known as *paging*, since it involves selectively moving specific data out of memory in sections know as *pages*. When those pages are needed, the system incurs a *page fault*, and the data is moved from disk back into memory.

In extreme situations, the system may undergo *swapping*. In this case, memory images of entire programs are moved from memory out to disk. This is a significant performance hit, and if the system starts swapping, some serious problems may occur. Unfortunately, the terms paging and swapping are often used interchangeably, perhaps because the disk storage is called “swap space,” but they are really very different.

Do not undervalue how important memory is to a running system. Not having enough memory is perhaps the single greatest cause of performance problems.

With the receptionist analogy, you can think of memory as the number of incoming phone lines available. Even if you have five receptionists (CPUs), it will not help the situation if you only have four phone lines (memory). The phone system will still be slow, since you have a bottleneck in the amount of requests you can handle concurrently. To accept another call, the current caller will have to be placed on hold (*page-out*) in order to get back to the first caller (*page-in*).

If the load gets too heavy for the phone system, and no more lines can be put on hold, calls will have to be disconnected (swap-out) to make room for others. The receptionists will then have to call the person back (swap-in), a much more time-intensive process.

Design Rules of Thumb

You can use a number of *rules of thumb* to design a system. Properly using these rules requires a firm grasp on your needs—that you have completed a statement of your requirements using the information and tables in Chapter 1 and Chapter 2.

This section describes the following design rules of thumb:

- Spread your I/O devices across as many PCI buses as possible.
- Decide how many CPUs the system needs.
- Decide how much memory the system needs.
- A well-designed system should seldom page, and never swap.
- The system should always have some idle time.
- Whenever you add additional CPUs, you should also add memory.

I/O Devices

You should always determine your I/O design first, as this along with your application needs determines your computing requirements. To get the best performance and reliability from your Sun Fire server, you should lay out the I/O carefully. An easy rule of thumb is:

Spread your I/O devices across as many PCI buses as possible.

Doing so distributes your I/O load across as many different controllers as possible, thus improving performance. In addition, you are reducing the number of single points of failure that could cause your data to go offline. Unfortunately, this rule of thumb has many caveats. Unlike CPUs and memory, the layout of your I/O intimately affects the reliability of your machine, and whether or not you can use features such as dynamic reconfiguration (DR). Chapter 4 discusses the issue of I/O design in detail, taking all of these factors into consideration.

CPUs

Regardless of what your tasks are—NFS service, CAD simulations, or compiling software builds—handling each request requires a certain baseline of time, as the receptionist example shows. Not only is the type of request important, but the quantity of requests is important too. In fact, it is often harder for a system to handle 100 small requests than 10 large ones, due to the inherent overhead of handling each request.

How many CPUs are enough? The rules of thumb you can use to help you determine how many CPUs you need are:

- One-half CPU per network card
- One-eighth CPU per I/O device (disk or tape)
- Two CPUs per application for mostly I/O-based applications (NFS, web servers and so forth)
- Four+ CPUs per application for mostly CPU-based applications (simulations, databases and so forth).

These figures assume a moderate load on your system. If you are expecting a high load on certain aspects, you should double the corresponding numbers. For example, if a system is going to have a lot of network traffic, you should have one CPU for every network card to handle the interrupts. Conversely, if you are designing a system you expect to have a very light load, cut the numbers in half or consider whether the tasks that system is going to be performing could be combined with another server to lower overhead.

To get an idea of how many CPUs you need, add up each of the criteria that affect you, then round up to the nearest multiple of two. We recommend that you only buy the four-CPU boards for your Sun Fire system. Purchasing a two-CPU board limits your future expansion room, since it takes up the same amount of space as a four-CPU board. However, there are merits to the 2-CPU board if you do not need expansion room, and the examples later in the book demonstrate a good use for it.

So, if you are designing an NFS server with a gigabit network card and six Sun StorEdge T3 arrays, you have the following CPU requirements (TABLE 3-1).

TABLE 3-1 CPU Requirements

Description	Number
Gigabit network card (1)	1/2
Disk arrays (6)	3/4
NFS server	2
Total	3 1/4

Rounding up, you should buy a four-CPU board to run this system.

Note – These rules work well for average systems. However, for high-intensity applications such as online transaction processing (OLTP), data mining, and so forth, you should research your needs more carefully. For details, see “Analyzing an Existing System” on page 38.

When buying CPUs, you should make sure you have enough memory to accommodate them, or else you run the risk of *thrashing*. This means that the system spends all its time moving things around in memory, and never does any real work. This is like the receptionist who spends time picking up phone lines and saying ”Please hold,“ without actually fulfilling any requests. ”Memory” discusses this in detail.

Finally, in terms of speed, getting the fastest processor you can buy is always an advantage. In addition to the speed of the processor, you also should consider the size of its cache. Generally this is decided for you based on the processor model, but you want to make sure to get as large a cache as possible. The cache determines how many operations can be handled at one time by the processor without having to make a trip back out to system memory. Processor cache is several orders of magnitude faster than memory, so *a large cache is always beneficial*.

Memory

Memory is perhaps the single most important part of a computer system, and has the most direct impact on performance. The more memory you have, the more things you can do, and the faster you can do them, since less disk access is needed. There is usually a greater correlation between perceived performance and memory than processors. With relational databases, for instance, being able to fit as much of the database in memory as possible can yield a big improvement in performance.

If your system is running slowly, you should probably buy more memory, not more processors. It is more likely that your system is running out of memory, not processor cycles, and is having to use swap space to run your applications.

It is possible to waste money and overbuy memory as well, though, so here some specific rules. On the Sun Fire system, memory is tied to a processor (see TABLE 1-5 in Chapter 1). So, you cannot buy a board with just memory and no CPUs. This actually simplifies the design process considerably because there are only two decisions to make:

- Whether to half-populate or fully-populate each CPU board
- Whether to buy larger or smaller DIMMs

Fully-populating a CPU board allows you to put more memory on it. In addition, though, you get better interleaving, which increases performance. Thus, the rules of thumb for memory are:

- For I/O-based applications, half-populate the CPU/Memory board.
- For CPU-based applications, fully-populate the CPU/Memory board.

Then, choose the appropriate DIMM size to provide enough memory for your application. Following these rules will naturally lead to smaller memory sizes in NFS servers (where memory is basically solely used for the file buffer cache), and

larger, faster memory configurations in database and compute servers. Most systems tend to be in one category or the other, but if there is a mix, fully-populate all boards.

Remember that, as discussed previously, paging is undesirable. So, another good rule of thumb is:

A well-designed system should seldom page, and never swap.

It is possible, in fact, to run a large memory system with very little (if any) swap space. This fact is somewhat different from other commonly available information. One commonly-used phrase is “Your swap space should be double the size of your physical memory.” Consider this for a moment. You can easily design a Sun Fire system that has 64 gigabytes of memory. If you were to follow this advice, you would have to have 128 gigabytes of swap space. While a few vendors may require you to have a large swap space, you should not rely on swap for real-world memory usage, as it is too slow. When designing a system, make sure that you purchase enough memory so that your system does not swap. If it does, you need more memory.

Analyzing an Existing System

Often, the purpose of designing a new system is to replace an existing system in your infrastructure. If so, you can benefit from analyzing your existing system because this analysis will give you a better idea of what problems you are facing. This analysis is also useful if you are trying to upgrade a Sun Fire server. A proper analysis will ensure that you are upgrading the right parts of the system to address the issues.

Before you go any further, you should revisit your design goals discussed and developed in Chapter 2. Doing so will help you properly formulate your statement of the performance problems you are encountering. A good problem statement is:

When many users are logged in, NFS performance is very slow.

A bad problem statement is:

There is a large number in the `w` column of the `vmstat` output.

Always start with the perceived problems and requirements. An improvement in these areas is the only way you can tell if your design is a success. You can only make use of statistics if you know what you are looking for.

The easiest way to analyze a system is by using the `stat` commands that ship with the Solaris OE, and which can be used to monitor performance of a running system. You can get a full list of the available commands by typing the following command in a shell prompt:

```
# ls /usr/bin/*stat
```

This command will display a series of commands, such as `vmstat`, `iostat`, `netstat`, and so on.

You should never use the `uptime` command to analyze a system. You can use it to show how long your system has been up, but the notion of a *load* is very outdated and fairly useless in the Solaris OE. Most notably, load varies widely from system to system; a load of 10 may indicate a lack of activity on one machine, but extreme activity on another. We recommend you get in the habit of using `vmstat 5` instead of `uptime` when a machine seems sluggish.

Some `stat` commands are more useful than others, so the following sections focus on the useful commands (TABLE 3-2).

TABLE 3-2 Useful `Stat` Commands

Command	Description
`/usr/bin/vmstat`	Virtual memory / paging statistics with CPU / process summaries
`/usr/bin/mpstat`	Extensive per-processor statistics
`/usr/bin/iostat`	I/O and NFS statistics
`/usr/bin/netstat`	Network statistics
`/usr/bin/prstat`	Summary of active processes very similar to the `top` utility

Collecting and understanding the output from these commands should give you a good idea of what problems your current system is having, and how to improve upon these problem areas in the design of your new server.

The following sections review each command in turn, along with how to properly use each one, so you can gather the best statistics possible. It is important to note that not all options of a given `stat` command produce useful—or even trustworthy—output in all situations. The focus is on the specific parts of the output of each command that are the most important.

How and When

How and when you monitor a system is just as important as what commands you use and why you use them to collect statistics. You should make sure that you are monitoring the system when it is doing what you want it to do.

In some situations, this is relatively straightforward, such as on a multiuser interactive system. In this case, you want to run your stats during the day, when everyone is doing their normal work. Conversely, if you have a system that serves mainly as a database server, and the load gets very heavy at night when batch jobs are running, you should gather your stats overnight.

When collecting stats unattended (such as overnight), use a simple shell script that writes to a log file in `/var/tmp` with periodic timestamps. You can use something like the script in CODE EXAMPLE 3-1 to run the `stat` commands mentioned previously:

CODE EXAMPLE 3-1 `nightstats`—Script for Unattended `Stat` Collection

```
#!/bin/sh

# nightstats - Script for unattended stat collection

if [ $# -lt 2 ]; then
    echo "Usage: $0 stat args ..." >&2
    exit 1
fi

# some basic vars holding date, etc.
stat=$1
shift
date=‘date +%Y%m%d‘
logfile=/var/tmp/$stat.$LOGNAME.$date

# run the stat, writing output to our logfile
exec 1>$logfile
echo "Running ’$stat $@’ as ’$LOGNAME’"
while true
do
    date
    $stat "$@"
done
```

The way this script works you will get timestamps at each interval count you specify. So, if you run:

```
# nightstats vmstat 5 12
```

you get a timestamp every 12 repetitions. Note that the `nightstats` script loops indefinitely so you must manually kill it when you want it to stop. You can use this script to kick off stats in the background, either using `cron` or before you go home. For example:

```
# nohup nightstats vmstat 5 300 2>/dev/null &
# nohup nightstats iostat -xcnz 5 300 2>/dev/null &
```

Then, when you come to work the next day, you will have a log file in `/var/tmp` for each stat, with timestamps every five minutes. Each file will be named with the name of the stat command, the date, and your user name (`$LOGNAME` is automatically set to your user name by the shell). This will allow you to collect stats during the times when your system is under the type of load you care about.

Note – You also want to collect some stats when your system is *not* busy, which you can then use as a baseline for comparison. Otherwise, you will not be able to tell what stats change when the load increases.

Simulating Loads

Trying to simulate loads is not very useful. In general, trying to simulate a load gives you a poor—if not misleading—picture of what the system is trying to do. For example, the common practice of using `dd` to write to disks is usually a misrepresentative measure of I/O load. While `dd` reads and writes sequentially, most real-world disk access is random, and is an unpredictable combination of reads and writes. Thus, your configuration could look good on paper, and work well when running `dd`, but work poorly in a real-world application.

To get an accurate picture of your requirements, you should monitor a system that is running what you want it to be running. If you need to simulate this, the best way is to create a test environment that mirrors what you want to design as closely as possible. If you cannot do so, then we recommend that you use the design rules of thumb, and avoid analyzing a dissimilar system as this can cause you to make poor decisions.

What and Why

Now that you understand how and when to measure your system, the following sections examine each of the different `stat` commands and what they tell you.

`prstat` Command

When looking at your stats, the first thing you should know is what your system is doing. Seeing a large amount of disk activity by itself does not tell you anything other than the system is undergoing a large amount of disk activity. This is where the `prstat` command comes in. It shows you what processes are active on the system, along with how much CPU time they are using, what processor they are bound to, their size in memory, their priority, and more. If you have used the freeware tool `top`, the output should look very familiar.

Unlike all of the other `stat` commands, to use `prstat` you just type the command with no arguments:

```
# prstat
```

The display fills the terminal window and refreshes every 5 seconds. You should launch `prstat` in a separate window and keep it going as you use each of the following `stat` commands. That way, you can correlate the performance of your system with what is actively running on it.

`vmstat` Command

Memory is always the first place to start. If you have a memory bottleneck, then all of your other stats are going to be unreliable, since the system will be introducing extra delays trying to manage memory. Often memory problems are misdiagnosed as I/O or CPU problems, since disk access or applications seem slow to the user. In reality, these operations are slow because the system is paging or even swapping.

So remember: *Always start by looking at memory.* Repeat that over and over as a kind of mantra whenever you are analyzing or designing a system.

The virtual memory management algorithms in the Solaris OE are complex. Basically everything is seen as a page of memory, including files. While this is a benefit as far as the system is concerned, it makes analysis more difficult. Therefore, properly analyzing memory takes several steps.

The simplest way to look at memory is by specifying a time interval to the `vmstat` command, and letting it run until you press `Ctrl-C` to interrupt it. The following `vmstat` command monitors the system in five-second intervals:

CODE EXAMPLE 3-2 How to Use the `vmstat` Command

```
# vmstat 5
 procs     memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr s0 -- -- --   in   sy   cs us sy id
 0 0 20 1461688 510080 37 185 30 1 2  0  0  1  0  0  0  667  650  292  4  2 94
 0 0 64 1468888 197976 8 43  0  0  0  0  0  0  0  0  0  638  571  269  1  1 98
 0 0 64 1469320 198528 0  0  1  0  0  0  0  0  0  0  0  642  467  256  0  1 98
```

The first line of the `vmstat` output is a summary.

Note – Always ignore the first line of any `stat` command. It does not provide any useful information because it is a summary for as long as the system has been up. Summaries span too long a period of time, and they give you no indication as to the use of the system during that time.

When looking at the output from `vmstat` (CODE EXAMPLE 3-2), you will notice a lot of columns. You should ignore all the fields about disks and device interrupts, as there are better tools for monitoring these stats, which we will describe in subsequent sections. In fact, only some of these columns (TABLE 3-3) are really useful.

TABLE 3-3 Important `vmstat` Command Output Columns

Column Heading	Meaning
`r`	Number of runnable processes (waiting for CPU time)
`b`	Number of blocked processes (waiting for I/O, paging, and so on)
`w`	Number of runnable but swapped-out processes (normally 0)
`re`	Page reclaims (memory pages taken from other processes)
`mf`	Minor page faults
`pi`	Kilobytes paged in (including process startup and file access)
`po`	Kilobytes paged out (should be close to 0)
`sr`	Pages scanned by page-out scanner (also close to 0)
`us`	Percentage of CPU time spent in user mode
`sy`	Percentage of CPU time spent in system mode
`id`	Percentage of CPU spent idle

First, look at the `procs` headings. Normally, the `r`, `b`, and `w` columns are fairly low numbers, if not 0. This is because, generally, these columns only become nonzero if a process is waiting for something, either a CPU (`r`), I/O (`b`), or enough memory (`w`). Large numbers in these columns are usually bad.

One caveat is that you may occasionally see a steady, unchanging number in the `w` column. This means that the Solaris software has decided these processes have been idle so long they should be swapped out to make room for other things. Do not be concerned about this.

The `cpu` columns give you a good *system-at-a-glance* snapshot of what the system is doing, averaged across all processors. In general, non-idle time should be spent in roughly a 2-to-1 ratio in *`usr`-to-`sys`* modes. Also, if idle time (`id`) is close to zero consistently, you probably need some additional CPUs, especially if the `r` column is a large number. Beyond this, to get a good view of your CPUs you should use the `mpstat` command, as explained in "`mpstat` Command" on page 48.

On to memory. First, note that the `free` column should be completely ignored, as it does not in any way correspond to what is thought of as *free memory*. Because of the way the Solaris software manages memory, the `free` list does not properly count multiple processes sharing the same pages, or unused pages that have yet to be reclaimed. In addition, the file cache grows to consume most of free memory to improve performance.

Consequently, the free list tends to decrease steadily over the uptime of a system, when in fact the system is efficiently reclaiming and reusing memory.

If you want a better picture of available virtual memory, you can use the `swap` command:

```
# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t1d0s0 227,6      16 4093712 4093712
# swap -s
total: 494360k bytes allocated + 35568k reserved = 529928k used,
25137440k available
```

If both the `free` column from the first command, and the `available` column from the second command are nonzero, the system is all right. Beyond that, you can ignore the concept of free memory.

Instead, the most important column of `vmstat` is the scan rate (`sr`). This column shows the number of pages scanned in an attempt to free unused memory. The pageout scanner starts running only when free memory goes below the kernel parameter *`lotsfree`*, which is a small percentage of physical memory. When you see an increase in the scan rate, you should also see a jump in the page-outs (`po`), indicating that pages are being moved from physical memory to swap space. If you

see this consistently, it is evidence of a memory shortage—*the system needs more memory*. If this only happens occasionally, then you should explore whether better job scheduling or `/etc/system` tuning could help. If not, you need more memory.

Note – A high number in the page-ins (`pi`) column is not necessarily significant. This is because when a new process starts, its executable image and data must be read into memory. Also, file system access appears in the pi column too. A large number in the `pi` column is only relevant if the `po` column is large too.

Here is an example of a system that is undergoing heavy paging because it is reading in a large file.

CODE EXAMPLE 3-3 `vmstat 5` Command Output Reading a Large File

```
# vmstat 5
 procs     memory            page            disk          faults      cpu
 r b w   swap   free  re  mf pi po fr de sr f0 s0 s6 s7  in   sy   cs us sy id
 0 0 0  2406032 431280 8  72  2  0  0  0  0  0  1  0  1 121   87  202  1  5 94
 0 0 24 2489472 643792 0  0  1  0  0  0  0  0  0  0  0  328   86  108  0  2 98
 0 0 24 2489472 643784 61 252 483 0 0 0  0  0  5  0  9  466  718  260  3  8 90
 0 0 24 2452936 605616 1396 1753 10950 0 0 0 0 0 9 0 77 1266 2363 801 50 45  5
 0 0 24 2383216 531176 1484 1860 11822 0 0 0 0 0 53 0 40 790 1897 357 55 33 12
 0 0 24 2309576 458256 1435 1773 11475 0 0 0 0 0 69 0 23 697 1791 247 51 30 19
 0 0 24 2236608 391168 1374 1761 11008 0 0 0 0 0 52 0 40 775 1613 235 49 35 17
 0 0 24 2165824 324224 1411 1700 11291 0 0 0 0 0 75 0 16 751 1652 239 47 32 21
 0 0 24 2097680 253816 1378 1720 11012 0 0 0 0 0 0 0 87 746 1687  246 47 33 20
 0 0 24 2028800 184168 1330 2020 10614 0 0 0 0 0 73 0 11 719 1608 239 52 33 16
 0 0 24 1948016 110880 1350 1649 10790 0 0 0 0 0 56 0 37 764 1605 246 49 32 19
 0 0 24 1886176 48208 1282 1666 10187 8 8 0 13 0 1 0 89 793 1934  312 44 37 19
 0 0 24 1835416 7280 688 836 5598 5328 5529 0 6586 0 94 0 47 1088 889 238 24 30 46
 0 0 24 1803768 6680 353 675 3052 6657 6808 0 6749 0 80 0 80 1287 478 435 16 26 58
 0 1 24 1790704 15856 236 393 832 4470 4481 0 1665 0 68 0 107 2579 792 1416 11 41 48
 0 1 24 1784160 18152 35 388 812 3136 3144 0 839 0 60 0 92 1473 724 800 10 29 62
 0 1 24 1777192 18488 29 317 988 2536 2540 0 634 0 66 0 97 988 446 422 7 15 78
 0 1 24 1770768 18664 20 326 942 2334 2345 0 616 0 77 0 77 953 518 409 7 18 75
 procs     memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr f0 s0 s6 s7   in   sy   cs us sy id
 0 0 24 1764704 18528 37 339 820 2636 2648 0 699 0 105 0 48 961 509 343 8 20 71
 0 1 24 1757544 18640 30 264 1051 2206 2214 0 602 0 124 0 43 963 331 398 5 15 80
 0 1 24 1753544 18248 19 255 1081 2048 2056 0 880 0 97 0 70 960 323 412 5 13 81
 0 1 24 1749440 18664 20 258 1046 2048 2062 0 632 0 99 0 63 974 491 443 8 14 77
 0 1 24 1744720 18720 17 255 1009 2152 2153 0 552 0 102 0 58 1012 344 449 6 15 79
 0 1 24 1739992 18920 16 256 1008 1974 1982 0 529 0 101 0 55 929 324 379 6 16 78
 0 1 24 1735416 18800 16 261 998 2048 2052 0 536 0 107 0 55 966 315 379 5 15 80
 0 0 24 1729704 18768 54 268 833 2177 2179 0 546 0 83 0 55 862 352 338 18 13 69
 0 1 24 1728480 18816 105 403 1140 1971 1974 0 552 0 110 0 62 1027 569 492 7 11 83
 0 1 24 1728600 18888 48 196 1118 1484 1489 0 470 0 110 0 53 1014 261 484 5 10 85
 0 1 24 1728496 18832 42 191 1304 1536 1544 0 525 0 123 0 51 1000 160 455 2 8 89
 1 0 24 1728344 37712 372 143 946 1176 1178 0 318 0 103 0 43 789 84 335 3 21 76
 0 0 24 2489048 652144 3 78 32  0  0  0  0  0  6  0  5  427  310  168  1  4 95
 0 0 24 2488840 651872 0  1 11  0  0  0  0  0  1  0  1  351  134  138  0  2 98
 0 0 24 2488792 651776 0  0  3  0  0  0  0  0  0  0  0  349  114  128  0  2 98
```

Notice that for the first half of the output, there is a large number of `pi`, but no `po`, due to the file system activity of reading the file. As it progresses, though, notice the abrupt jump in `po` as well as the `sr`. Also notice how much the `pi` and user time (`us`) drop. The system is spending an inordinate ratio of time managing memory, slowing down how quickly it can read in the file.

As with all stats, brief periods of paging are not important. The purpose of having virtual memory is to allow you to temporarily exceed your available physical memory. You just want to make sure the system is not paging continuously for extended periods of time.

By now you should have a rough idea of what your system is doing. To really understand what is going on, though, you must be able to differentiate between file system pages, executable pages, and so on. To do this you can use the `vmstat -p` option.

`vmstat` Command `-p` Option

Using the `vmstat -p` option fundamentally changes the type of data reported by the command. The `-p` option replaces the columns on processes, CPUs, disks, and interrupts with extended statistics on memory and paging, and displays for executable, anonymous and file system `pi`, `po`, and `pf`.

Examine each of the three types of pages shown by the `vmstat -p` option:

TABLE 3-4 Page Types Shown By `vmstat -p` Option

Page type	Meaning
`executable`	Images of executable programs and their data
`anonymous`	Used for a process heap space, stack, and private pages
`filesystem`	Files mapped into address space through the `mmap command`

Under each page type heading are the following fields, where ? is replaced with the first letter of the page type:

TABLE 3-5 Page Stats Shown By `vmstat -p` Option

Column Heading	Meaning
?`pi`	Kilobytes paged in
?`po`	Kilobytes paged out
?`pf`	Page faults

As with the `vmstat` output, the key field is still `sr`, showing the scan rate. The benefit you get with `-p` is that you can now see what types of pages need the space, allowing you to better understand what the system is doing.

Look again at the system that is reading in a large file, only this time with the `vmstat -p` option.

CODE EXAMPLE 3-4 `vmstat -p 5` Command Output Reading a Large File

```
# vmstat -p 5
     memory           page          executable      anonymous      filesystem
   swap  free  re  mf  fr  de  sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf
 2406040 431296 8  72   0   0   0    0    0    0    0    0    0    2    0    0
 2489992 630792 0   0   0   0   0    0    0    0    0    0    0    0    0    0
 2480080 620344 785 1021 1  0   0    6    0    0   67    0    0 6174    1    1
 2417296 557472 1514 2830 0 0   0    0    0    0    0    0    0 10777   0    0
 2349520 493576 1330 2515 0 0   0    0    0    0    0    0    0 9523    0    0
 2293456 459296 1295 2684 0 0   0    0    0    0    0    0    0 9088    0    0
 2230072 399424 1256 1751 0 0   0    0    0    0    0    0    0 9881    0    0
 2164832 334864 1403 1700 0 0   0    0    0    0    0    0    0 11212   0    0
 2097432 267288 1415 1716 0 0   0    0    0    0    0    0    0 11212   0    0
 2021736 192344 1330 2024 0 0   0    0    0    0    0    0    0 10638   0    0
 1947168 122688 1330 1604 0 0   0    0    0    0    0    0    0 10558   0    0
 1883288 59216  1324 1658 0 0   0    0    0    0    0    0    0 10568   0    0
 1832784 12056 836 863 3936 0 5059   1    0   76    1 3548 3846 6808    4   12
 1798648  8656 207 654 6531 0 6519   0    0   72  353 6374 6446 1502    1   12
 1787016 17864 49 461 4094  0 927    6    0    6  646 4076 4084   12    3    3
     memory           page          executable      anonymous      filesystem
   swap  free  re  mf  fr  de  sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf
 1776040 18448 38 530 3036  0 678    8    0    6  774 3020 3028    3    0    1
 1766800 18488 32 319 2592  0 625    4    0    3  952 2585 2585    1    1    3
 1761080 18696 32 309 2465  0 549    0    0    1  963 2457 2460    1    0    3
 1754696 18600 31 302 2420  0 534    0    0    8  937 2406 2412    1    0    0
 1748608 18640 30 308 2488  0 534    3    0    3  945 2483 2484    1    0    0
 1742504 18784 23 285 2318  0 508    3    0    6  968 2304 2307    3    1    4
 1736960 18784 21 291 2268  0 491    3    0    8  979 2252 2259    3    1    1
 1731008 18584 94 291 2369  0 535    0    0    9  811 2355 2358    3    0    1
 1729800 18744 75 214 1697  0 497    0    0    4 1112 1689 1692    1    0    0
 1729840 18664 57 202 1601  0 538    0    0    4 1156 1587 1595    0    1    1
 1881984 149608 470 122 984 0 366   30    0    6  728  972  976    0    0    1
 2490440 672488 0   0   0   0   0    0    0    0    0    0    0    4    0    0
 2490744 672672 10 168  0   0   0    8    0    0    6    0    0   16    0    0
 2490768 672512 0   2   0   0   0    0    0    0   16    0    0    0    0    0
```

As you can see, this makes what is happening to the system much clearer. The system starts by paging in the file very effectively, until it hits the *lotsfree* limit and the page-out scanner starts. At this point, there is a big jump in the `sr` column. Also notice the abrupt shift from file system page-ins (`fpi`) to anonymous `pi`, `po`, and `pf`. This means that pages are being taken from other processes to make room for the file in memory. Thus, if you see a lot of activity in the `apo` and `sr` columns, you need more memory.

While memory analysis can be complicated if you pay attention solely to the `sr` and `po` columns, you should be able to tell if your system needs additional memory.

mpstat Command

The Sun Fire system is designed to be a multiprocessor system, as evidenced by the fact that you cannot even buy a system with only one CPU. Even though you are looking at CPUs secondarily, being processor-bound is the least likely candidate for bad performance. If anything, you are exploring CPUs secondarily so that you can double-check this assumption, and rule it out as a possible factor. CPUs usually only become a factor in heavily loaded systems that are doing lots of interactive or transactional processing. In most other cases, if you buy enough system boards to hold all your memory, the CPUs that are included are usually sufficient.

As mentioned previously, the `cpu` columns of the `vmstat` output are a good place to start. Generally, a large percentage of idle time indicates that your processing power is sufficient. However, measuring idle time across a lot of processors can mask situations such as one processor getting swamped with interrupts while the rest do nothing. So, it is important to look at your CPUs in detail to make sure you are not missing anything.

Like `vmstat`, just launch `mpstat` with a time interval and let it run:

CODE EXAMPLE 3-5 How to Use the `mpstat` Command

```
# mpstat 5
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0  372   2  836   447  300  393   47   25   26    1   918   23  10   0  67
  1  370   2  622   543  523  301   40   23   35    0   932   24  11   0  65
  2  376   2  527   151  100  396   48   25   26    0   926   24  10   0  66
  3  372   2  531   151  100  397   48   25   26    0   921   23  10   0  67
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0  229   0  546   400  300  458    0   12   13    1   563    2   9   0  89
  1  132   0 2018   585  585  111    0    9   16    0   621    4   8   1  88
  2  265   0  199   100  100  354    1    9   15    0   770   21   9   1  68
  3  363   0  491   101  100  671    1   14   18    0  1339   22  11   0  67
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0  155   0  445   400  300  495    0   12    7    0   398    1   6   0  92
  1   99   0  145   348  347  134    1   10   10    0   487   13   4   0  83
  2  154   0  401   101  100  255    1    8    4    0   723   21   5   0  73
  3  307   0  227   100  100  178    0   11    9    0   989   23   8   1  69
```

This command produces a lot of columns, only some of which you care about:

TABLE 3-6 Important `mpstat` Command Output Columns

Column Heading	Meaning
`xcal`	Interprocessor cross-calls
`intr`	Interrupts
`csw`	Context switches
`icsw`	Involuntary context switches

TABLE 3-6 Important `mpstat` Command Output Columns *(Continued)*

Column Heading	Meaning
`smtx`	Spins on mutex locks
`usr`	Percent user time
`sys`	Percent system time
`wt`	Percent wait time
`idl`	Percent idle time

A cross-call (`xcal`) is a call used by a processor to tell other processors to do something. Cross-calls are used for a variety of things, such as delivering a signal to another processor or ensuring virtual memory consistency. This latter use is very common, as it happens during file system activity. Heavy file system activity (such as NFS) can result in a lot of cross-calls. Also, it is not unusual for the `boot proc` to show thousands of `xcal`s, as it maintains lots of information about the others.

An interrupt (`intr`) is the mechanism that a device uses to signal to the kernel that it needs attention, and some immediate processing is required on its behalf. I/O is the major contributor of interrupts, although there are also “special” interrupts such as the system-wide clock thread that occurs regularly. Interrupts, unlike everything else, are not distributed across all CPUs. Instead, the Solaris OE binds each source of interrupts to a specific CPU.

The term context switch (`csw`) refers to the process of moving a thread on and off a CPU. Context switches are a normal but somewhat expensive occurrence because switching context involves certain overhead, such as populating the stack. Normally, a context switch occurs when a process is done with the CPU and another process is given a chance to run. Thus, a steady number of context switches is insignificant.

Involuntary context switches (`icsw`), on the other hand, are much less favorable. When a process is given access to the CPU, it is has a limited time window in which to run, depending on how many other processes are running, what their priority is, and so on. This is the nature of scheduling. An involuntary context switch means that the process was forcibly stopped by the scheduler before it was finished; the time allotted was too short for the process to finish in, or a higher-priority thread preempted it. A few of these is nothing to be concerned about, but getting a large number of these regularly indicates that the system does not have enough processing power to handle all of the things that need to run. You need additional CPUs.

Finally, a spin on a mutex lock (`smtx`) happens when a thread cannot access a section of the kernel that it needs on the first try. The term mutex is short for a *mutual exclusion* lock, and is used in multithreaded operating systems like the Solaris OE to allow multiple threads to run concurrently in system mode. When a thread enters system mode, it locks the part of the kernel it is using by acquiring the mutex

lock for that section. Once the thread is finished, it releases the lock so other threads can have access. A spin happens when two threads want the same section of the kernel, and one of them has to wait for the other to finish. A few of these are perfectly normal, but a large number means there is contention for the kernel resources.

The difficult thing about CPU performance analysis is that it is nearly impossible to provide hard-and-fast rules of thumb. For example, a high number of interrupts is not necessarily a negative. If you have a lot of I/O, you are going to have a lot of interrupts. Instead, it is the interaction of several statistics that can tell you if you are operating efficiently or having serious problems.

When looking at the `mpstat` command output, you always want to take into account the last four columns—the amount of CPU time spent in the different modes. The only real rule of thumb is:

The system should always have some idle time.

Having consistently low idle time means that your system is getting pushed to its limits in one way or another. Even if your system is properly tuned and running at peak efficiency, if you ran anything else on it you would be out of processing capacity.

However, it is also possible for the system to have a significant amount of idle time, but the system still needs more CPUs. Why? Because the amount of time a CPU spends handling device interrupts cannot be seen by the operating system as device interrupts are higher priority than the clock interrupt. This means the time it takes a CPU to handle network and I/O interrupts will be reported as idle time. The `intr` column only lists disk-based interrupts. (This is different from wait time, which is after an interrupt has been handled and the CPU is waiting for a response from the device.)

Because CPU analysis is so complicated, TABLE 3-7 can help you decipher the different stats. The terms “high” and “low” are used because the exact numbers are very system dependent.

TABLE 3-7 Analyzing the `mpstat` Command Output

If you see this...	It probably means...
High `intr` High `idl` Low `usr`	Your system is very busy handling I/O interrupts.
High `intr` High `sys` Low `usr`	If the system is an NFS server, this is perfectly normal. Otherwise, the system is very busy handling I/O interrupts.
High `intr` High `wt`	I/O requests are taking a long time to fulfill. This is likely an I/O performance problem, not a CPU resource problem. Check `iostat`.

TABLE 3-7 Analyzing the `mpstat` Command Output *(Continued)*

If you see this...	It probably means...
High `smtx` High `sys` or `idl`	Contention for kernel system resources exists.
High `icsw`	Contention for basic CPU resources exists.
High `csw` or `xcal` High `sys` Low `usr`	If this happens consistently, you may require more CPUs, depending on your applications. If you are not noticing any slowness in applications or system problems, however, ignore it.
High `sys` Low `usr` All other stats low	Your system is spending too much time managing resources. Check `vmstat` first.

One nice thing is that the solution to all of these problems is the same. The system needs more and/or faster CPUs. Once again though, the importance of having enough memory is emphasized here. When you add a CPU, you incur additional overhead in the form of more kernel space needed to manage that CPU, and space for that CPU to do its own work. Therefore, the rule of thumb is:

Whenever you add additional CPUs, you should also add memory.

Doing so will help prevent accidental memory shortages, which can actually make your system run *slower* as you add more CPUs.

`iostat` Command

Proper I/O layout is complicated; it is almost never done right the first time. Part of the reason for this is that usage patterns and requirements change over time. Also, where you add memory and CPUs is somewhat predetermined. Where you add disk devices and controller cards, though, has a big impact on the system. Therefore, it is important to make sure that the I/O layout is flexible enough to handle future changes and expansion.

Fortunately, I/O analysis is very straightforward. There is only one version of the `iostat` command to run, `iostat -zxcn`.

CODE EXAMPLE 3-6 How to Use the `iostat` Command

```
# iostat -zxcn 5
<summary omitted>
cpu
 us sy wt id
  0  1  5 93
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
    0.0    0.2    0.0    1.6  0.0  0.0    0.0    8.2   0   0 c0t0d0
    0.0    0.2    0.0    1.6  0.0  0.0    0.0   10.1   0   0 c5t0d0
    0.0   34.6    0.0 2201.0  0.0  1.0    0.0   27.9   0  97 c12t1d0
    0.0    0.2    0.0    1.6  0.0  0.0    0.0   12.2   0   0 c20t122d0
    0.0    0.2    0.0    1.6  0.0  0.0    0.0   14.1   0   0 c20t98d0
    0.0   14.2    0.0  113.6  0.0  0.1    0.0    5.5   0   8 c20t101d0
    0.0    0.2    0.0    0.5  0.0  0.0    0.0   30.1   0   1 c10t1d0
    0.0   58.4    0.0  135.4  0.0  0.3    0.0    5.6   0  30 c2t17d0
    1.0   12.8    8.0  135.0  0.0  0.2    0.0   11.6   0  13 c2t16d0
    0.0    3.4    0.0   19.2  0.0  0.1    0.0   17.9   0   4 c2t9d0
    0.0    0.4    0.0    0.8  0.0  0.0    0.0    5.3   0   0 c2t21d0
    0.0    1.8    0.0    1.4  0.0  0.0    0.0    4.2   0   1 c27t42d0
    0.4    9.2    3.2  155.9  0.0  0.1    0.0   10.9   0   8 c28t69d0
    0.0    9.0    0.0  157.5  0.0  0.1    0.0    9.0   0   6 c28t68d0
    0.0    1.8    0.0    1.4  0.0  0.0    0.0    4.8   0   1 c29t1d0
    0.0    9.0    0.0  157.5  0.0  0.1    0.0    9.5   0   7 c30t35d0
    0.0    0.4    0.0    0.8  0.0  0.0    0.0    5.1   0   0 c30t52d0
    0.0    9.2    0.0  155.9  0.0  0.1    0.0   10.3   0   7 c30t36d0
    0.0   58.4    0.0  135.4  0.0  0.4    0.0    6.5   0  35 c31t66d0
    0.4   12.8    3.2  135.0  0.0  0.2    0.0   12.1   0  12 c31t64d0
    0.0    3.4    0.0   19.2  0.0  0.1    0.0   17.7   0   4 c31t90d0
    0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 tomax:/export/mirrors/
pkg.eng/export/pkg
    0.0    0.2    0.0    0.4  0.0  0.0    0.1    1.0   0   0 twinsun-n1:/export/workspace/
d0/nwiger
```

For this version of the `iostat` command, the output shows extended statistics for only those disk devices with nonzero activity, by physical device path instead of the logical kernel disk name (that is, `c0t0d0` instead of `sd0`). If you are using individual disk partitions, you may also want to use the `-p` option. However, most production environments manage their disks with some type of volume manager package, so in practice this option is not that useful.

As with the other `stat` commands, there are only a few columns you care about (TABLE 3-8).

TABLE 3-8 Important `iostat` Command Columns

Column Heading	Meaning
`kr/s`	Kilobytes read per second
`kw/s`	Kilobytes written per second
`wait`	Number of transactions waiting for service
`wsvc_t`	Average service time in wait queue, in milliseconds
`asvc_t`	Average service time for active transactions, in milliseconds

You can ignore two commonly used columns, `%w` and `%b`, which are supposedly the percentage of time spent waiting and busy, respectively. Because of the complexity of modern disks and controllers, these calculations are very inaccurate. Often the two will total more than 100 percent, which should be impossible. Besides, these columns do not tell you anything that you cannot find out by looking at `wsvc_t` or `asvc_t`.

Analogous to the `mpstat command`, when looking at `iostat` you should always watch the first two columns listed (`kr/s` and `kw/s`) to see how much activity the disks are undergoing. Then, basically, the last three columns should be as close to zero as possible. This indicates that the system has very fast disks, and that the I/O is laid out correctly to avoid controller bottlenecks.[1]

In practice, `asvc_t` will be nonzero for any disks undergoing activity, since it always takes some amount of time for a disk to fulfill a request. As with any stat, you will only be able to tell if the system is particularly busy after establishing a baseline. However, several facts are true:

1. Service times across equally active disks should be fairly even.
2. You should not see huge peaks and valleys under normal conditions.
3. You should rarely, if ever, see a nonzero number in `wait` or `wsvc_t`.

You may, occasionally, see a temporary jump in service times (`asvc_t`) even though there is nothing apparently going on (that is, `kr/s` and `kw/s` are almost 0). This is due to a somewhat strange behavior of `fsflush`, the daemon responsible for flushing disk buffers. Periodically, it will generate a long, random series of writes in a short time period. This results in a queue forming, which bumps up the service time, even though there is no real apparent activity on the disk. If you see this, ignore it.

1. Without the `-n` option, `wsvc_t` and `asvc_t` are combined into a single `svc_t` column.

Now look at some output from a small NFS server undergoing a fairly heavy load from several different concurrent requests to read and write several files:

CODE EXAMPLE 3-7 `iostat` Command Output On an NFS Server

```
# iostat -zxcn 5
     cpu
 us sy wt id
  2 72 13 13
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
    0.2    0.2    1.6    1.6  0.0  0.0    0.0   11.9   0   0 c0t0d0
    0.0    0.2    0.0    1.6  0.0  0.0    0.0   13.0   0   0 c0t8d0
   24.6   42.4  390.9  675.5  0.0  0.9    0.0   13.9   0  34 c4t17d0
   25.6   42.6  409.4  681.3  0.0  0.9    0.0   13.6   0  34 c4t20d0
   25.4   43.6  403.4  694.5  0.0  0.9    0.0   13.0   0  35 c4t22d0
   24.6   43.0  393.5  687.7  0.0  0.9    0.0   13.9   0  34 c4t4d0
   25.0   42.4  397.8  676.1  0.0  1.0    0.0   14.3   0  36 c4t18d0
   24.2   42.8  385.5  684.5  0.0  0.9    0.0   13.4   0  33 c4t16d0
   24.8   43.4  393.9  691.3  0.0  0.9    0.0   13.8   0  34 c4t3d0
   25.2   43.8  403.0  700.5  0.0  1.0    0.0   13.9   0  36 c4t2d0
   25.6   43.4  409.4  694.1  0.0  0.9    0.0   12.9   0  32 c4t21d0
    0.0  132.9    0.0 7936.8  0.0  5.5    0.0   41.5   0  84 c4t6d0
   25.2   43.8  403.0  700.5  0.0  1.0    0.0   13.8   0  34 c4t1d0
   25.2   42.0  403.0  671.7  0.0  0.8    0.0   12.6   0  31 c4t19d0
     cpu
 us sy wt id
  1 63 18 18
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
    0.0    5.8    0.0   44.2  0.0  0.4    0.0   63.3   0   3 c0t0d0
    0.0    5.8    0.0   44.2  0.0  0.3    0.0   58.5   0   3 c0t8d0
   25.6   42.8  407.7  685.0  0.0  0.9    0.0   13.7   0  34 c4t17d0
   25.2   43.8  400.5  698.3  0.0  0.9    0.0   13.4   0  35 c4t20d0
   25.4   43.0  403.7  685.4  0.0  0.9    0.0   13.8   0  34 c4t22d0
   25.2   43.0  403.3  688.2  0.0  1.0    0.0   14.0   0  35 c4t4d0
   25.6   43.6  405.3  693.5  0.0  1.0    0.0   14.3   0  37 c4t18d0
   25.4   42.4  404.9  678.6  0.0  0.9    0.0   13.5   0  35 c4t16d0
   25.2   43.2  398.5  688.7  0.0  1.0    0.0   14.5   0  36 c4t3d0
   25.2   43.6  400.3  694.9  0.0  1.0    0.0   14.0   0  36 c4t2d0
   25.2   43.4  403.3  694.7  0.0  0.9    0.0   12.6   0  32 c4t21d0
    0.0  134.8    0.0 7922.1  0.0  5.6    0.0   41.7   0  86 c4t6d0
   25.2   43.4  400.5  691.9  0.0  1.0    0.0   14.5   0  37 c4t1d0
   25.4   43.2  402.0  687.0  0.0  0.9    0.0   13.2   0  34 c4t19d0
```

If you look at `kr/s` and `kw/s`, the disks on `c4` are moving a lot of data in reads and writes. On this server, these are laid out in a striped / mirrored logical volume mounted as `/export`. From the output, it appears that the volume manager software is doing a good job making sure that the load is spread evenly. The one hot spot is on `c4t6d0`, which happens to be the volume log. This disk is getting hit heavily because all of the transactions must be logged. While much higher than the others, this disk's performance is still well within acceptable numbers because the `asvc_t` is not even over 100. This means that the requests are being fulfilled in a very reasonable amount of time.

Notice that in the second set of output, there is a jump in the `asvc_t` on `c0`, even though there is no real activity (the disks on `c0` are set up as a mirrored root volume). This is likely due to the peculiar activity of `fsflush` mentioned earlier, so you can ignore it. Look at how the `asvc_t` is higher than the disk doing 7922kw/s.

Note that the majority of CPU time was spent in system mode (`sy`), based on the CPU snapshot (provided via the `-c` option). Since this is an NFS server, this activity is nothing for concern. If this was a system for local users doing interactive work, however, you might want to rerun `vmstat` and `mpstat` to make sure there are no memory or processor bottlenecks.

While finding I/O problems is fairly easy, solving I/O problems can be quite difficult. Requirements for I/O vary widely, so a trial-and-error approach is the only reliable way to get good performance from your I/O. Even within a single data center, different servers may undergo vastly different usage patterns and encounter different types of problems.

Do not try to micromanage hot spots on disks. By the time you get everything tuned correctly, the usage patterns will change. This means you will then have a *worse* configuration than you would have otherwise, since it is tuned to match your now-inaccurate requirements. Instead, use your time making sure that:

1. The volumes are spread across as many controllers as possible.
2. You use the proper type of volume for your requirements.
3. You use the proper stripe unit size to match your needs.

These simple steps are often missed, but will solve virtually any disk I/O performance problems. *Just changing the stripe unit size from its default can result in huge performance gains.*

The final caveat on I/O layout is fulfilling the RAS requirements of your system, including making sure it supports dynamic reconfiguration (DR) if you require it. Since these are both major concerns on their own, Chapter 4 discusses this issue and I/O design in detail, taking all of these factors into consideration.

`netstat` Command

Network analysis can be difficult only because the Solaris software does not currently have a solid network utility that really tells you everything you want to know. While you can get a general idea of number of packets, you cannot see things like octets, TCP/UDP throughput rates, or retransmissions. Fortunately, improving the network performance of a system usually amounts to installing an additional network interface card for more bandwidth, even if it is a bit of a “black box” approach.

Despite its limitations, you can tell several things from the `netstat` command output. Unlike the other stats, you must run the `netstat` command separately for each interface you have configured by specifying the `-I` option along with the interface name.

CODE EXAMPLE 3-8 How to Use the `netstat` Command

```
# netstat -I ge0 5
     input   hme0      output           input  (Total)    output
packets errs  packets errs  colls  packets errs  packets errs  colls
909076714 0     837319344 0     0      918674892 0     846917522 0     0
667     0     681     0     0      673     0     687     0     0
426     0     402     0     0      428     0     404     0     0
1886    0     3684    0     0      1886    0     3684    0     0
1878    0     3117    0     0      1882    0     3121    0     0
411     0     391     0     0      411     0     391     0     0
```

You can tell two things from this display:

1. Total number of packets received (input) and transmitted (output) during that interval, both for that interface (left set of columns) and for all interfaces (right set of columns). This is *not* an average per second, but a total count.

2. Number of errors and collisions, which should always be low or zero.

Network capacity is very difficult to gauge with this limited information. Without the sizes of each packet, it is impossible to know if you are anywhere near the throughput limits for the interface you are analyzing. Given this information, if the network seems slow, and you are seeing thousands and thousands of packets each second, try adding another network interface card to see if it helps. If not, you should examine your network as a whole to see if you have more widespread issues.

Many available freeware tools, such as the SE Toolkit and Multi Router Traffic Grapher (MRTG), provide better network analysis than `netstat`. You can use tools such as these to more properly gauge the bandwidth being used by each interface. MRTG is especially useful, as it graphs utilization over time so you can easily see when your network interfaces are getting busy, as well as how much bandwidth they are pushing.

Analysis Reveals...

By this point, you should have a good idea about where the system is weak. Make sure you have good notes, as you need this information in the next chapter when you design your new system.

Giving performance tuning a full treatment is beyond the scope of this book. True performance tuning gets exponentially harder; it is much more difficult to get the last 10 percent out of a system than the first 90 percent. If you are interested in high-

end performance tuning, read *Sun Performance and Tuning—Java and the Internet, 2nd Edition* by Adrian Cockcroft and Richard Pettit (ISBN 0-13-095249-4) and "Application Performance Optimization" by Börje Lindh—Sun Microsystems AB, Sweden Sun BluePrints™ OnLine—March 2002.

Designing for RAS

This is the final step in the design process. By now, you should have a fairly clear understanding of what your requirements are, as well as any possible problems with your existing system. Up until now, this book focused mainly on performance because you should make sure any solution you develop can meet your fundamental application requirements. However, properly designing for RAS is just as important, and requires some thought.

Always keep three principles in mind when designing for RAS:

- The more RAS you want, *the more hardware you must add* to the system.
- RAS is not just a function of the Sun Fire server, but of your *entire site.*
- Maximizing RAS can decrease performance.

The first point is almost always overlooked. As an example, to effectively use DR, you should *add* boards in your design beyond those required for your applications. Why? Because otherwise, when the system dynamically reconfigures a board out of the system, it will not have enough resources to run your applications. The system could start paging, or the CPUs could get too busy handling I/O interrupts to do any real work. *The requirements you have formed up to this point are the minimum you need for your system.*

As for the second point, purchasing redundant power supplies does not benefit you if your site has only a single power grid with no UPS system. RAS is a function of your entire site, not just one server in isolation. As with performance, getting that final 10 percent of reliability out of a site gets exponentially more difficult—and costly. Therefore, you should be realistic about both your requirements and expectations—and your ability to fund them.

Third, taking advantage of certain RAS features and methodologies can decrease the performance of your system. For example, if you mirror file systems, for each write the system must now perform *two* writes, one to each half of the mirror. Some of these effects can be mitigated, for instance by placing the two halves of the mirror on different I/O controllers.[1] However, such performance hits can add up, so it is important to realize it is impossible to maximize both RAS and performance.

1. In fact, many volume managers will "round robin" between the two halves of a mirror on reads, actually increasing your read performance over a single disk.

Uptime Requirements

You were first asked to consider your uptime requirements in Chapter 2, “What are the uptime requirements of the system?” To help answer this question, you can consider the following:

- How much time do you have available for planned maintenance?
- How long can you afford to be offline during an unplanned downtime?

There are two types of downtime—planned and unplanned. Planned downtime includes hardware and software upgrades, whereas unplanned downtime includes system crashes and emergency reboots. All computer systems have some amount of downtime; the goal of a good server design is to minimize the impact this downtime has on your organization.

For some organizations, scheduled maintenance is not an issue; the systems undergo heavy usage during the day from employees, so taking the machine down after-hours is a viable solution. Other organizations, however, serve a worldwide audience and can afford little scheduled maintenance due to time zone differences. Also, it is not uncommon to have a mix of different requirements for different systems at a single site. One thing that every organization has in common, though, is the desire to minimize unplanned downtime as much as possible.

There is no reason to differentiate between the two types of downtime, other than to help you come to a conclusion regarding your overall requirements. When you have a good idea of the uptime required for this system, TABLE 3-9 will help you determine what your design should include to ensure that its RAS properties meet your requirements.

Note – You should always purchase redundant SCs for a system to ensure availability in the event of a System Controller board failure. Without a functioning System Controller board, none of the domains in a system will work.

TABLE 3-9 RAS Design Decision Table

Allowable downtime	Your design should include...
Some	Redundant fan trays Redundant power supplies and transfer switches[1]
Little	Redundant CPU/Memory boards DR for CPU/Memory boards Volume management software (such as Solaris™ Volume Manager (SVM) or VERITAS Volume Manager (VxVM)
Very little	Redundant paths to I/O devices Multipathing software for I/O (such as Multipath I/0 (MPxIO) or VERITAS Dynamic Multipathing (VxDMP) Redundant network connections Multipathing software for networks—such as Internet protocol multipathing (IPMP) DR for I/O devices and networks
Almost none	Multiple instances of fully redundant systems Clustering software (such as Sun™ Cluster 3.0)

1. Remember, redundant power helps only if your site is equipped to supply it.

Note – Even though you can use DR to replace failed components, a critical component failure on a running system (such as a failed CPU) will still cause the system to crash. If you cannot afford this type of downtime, you fit in the *almost none* category, and should use a clustering product to guard against system failures.

For most organizations, the *little* downtime category is a good cost/benefit tradeoff. You will have a system that is resilient to failures and, if properly configured, relatively easy to service. You can use DR to add more CPU/Memory boards for increased capacity, or to replace failed components.

Make a note of what category your system fits into, as well as the additional components you will need. You are going to use this in the next chapter to design your system. You will also use it later in the book during the discussion on configuring the system to integrate with your site.

Finally, some closing words on RAS. It is very important that you do not sacrifice parts of your required configuration for additional RAS features. For example, do not decide to buy less memory so that you can afford additional fan trays. You should ensure that your base requirements are met, or else you will not benefit from additional RAS because your system will have fundamental shortcomings.Disk Redundancy and RAID Basics

To ensure the integrity of the data, some type of disk redundancy should be used on any system with important local data storage. The different schemes for achieving such redundancy are often denoted by their *RAID level*. The term RAID comes from Redundant Array of Inexpensive Disks, and there are numbers from 0 all the way up through 53 denoting different ways of laying out sets of disks.

For most applications, however, only three RAID levels are useful: 0, 1, and 5. Each of these allow you to combine multiple physical disks into a single logical volume. The operating system then sees this volume just like a normal disk, and it can be mounted and used in the regular manner.

RAID 0

RAID 0, commonly called *striping*, provides no additional data safety. Instead, it is designed to increase the speed of file system access. With striping, disks in a volume are interleaved at a certain data interval, called the *stripe unit size*. This means that when reading or writing data, multiple disks are accessed in parallel, decreasing the amount of time it takes to access the data. Striping is very common on any system that needs fast data access, such as database servers.

RAID 1

RAID 1, also referred to as *mirroring*, is just the reverse. It provides full data redundancy, but with some performance costs. In mirroring, twice the number of disks are used for the data that needs to be stored. These disks are then arranged in pairs, and identical data is stored on both disks. On a file system write, two physical writes must be performed, one to each disk of the pair. The advantage is you now have two complete copies of your data.

This means you can lose half of your disks and still continue running without data loss. In a large volume, this is obviously an advantage.

RAID 0+1

RAID 0+1, usually called *striping and mirroring*, is a combination of these two techniques. In a striped / mirrored volume, a set of disks is striped together to form each half. Then, these two halves are mirrored to one another. It is possible to design

a striped / mirrored volume so that the performance is better than the individual disks (due to striping), and that fully half the disks can fail without impacting the volume (due to mirroring). This technique is widely-used in production systems.

RAID 1+0

RAID 1+0 is very similar to RAID 0+1, except the volumes are assembled in the reverse order. Here, pairs of disks are mirrored to one another, and then these mirrored pairs are striped together. Volumes created in this manner are slightly more complicated to manage, but are slightly more reliable because of the ways in which disks typically fail. Generally, vendors decide to implement either RAID 0+1 or RAID 1+0, but not both, so the choice of which to use is often made for you.

RAID 5

Finally, RAID 5 is one of the most economical forms of redundancy. In this scheme, a portion of each disk in a volume is used to hold parity. On a write, data is distributed across all the disks in the volume except one, with the parity being written to the remaining disk. This process is repeated in a "round robin" fashion, so that each write places the parity for that write on a different disk. In the event of a single disk failure, the parity is used to recreate data that was on the failed disk. This allows you to lose a single disk (the most common type of failure) and continue running without interruption. RAID 5 is somewhat slow, though, since it must perform all those additional writes for the parity.

While RAID 5 is not as reliable as RAID 0+1 (striping and mirroring), it can still be a good solution, especially for NFS servers. While you can only lose one disk, it is uncommon to lose a whole enclosure barring human error or a power failure, both of which will probably affect much more than your disks. To make use of RAID 5, you should consider only those enclosures that support hardware RAID, since otherwise it is too slow for many applications.

Once you have selected what type of RAID you wish to use for each of your different volumes, you should adjust your storage purchase accordingly. For example, if you want to mirror a set of data, you must purchase double the amount of disk you calculated above. You will need to make sure to increase your controller cards as well.

With RAID 5, check the enclosure you are considering purchasing to verify that it supports hardware RAID.

A Logical Design Specification

By now you should have available all of the information you need to create a logical design specification:

- Design rules of thumb
- Your existing system analysis (if applicable)
- Your RAS design requirements

As you did in the Statement of Requirements Worksheet in Chapter 2, formalize this information into a design for your logical system that will serve as an accurate picture of your needs, so you can use it in the next chapter to choose the appropriate physical system. List your specifications in TABLE 3-10.

TABLE 3-10 Logical Design Specification Worksheet

Item	Description

TABLE 3-10 Logical Design Specification Worksheet

Item	Description

CHAPTER 4

Choosing a Sun Fire System

At this point, you have completed a logical design worksheet for each server you are creating. It is now time to create a specification for a physical Sun Fire server, which you can use to purchase the appropriate equipment. Basically, this process consists of deciding how to lay out the design of your logical servers in a physical Sun Fire system(s).

If you have never designed a system that supports domains before, this aspect can be somewhat confusing. It is important to remember that all Sun Fire systems support domains, so you can have *multiple servers inside a single physical system*. For example, if you are designing a database server and two web servers, you actually have several alternatives with the Sun Fire product line. For example, you could purchase all of these as separate physical boxes, that is, as a Sun Fire 6800 system (for your database) and two Sun Fire 3800 systems (for your web servers). Or, you could put all of these in a single Sun Fire 15K/12K system by using domains to create the individual servers. You could also use some combination thereof, combining some servers but not others.

This is why the first step was to design each server without regard to its physical location. Now that you have done so, you can look at these design specifications and determine how to lay out these servers physically. Because there are advantages and disadvantages to using domains, deciding to use a domain for a given server is a decision that bears careful consideration.

This chapter covers the following topics:

- Using Domains
- Selecting a Sun Fire Model
- Design Process Summary
- System Design Examples

Using Domains

Chapter 1 discusses domains while highlighting the features of the Sun Fire system product line. Basically, a domain is a completely separate logical server. It runs its own instance of the Solaris software, and can be rebooted or serviced without affecting any of the other domains. You access domains just as you would if they were in their own physical boxes; there is no apparent difference to users or applications. In fact, domains are fully integrated in the Sun Fire product line, so even if you purchase a Sun Fire 3800 system to use as a single server, you still must define a domain and place all of the components in it to use it.

Domains, therefore, are inescapable in Sun Fire systems. So, the question really becomes not *if*, but *how* you should use domains. Specifically, should you use multiple domains in a single box? To help you decide, the next sections of this chapter discuss the advantages and disadvantages of domains.

Advantages of Domains

Using multiple domains within a single physical system has several advantages:

- Creation of test environments to stage your production server upgrades
- Ability to reassign resources from one server to another as your needs change
- Consolidation of multiple servers for ease of administration and monitoring

Test domains are perhaps the most common and useful application of domains. You can stage your new environment in the same chassis as your existing one, then once your testing is complete, reboot from the new domain to “upgrade” your production server. This results in an upgrade that seems extremely quick to end users, saving many hours of downtime that would normally be needed to upgrade the server’s operating system, applications, patches, and so forth.

Using features such as dynamic reconfiguration (DR) to reassign resources between domains is also useful, but usually only if your needs tend to vary widely. For example, you may tend to run a large number of batch jobs over the weekend that require a large increase in the computing power. In this case, using domains would be a good solution, as you could dynamically reconfigure the boards from a lesser-used domain into your production domain over the weekend. Situations like this are good candidates for multidomain systems.

Finally, server consolidation can be a worthwhile goal, but only if you are able to consolidate three or more servers. Administering multiple domains takes some planning and conscious effort; it is not as easy as slapping a sticker on the side of a box and calling it “bob.” Instead, you have to remember that both “bob” and “jim”

are inside the same chassis, which CPU/Memory boards and I/O boats comprise them, what letter domains they are, and so on. Unless you can consolidate several servers in the same chassis, you are probably not breaking even on the added complexity needed to administer multiple domains (assuming server consolidation is your only goal).

Disadvantages of Domains

As with everything in life, there are two sides to the coin. Some of the disadvantages of domains that you should be aware of are:

- Increased administrative complexity
- Certain physical failures can affect multiple domains (such as a bad Fireplane)

These two points may seem somewhat paradoxical, considering not only did we just mention domains were totally isolated from one another, but also that consolidating domains can ease administration! Basically, using domains gives you flexibility in exchange for complexity. This is not an unusual tradeoff with computer systems.

We believe that if you need one of the previously described advantages of domains, a multidomain system is worth considering. If you need two or more of the features listed, you should definitely use domains. Otherwise, buy a separate Sun Fire system for each server you are designing.

Selecting a Sun Fire Model

Now that you have decided how you are going to use domains, you can select your physical Sun Fire system. This process should be repeated for each server that you are designing.

Pull together each of the server specifications for the logical servers you are going to put inside this system using domains. So, if you are going to put your two web servers inside the same system, use those two Logical Design Worksheets (TABLE 3-10). Then, fill in TABLE 4-1 based on those two sheets.

TABLE 4-1 Resource Requirements for Your Sun Fire System

Resource	Total needed
Domains	
CPUs	
Memory	
I/O slots	

Now, compare your answers in this table to TABLE 1-5. Choose the Sun Fire system model that meets these requirements. Make sure that your answers take into account future expansion and RAS requirements. You may want to refer to table TABLE 3-9 for a recap of the RAS categories. If you must use any redundant components, make sure to double the corresponding numbers (that is, if you need redundancy for four CPU boards, you need eight CPU boards).

The only variable not accounted for in this table is the system footprint. Our experience shows that system footprint is really a function of how much computing power you need. If you need lots of computing power, you are going to need more space. You can have a somewhat smaller total footprint if you use domains, but remember to take into account I/O cabinet space too.

For most applications we recommend that you purchase the rack-mounted versions of the Sun Fire systems. This enables you to place at least some of your I/O in the same cabinet, thus saving floor space.

Creating a Final Specification

Now you can pull everything together and create your final specification so you can buy your system(s). Fill out the following table, then use it when contacting your Sun sales advisor to ensure that you purchase a system that meets your needs.

TABLE 4-2 Sun Fire System Specification

Item	Details
Sun Fire Model	
CPUs: Total needed	
CPUs: Number of boards	
Memory: Total needed	
Memory: Half or fully populated?	
I/O: External devices	
I/O: Internal cards	
Redundant SCs?	Yes
Redundant fan trays?	
Redundant power?	

In many cases, configuration restrictions will result in a system that does not exactly match your specification. When consulting with your sales advisor, be sure to meet the *minimum* requirements for your system.

For example, assume that your configuration requires 11 CPUs and 17.5 gigabytes of memory. Buying such a system is not possible. Instead, you will probably have to buy a system with 12 CPUs and at least 24 gigabytes of memory. This is not a problem because your minimum requirements are met. If you make sure that you purchase this configuration as three system boards with four CPUs on each one, you will be in good shape.

Finally, special incentives may be offered occasionally on some systems. Taking advantage of these is often desirable because they can offer a substantial price break. If you do so, make sure to use the specification you developed, so you do not spend money on a solution that does not work for you.

Design Process Summary

Even though this is a rather long process it helps you formalize your needs and better understand how all of the Sun Fire features interact. To summarize, the design process consists of the following four steps:

1. Determine your needs.
    a. Answer the six questions.
    b. Look for any often-overlooked details.
    c. Plan for future expansion.
2. Design each logical server.
    a. Use the design rules of thumb.
    b. Analyze your existing system, if available.
    c. Take into account your RAS requirements.
3. Choose a Sun Fire system.
    a. Decide how to use domains.
    b. Calculate your total system requirements.
    c. Select the appropriate Sun Fire model.
4. Complete the final specification and contact your sales advisor.

System Design Examples

To tie everything together, the following sections provide some concrete examples of system designs. These sections describe each of the steps highlighted previously, and finish with a complete Sun Fire system specification for each one. Subsequent sections of this chapter revisit these same systems using real-world configuration examples. Each of these systems utilize the entire design process to help you understand it more readily.

A Note on I/O

Specific I/O devices were not mentioned up to this point because we wanted to emphasize the basic concepts of designing systems. However, I/O devices do affect your design because their capabilities determine how you can address your network requirements and create the necessary redundancy for your data storage too.

These examples use the following Sun products:

- Sun Quad FastEthernet™ network adapter card
- Sun™ Gigabit Ethernet FC-AL/P Combination Adapter card
- Sun StorEdge™ S1 array (LVD SCSI)
- Sun StorEdge D2 array (LVD SCSI)
- Sun StorEdge T3 array (Fiber FC-AL)
- Sun StorEdge D240 media tray (single-ended SCSI)
- Sun StorEdge 9970 system (Fiber FC-AL)

Sun's storage and network products change periodically, so you should consult your sales advisor to obtain a list of the current I/O products.

Example 1—NFS Server

Assume you must create a new NFS server for a graphic design company of about 50 people. It is going to support home directories and tools, but since the company is graphics-oriented, each person needs a lot of storage space. The server will be replacing an older, much smaller server. Go through each of the steps summarized in the following sections to design the new server.

Determine Your Needs

1. **Answer the six questions.**

   a. What types of applications will it be running?

   NFS service for home directories and tools, including some very large files. Since desktop machines are mainly PCs, the system will run SAMBA to allow NFS mounting on PCs. Finally, the system will also be the NTP server for the site.

   b. How much data storage is required?

   Estimate approximately 1 to 2 terabytes to start. About 150 gigabytes will be used for tools. All of the remaining space will be allocated to home directories. The current server has 500 gigabytes of data and is full.

c. How will the system be connected to the network?

Gigabit Ethernet (fiber short-wave multimode cable).

d. What type of user traffic do you expect?

No local traffic, as users will not be allowed to log in to this system for stability reasons. The vast majority of the usage will be via NFS, with some minimal FTP usage too.

e. What are the uptime requirements of the system?

Extended business hours 5 days a week, from roughly 8:00 AM to 10:00 PM. Downtime can be accommodated if scheduled in advance.

f. How much money can you spend realistically?

The budget is flexible enough for some upgrades, but the system should be sized reasonably.

2. Look for any often overlooked details.

a. Backups

This server will have to back itself up because the old server it is replacing does so currently. The backup policy will include weekly incremental backups and monthly full backups. A copy of the monthly full backups will be made and stored off site. This means you should directly attach some type of tape stacker.

b. System monitoring software

Not needed; standard Solaris OE tools should be sufficient.

c. Multiple network connections

None.

3. Plan for future expansion.

This department typically hires approximately five new people each year and needs approximately 30 to 40 gigabytes of space for each new person, so you should be able to add 150 to 200 gigabytes of space each year, not including normal file size increases. You should not need more CPUs or memory unless disk space grows exorbitantly.

Design Each Logical Server

1. Use the design rules of thumb.

a. I/O

For the amount of data you need, a pair of Sun StorEdge T3 arrays will serve your needs the best. This will give you approximately one terabyte of space, and allow you to mirror the disk cache for redundancy. You should connect each of the Sun StorEdge T3 arrays to a separate controller card for best performance and reliability so you will have two disk arrays and two controller cards.

You should also purchase a boot device. For this, the Sun Storage D240 media tray is a good fit. This device gives you a local tape drive, which might be all that is required, and gives you DVD-ROM access.

b. CPUs

Based on the preceding requirements, TABLE 4-3 lists the CPU requirements.

TABLE 4-3 CPU Requirements

Description	Number
Network card	1/2
I/O controllers (4)	1/2
NFS	2
Backups	1 (light)
NTP	0
Total CPUs	4

This equates to one CPU board with four CPUs. Note that zero CPUs are allocated for NTP because you project the load to be so small. The normal estimate of two CPUs for backups (an I/O technology) is cut in half because the load will be light and off-hours (when backups are running).

c. Memory

The rules say that for I/O-based systems, you should half-populate each of the system boards with memory. This should give the system enough memory for a good NFS buffer file cache as well as backups.

2. **Analyze your existing system.**

   Since this system will replace an existing NFS server, you can analyze it to see if there are any specific shortcomings. You already know that you need more disk space, and that overall performance is slower than you want. Specifically, NFS performance is slow during the middle of the afternoon, when a lot of artists are working on projects concurrently.

   After running your stats, you recognize some trends. During high periods of device interrupts, the CPUs get very busy and have very little, if any, idle time. This is especially bad during backups. (The CPUs also use lots of time in `sys` mode, but as NFS is a kernel-based technology this fact is nothing to be concerned about.)

   The current system has two processors, so you need to ensure that the new system has more computing power than this. The four CPUs calculated from the preceding rules of thumb should be sufficient, especially considering their increased speed.

3. **Take into account your RAS requirements.**

   As stated previously, the uptime requirements for this system are reasonable. After-hours downtime can be accommodated with some advance notice. However, unplanned downtime will result in the company being offline until the system recovers. Therefore, the system fits into the *little* downtime category.

   Since the site does not have multiple power grids, do not purchase the redundant transfer switch (RTS), but do buy the redundant SC, fan trays, and power supplies.

   In addition, you should purchase redundant CPU boards. You have two choices: purchase a second four-CPU board or split the CPUs across two separate boards. The former will prevent a performance hit if the system loses one CPU board, but at a cost premium, so you should split the CPUs across two separate boards. As alluded to earlier, this type of design is a good use of the two-way boards. While you are trading expandability for increased RAS, an NFS server will likely not require CPU upgrades in the future.

   Also, you should create a redundant root device. Since the Sun StorEdge D240 media tray can be split in half, you can connect each of the disks in it to a separate controller card, and then mirror the two disks with VERITAS Volume Manager (VxVM). This means that you should buy one additional SCSI/Ethernet card.

   You also need redundancy for your main NFS file systems, so you decide to take advantage of the hardware RAID5 capabilities of the partner-pair of Sun StorEdge T3 arrays. This choice also allows you to mirror the cache across the partner-pair. A good way of utilizing the Sun StorEdge T3 arrays is to create a single RAID5 volume, which the Solaris OE sees as one huge disk. You can then use the VERITAS Volume Manager (VxVM) to split this up into smaller volumes. Placing all of the disks (both root and data) under VxVM also allows you to manage all of the file systems using one package.

   Remember that the cost of the volume manager software and extra hardware components should be taken into account when considering the total cost of the system.

Choose a Sun Fire System

1. **Decide how to use domains.**

   You are only designing one server, so you can create one large domain in a single system.

2. **Calculate your total system requirements.**

   Filling in the Resource Requirements Worksheet (TABLE 4-1), you list the following system resource requirements (TABLE 4-4).

TABLE 4-4 Sun Fire System Resource Requirements—NFS Server

Resource	Total needed
Domains	1
CPUs	4
Memory	Half-populated CPU/Memory boards
I/O slots	6 (4 disk controllers, a network card, and a tape stacker)

3. **Select the appropriate Sun Fire system model.**

   Referring to TABLE 1-5 at the end of Chapter 1, a Sun Fire 3800 system should fit your needs. Since you are spreading your data storage across separate controllers, and each FC-AL card has two ports, you can add much additional storage to the configuration in the future.

Complete the Final Specification

Now you can fill out the worksheet (TABLE 4-2) for the final specification (TABLE 4-5) so you can contact your Sun sales advisor.

TABLE 4-5 Sun Fire 3800 System Specification

Item	Details
Sun Fire model	Sun Fire 3800 system
CPUs: Total needed	4
CPUs: Number of boards	2 (2 CPUs/ per CPU/Memory board)
Memory: Total needed	8 Gbytes

TABLE 4-5 Sun Fire 3800 System Specification *(Continued)*

Item	Details
Memory: Half- or fully-populated?	Half-populated CPU / Memory board
I / O: External devices	1 Sun StorEdge D240 media tray
I / O: Internal cards	2 SCSI controllers
	1 Gigabit Ethernet network card
Redundant SCs?	Yes
Redundant fan trays?	Yes
Redundant power?	Purchase an additional power supply, but do not get the extra RTS since the site does not have dual power grids.

Example 2—Simulation Server

For this example, assume that you must set up a large simulation server to be used by a university chemistry department. Faculty and graduate students will be running simulations of chemical reactions and modeling different types of molecules for a new line of research. You will also need to build an NFS server on which to store the data. You would prefer to keep all of these in one box for flexibility.

Determine Your Needs

1. **Answer the six questions.**

   a. What types of applications will it be running?

   This server will be running a variety of CAD-style simulation applications, as well as some homegrown modeling programs. At any given time, there will probably be about a dozen people logged in, some of whom will be running simulations. Many of the jobs launched will run multiple days before completing.

   b. How much data storage is required?

   All of the important data will be stored on the existing NFS server and should be upwards of 2 terabytes. In addition, you will need a sizable local scratch area where interim data files can be written on your simulation server. This scratch area should be approximately 200 to 300 gigabytes.

   c. How will the system be connected to the network?

You should use Gigabit Ethernet on both servers just to prevent any possible bottlenecks, since the users will be transferring lots of data.

d. What type of user traffic do you expect?

Users will be logging into the simulation server and doing their work locally. Though primarily a simulation server, users will probably do software compiles and general shell work on the system too. The NFS server will be restricted; only system administrators will be able to login to it.

e. What are the uptime requirements of the system?

The chemistry department does a lot of research activity on the weekends so the system must be generally available seven days a week, from roughly 7:00 a.m. to 7:00 p.m. However, the system must be fairly stable, since most simulations will run multiple days. If they are interrupted, their progress will be lost and they will have to be restarted. Generally, because the two systems will be interrelated, the NFS server and the simulation server have the same uptime requirements.

f. How much money can you spend realistically?

A research grant has given the department approximately $500,000 to spend on this system.

2. Look for any often overlooked details.

a. Backups

The simulation server will not have to be backed up, since all the data will be on the NFS server domain. The local scratch area is designed to be disposable. The NFS server will be backed up by an existing backup system.

b. System monitoring software

A freeware package will be used to monitor the basic health of the system, because it is already in use on other systems.

c. Multiple network connections

None.

d. Plan for future expansion

The simulation server needs space for future CPU/Memory expansion because the computational requirements may increase as the research continues. You will need I/O expansion for the NFS server to hold all of the data sets.

Design Each Logical Server

1. Use the design rules of thumb.

a. I/O

For the simulation server, you should keep the boot device and scratch area separate, so that a failure on the scratch device does not affect the rest of the system. You can use the Sun StorEdge media tray as a boot device, and purchase it with the DVD/CD drive for installation of software.

For the scratch device, you can use the Sun StorEdge D2 array, which can hold sufficient disks to support the scratch area requirements. This means you need enough space for two disk controller cards, as well as a single network card.

For the NFS domain you can use four Sun StorEdge T3 arrays, configured as two partner pairs, to hold your data. For best performance you should connect each array to a separate FC-AL controller, thus requiring a total of four cards. You also need a network card for the domain.

b. CPUs

Based on the preceding requirements, TABLE 4-6 lists the CPU requirements for the simulation server.

TABLE 4-6 CPU Requirements

Description	Number
Network card	1/2
I/O controllers (3)	3/8
Simulations	4
Total CPUs	4 7/8

Note that these CPU requirements are not accurate. They would probably be fine if you wanted to run just a single simulation, but the system must be able to support three or four multiday simulations running concurrently. So you should multiply the middle number by the number of instances. TABLE 4-7 lists the recalculated CPU requirements.

TABLE 4-7 Recalculated CPU Requirements

Description	Number
Network card	1/2
I/O controllers (3)	3/8
Simulations (3)	12
Total CPUs	12 7/8

Rounding off yields 12 CPUs on three CPU/Memory boards.

For your NFS server, you can look back at the previous example for a similarly sized system. Since this server is not going to be doing backups, and will have a lighter load (only a few users), two CPUs should be fine.

c. Memory

Simulations can use a lot of memory, so you should fully populate all of your CPU/Memory boards with large DIMMs.

2. Analyze your existing system.

You can look at a few of the smaller simulation servers in the department for comparison. Though these servers are insufficient for your needs, their configurations can be useful reference points.

Each of the servers has four to six CPUs and two to four gigabytes of memory. During normal activity these systems are fine, but they can really only run one simulation. When two or more simulations begin running, the systems start paging, and the CPUs see a large number of involuntary context switches with little idle time, so there is contention for processing resources.

Since you are planning to be running many multiday simulations simultaneously, the system probably needs three to four times the amount of processing power of the current system. Multiplying the existing CPUs by the increased number of users yields 12 to 24 CPUs and 24 to 40 gigabytes of memory. However, you should take the increased speed of the UltraSPARC III processors into account too. This means the lower end of the range is probably sufficient, and the 14 CPUs specified using the rule of thumb should be a good start.

3. Take into account your RAS requirements.

Scheduling downtime should be relatively easy. Unplanned downtime, while unfortunate, will not cause huge setbacks except in the case of multiday outages. This means your system is in the *some* downtime category. So, you should purchase redundant SCs, fan tray, and power supplies. You do not need any redundancy for your storage or network, beyond what you may configure using the properties of the Sun StorEdge T3 arrays.

Choose a Sun Fire System

1. **Decide how to use domains.**

   You are designing two separate servers and wish to keep them in one system, so you need two domains.

2. **Calculate your total system requirements.**

   Filling in the worksheet (TABLE 4-1), you find the following system resource requirements (TABLE 4-8).

TABLE 4-8 Sun Fire System Resource Requirements—Simulation Server

Resource	Total needed
Domains	2
CPUs	14 (12 for the simulation domain and two for the NFS server)
Memory	24 to 40 Gbytes (estimated)
I/O slots	8 (6 disk controllers and 2 network cards)

3. **Select the appropriate Sun Fire model.**

   Referring to TABLE 1-5, a Sun Fire 6800 system should fit the server requirements. This system will enable you to fit both domains in the same system, and also allow sufficient expansion room.

Complete the Final Specification

Now, you can fill out the worksheet (TABLE 4-2) for the final system specification (TABLE 4-9) to use when contacting your Sun sales advisor.

TABLE 4-9 Sun Fire 6800 System Specification

Item	Details
Sun Fire model	Sun Fire 6800 system
CPUs: Total needed	14
CPUs: Number of boards	4 (4 CPUs per CPU/Memory board for first 3, 2 for last board)
Memory: Total needed	24-to-40 Gbytes (estimated)
Memory: Half- or fully-populated?	Fully-populated CPU/Memory boards

TABLE 4-9 Sun Fire 6800 System Specification *(Continued)*

Item	Details
I/O: External devices	2 Sun StorEdge D240 media trays 1 StorEdge D2 array 4 Sun StorEdge T3 arrays (2 partner pairs) 2 SCSI/Ether net cards 4 FC-AL controllers 2 Gigabit Ethernet network cards
Redundant SCs?	Yes
Redundant fan trays?	Yes
Redundant power?	Yes

Example 3—Database Server

For this example, assume you must purchase a very large database server that is to be used for OLTP at a large online retailer. You also want to include a small test domain so that you can stage upgrades to your production environment. This test domain should have access to the production server's data for testing purposes. The goal for this system is to replace a Sun Enterprise 10000 that is currently serving this purpose.

Note – This example provides a good starting point, but if you are going to buy a server that your organization depends on for key revenue, we recommend that you get some personalized consulting. With such systems, the layout becomes so application dependent that you really need someone to look at your specific needs.

Determine Your Needs

1. **Answer the six questions.**

   a. What types of applications will it be running?

   This server will be running the Oracle9*i* Real Application Cluster database and application server. It must be able to support several thousand users concurrently who are primarily doing queries (about 4:1 read/write ratio). Their applications are written in a mix of C and Java.

Currently, a Sun Enterprise 10000 is filling this role, but the users need more power. The goal in replacing the existing system is twofold—to be able to fit much more of the database into memory, thus reducing latency, and to increase the computational capacity to keep up with demand.

b. How much data storage is required?

After taking the application and database requirements into account, along with the existing data requirements for OLTP, the system needs five to six terabytes of storage to start, and it must have room for much potential growth.

c. How will the system be connected to the network?

This system will be connected to four other separate tiers of machines, each of which require a separate 10/100 Ethernet port. A single quad card should be sufficient.

d. What type of user traffic do you expect?

All activity will come directly through TCP connections to the database. Most requests will be small, but may be accompanied by a huge amount of data in return. For example, small queries against the database may return tens of thousands of rows. This activity could periodically peak as well, with periods of very high activity followed by periods of less activity.

e. What are the uptime requirements of the system?

This is a 7x24 production system. Since this will be the main OLTP server for the entire company, any downtime translates directly into lost revenue.

f. How much money can you spend realistically?

The budget for this system is approximately $5 million.

2. Look for any often overlooked details.

a. Backups

An established backup infrastructure is already in place to back up this system. However, you should doublecheck its capacity to ensure that it can handle the new system.

In addition, you should make a separate private subnet just for backups, so that you can back up the data without disrupting the normal OLTP operations. This means you should purchase an additional network card. This card should be fast, like a Gigabit Ethernet card, to make sure you are getting as much data streaming to backups as possible.

b. System monitoring software

Because of the importance of this server, you should use a package such as Sun™ Management Center (SunMC) to monitor it.

c. Multiple network connections

As your backup analysis showed, you also should purchase a Gigabit Ethernet card for a private network, for a total of two network cards.

d. Plan for future expansion

This system must be capable of large amounts of CPU/Memory and I/O expansion.

Design Each Logical Server

1. **Use the design rules of thumb.**

a. I/O

For this application, you could get 5 to 6 terabytes of storage a number of ways. You could buy six partner-pairs of Sun StorEdge T3 arrays, for example, but 12 enclosures can be difficult to manage. You decide to go with the Sun StorEdge 9970 system, which is easier to manage because it is a self-contained storage system, and has lots of built-in redundancy features.

The Sun StorEdge 9970 system has 16 fiber connections for this amount of data. Remember that you want both domains to be able to access the I/O. For redundancy, you should connect to the Sun StorEdge 9970 in pairs, using two connections for your test domain and 14 for your production server. Each connection requires a separate FC-AL controller.

Also note the Sun StorEdge 9970 system does not connect directly to the server. Instead, its fiber connections are routed through a separate switch, in a storage area network (SAN) configuration. Chapter 6 describes the exact layout when you physically configure this system. For now, purchase an external fiber switch for the SAN.

As always, you want to keep your boot device separate, so for this you can use the Sun StorEdge S1 array with a separate disk controller. You also need three network cards, as mentioned before.

Finally, for your test domain, you should purchase a Sun StorEdge S1 array for the boot device, and a quad Ethernet network device for connectivity. You will also need two FC-AL controllers to connect to the Sun StorEdge 9970 system.

b. CPUs

Based on the preceding requirements, TABLE 4-10 lists the CPU requirements for the production server.

TABLE 4-10 CPU Requirements

Description	Number
Network cards (3)	1 1/2
I/O controllers (10)	2 1/4
Sun Management Center	1
Oracle Database	?
Application Server	?
Total	4 3/4

Rounding up, you can fit four CPUs on one CPU/Memory board. Since you want to have a separate test domain, you should add an additional board to this, for three boards and 10 CPUs so far.

Note the question marks in TABLE 4-10. Recall that the rule of thumb cannot really be used for high-end systems, like the one you are trying to design here (thousands of users, millions of transactions). Instead, you must do the analysis described in the following paragraphs to determine these numbers.

c. Memory

To size this system, you should have a good idea of the SGA size of the database, so you can make sure there is enough memory to accommodate it. Again, you cannot use the rule of thumb in this situation, but you must analyze your current system.

2. Analyze your existing system.

As mentioned before, a Sun Enterprise 10000 server is performing these tasks currently but it is being maxed out due to the extremely high volume of traffic at the site. The current system has 48 CPUs and 48 gigabytes of memory. It has about 4 terabytes of disk storage.

After some examination, you determine that the total database size is about 1 terabyte. You know that the SGA is configured to be 32 gigabytes. Using the `prstat` command, you find that the normal resident size is approximately 44 gigabytes;

basically, all available memory. The system undergoes some paging during high levels of activity. To be able to fit more of the database into memory and eliminate paging, you should start with two to three times the current amount of memory.

As for computational capacity, some analysis from the `mpstat` output shows that the current system is fairly busy even under average loads. Typically there is approximately 15 to 20 percent idle time, with the rest in a good split between user and system time. During high activity, though, the idle time drops down to less than 10 percent, and a big jump in the involuntary context switches (`icsw`) occurs, along with spins on mutexes (`smtx`).

Given that the UltraSPARC III processor is several times faster than the UltraSPARC II processor due to its improved feature set, you can substantially increase the computational capacity with the same number of CPUs. Database applications benefit the most from the new processor, since the cache is much larger and more efficient. To avoid oversizing, you should start with the same number of processors. You can then add boards later if you need more capacity.

In this case, your analysis yielded better results than the rule of thumb. As discussed before, this means you can *throw out the earlier results*. The new total is 52 CPUs—48 for the production domain, and four for the test domain.

3. Take into account your RAS requirements.

Any downtime is a huge impact to the company. It probably fits in the a*lmost none* downtime category, but you cannot afford two of these systems. Thus, the *very little* category seems to be the best match.

This category requires you to modify the design to take into account numerous multipathing options. First, you should purchase all of the redundant hardware components (SC, fan trays and power).

Since you cannot afford to take a performance hit if you dynamically reconfigure out a CPU board, you should purchase an additional CPU board for the system. This brings the total to 56 CPUs—52 for the production server and four for the test domain.

Next, to increase the redundancy of your storage, you should purchase two of the SAN switches used to connect the Sun StorEdge 9970 system, and split the connections so that half of them go to each switch. The Sun StorEdge 9970 has numerous built-in redundancy features, so you should use those features to achieve the RAS for the data storage. Note that you do not need any additional FC-AL controllers for this because it is a SAN configuration. To mirror the root device, you should purchase an additional Sun StorEdge S1 array and an additional SCSI controller. You can use VxVM to create a mirrored root volume.

Finally, you should also enable network multipathing using Internet protocol multipathing (IPMP). To do so, you should buy twice as many network cards. Do not worry about the private backup network, since this can go down without affecting the production operations. But, you should buy an additional card for the public network, bringing the total to two quad Ethernet network cards for the production server.

Choose a Sun Fire System

1. **Decide how to use domains.**

   As decided earlier, the system will have two domains, one for the production server and one for the test bed.

2. **Calculate your total system requirements.**

   Filling in the worksheet (TABLE 4-1), you find the following system resource requirements (TABLE 4-11).

TABLE 4-11 Sun Fire System Resource Requirements—Database Server

Resource	Total needed
Domains	2
CPUs	56
Memory	Fully populated CPU boards
I/O slots	23 (19 disk controllers plus 4 network cards)

3. **Select the appropriate Sun Fire model.**

   Referring to TABLE 1-5 at the end of Chapter 1, the only option is a Sun Fire 15K system. This will result in a 13-board system with plenty of room for future expansion.

Complete the Final Specification

Now, you can fill out the worksheet (TABLE 4-2) for the final system specification (TABLE 4-9) to use when contacting your Sun sales advisor. Remember, for a high-end system such as this, we recommend you first doublecheck your configuration with someone familiar with your specific application needs.

TABLE 4-12 Sun Fire 15K System Specification

Item	Details
Sun Fire model	Sun Fire 15K system
CPUs: Total needed	56
CPUs: Number of boards	14 (4 CPUs per CPU/Memory board)
Memory: Total needed	88 to 132+ Gbytes (estimated)
Memory: Half- or fully-populated?	Fully-populated CPU/Memory boards

TABLE 4-12 Sun Fire 15K System Specification *(Continued)*

Item	Details
I/O: External devices	3 StorEdge S1 arrays 1 Sun StorEdge 9970 system
I/O: Internal cards	2 LVD controllers 16 FC-AL controllers 2 external SAN switches 3 quad Ethernet network cards 1 Gigabit Ethernet network card
Redundant SCs?	Yes
Redundant fan trays?	Yes
Redundant power?	Yes

CHAPTER 5

System Controller

After you have designed and purchased your Sun Fire system, the next step is to configure it. Since the Sun Fire system is managed using the SC, this chapter examines its role first. The system controller (SC) is responsible for configuration, control and monitoring of the platform. It does the following:

- Configures, tests, and boots domains
- Supplies all system clocks
- Monitors the hardware for power and temperature status
- Provides console access and the user interface for administration and maintenance tasks

This chapter covers the following topics:

- Platform SC Differences
- Accessing Domain Consoles
- Environmental Monitoring

Platform SC Differences

Although the SC provides similar functionality for all the Sun Fire platforms, the implementation between the Sun Fire 6800/4810/4800/3800 systems and the Sun Fire 15K/12K systems is different. Due to the size and complexity of the Sun Fire 15K/12K systems, the SC is more complex.

Sun Fire 6800/4810/4800/3800 Systems

The SC for the midrange servers consists of a single PC board that contains a microSPARC®-IIep processor, memory and PROMS that contain the UNIX-based real time operating system (RTOS), the platform administration application (ScApp), and the diagnostics for all of the platform components. The operating system is not

accessible to the administrator. Updates to the operating system, ScApp or the diagnostics are accomplished with a flash update utility. When connecting to the SC, a menu is displayed. You can choose between a platform level shell or a domain level shell. Different management tasks are performed in the platform and domain shells, which are explained in subsequent sections of this chapter.

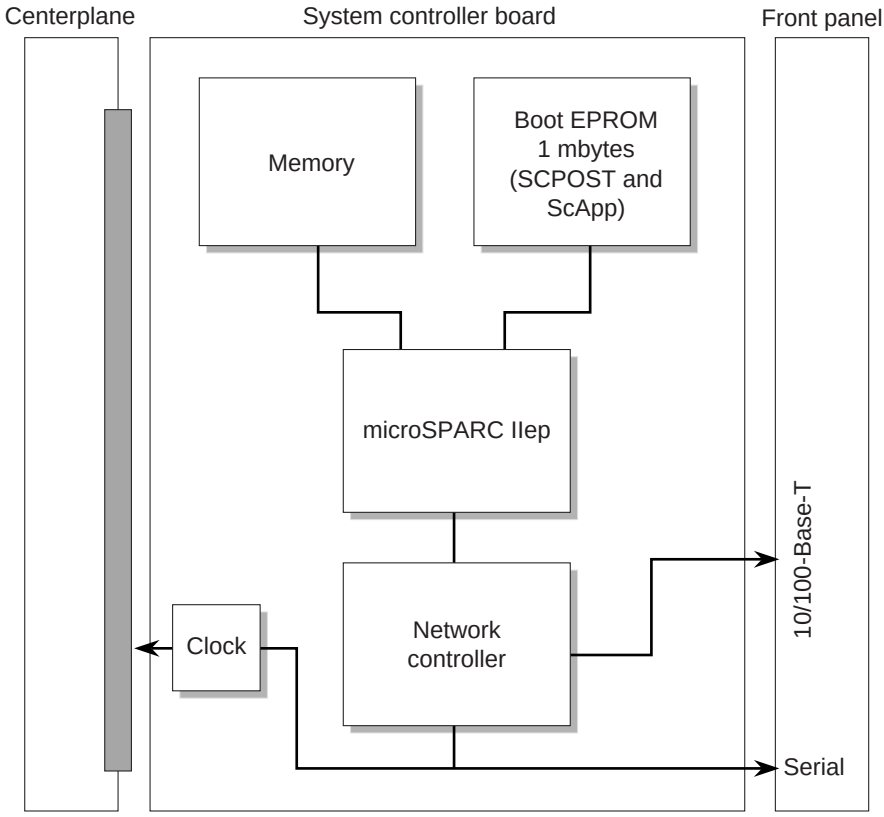


FIGURE 5-1 Sun Fire 6800/4810/4800/3800 SC Interconnect Diagram

Sun Fire 15K/12K Systems

The SC for the high-end systems is a board set that consists of a centerplane support board, an SC board and an SC peripheral board. This SC board set runs the Solaris OE on a microSPARC-IIi processor. The Solaris OE is contained on an internal disk drive that is located on the SC peripheral board. This board also contains a second disk drive, a DVD ROM drive and a 4mm DAT tape drive. The platform administration application, system management services (SMS), runs in the Solaris OE. Access to the SC is by logging in to a normal UNIX® shell.

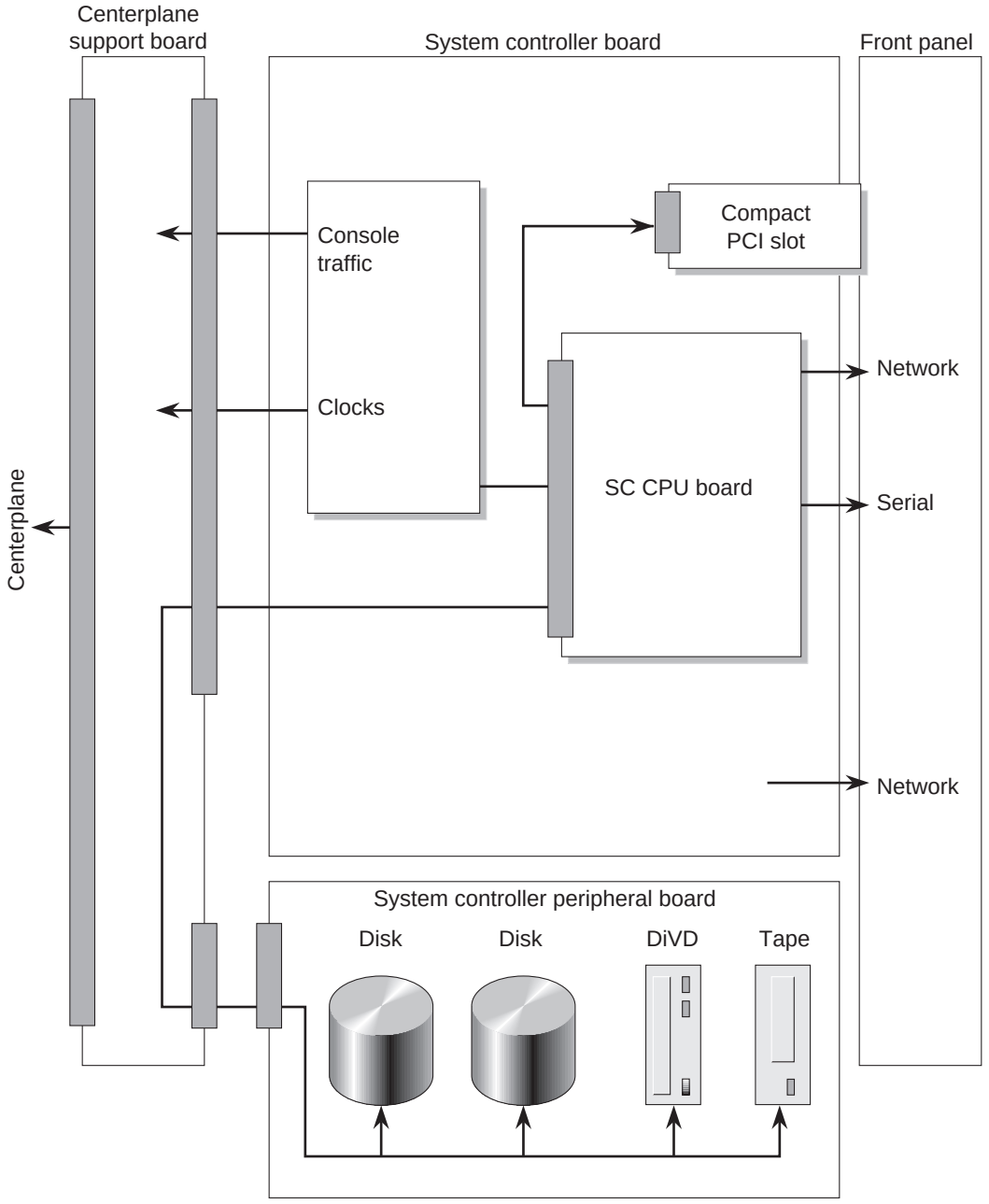


FIGURE 5-2 Sun Fire 15K/12K SC Block Diagram

Accessing Domain Consoles

Domain console access for the midrange Sun Fire systems is different than the high-end systems as the following paragraphs describe.

Sun Fire 6800/4810/4800/3800 Console

For the midrange systems, one console connection is available for each domain in addition to the console connection to the SC. To gain access to a domain's console, you must first connect to the SC by serial or network connection. No operating system is accessible. When the connection is made, the ScApp login menu is displayed.

CODE EXAMPLE 5-1 Console Login Menu

```
[cs1:admin] telnet 3800-sc0
Trying 192.38.102.97...
Connected to 3800-sc0.
Escape character is '^]'.


System Controller '3800-sc0':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console

    Input: 2
Enter Password: password
Connected to Domain B

Domain Shell for Domain B

3800-sc0:B>
```

Note – You can also connect to a domain console from the platform console using the `console` command.

All console connections can be password protected. When connected to the domain console, only commands that can affect that domain are available. For example, the command `showboards` will display only system boards assigned to that domain. Similarly the `poweroff` command can control only the assigned boards.

CODE EXAMPLE 5-2 Platform Level Commands

```
3800-sc0:SC> showboards

Slot     Pwr Component Type                State      Status     Domain
----     --- --------------                -----      ------     ------
SSC0     On  System Controller             -          Passed     -
SSC1     On  System Controller             -          -          -
ID0      On  Sun Fire 3800 Centerplane     -          OK         -
PS0      On  A145 Power Supply             -          OK         -
PS1      On  A145 Power Supply             -          OK         -
PS2      On  A145 Power Supply             -          OK         -
FT0      On  Fan Tray                      Low Speed  OK         -
FT1      On  Fan Tray                      Low Speed  OK         -
FT2      On  Fan Tray                      Low Speed  OK         -
FT3      On  Fan Tray                      Low Speed  OK         -
RP0      On  Repeater Board (F3800)        -          OK         -
RP2      On  Repeater Board (F3800)        -          OK         -
/N0/SB0  On  CPU Board                     Active     Passed     A
/N0/SB2  On  CPU Board                     Active     Passed     B
/N0/IB6  On  CPCI I/O board (F3800)        Active     Passed     A
/N0/IB8  On  CPCI I/O board (F3800)        Active     Passed     B


3800-sc0:SC> poweroff all
/N0/SB0: powered off
/N0/SB2: powered off
/N0/IB6: powered off
/N0/IB8: powered off
RP0: powered off
RP2: powered off
PS0: powered off
PS1: powered off
PS2: powered off
```

CODE EXAMPLE 5-3 Domain Level Commands

```
3800-sc0:A> showboards

Slot     Pwr Component Type                State      Status     Domain
----     --- --------------                -----      ------     ------
/N0/SB0  On  CPU Board                     Active     Passed     A
/N0/IB6  On  CPCI I/O board (F3800)        Active     Passed     A


3800-sc0:A> poweroff all
/N0/SB0: powered off
/N0/IB6: powered off
```

Note – The prompt indicates platform or domain command levels.

Sun Fire 15K/12K Console

To connect to a domain's console, you must first log into the SC. Access can be by serial or network connection. Multiple console connections can be made to the same domain. Only one console has write permission; the additional connections are read only. You can also exclusively use the `console` command to connect to a domain.

CODE EXAMPLE 5-4 `console` Command

```
15K-sc0:someuser:6> console -d A
Trying to connect...
Connected to Domain Server.
Your console is in exclusive mode now.

SunOS 5.8

Console login:
```

By default the console connection is made in the exclusive *locked write* mode. This starts a console session with write privileges and forcibly detaches all other console sessions for the domain. It also will not allow new console sessions. Other options are available to obtain *read only* console connection or an *unlocked write* connection. This connection allows additional *read only* connections to connect to the domain. This connection is not as secure as the *locked write* mode as other users with domain administration privileges have the ability to "grab" the *unlocked write* from you.

Platform Administration

Administration of platforms and domains is implemented differently as well.

Sun Fire 6800/4800/3800 Platform Security

On the midframe systems security is handled by passwords. A different password can be set for the platform and each domain.

Sun Fire 15K/12K Platform Security

Instead of platform and domain shells, there are several levels of platform and domain administrative privileges. This method allows great flexibility for a platform that contains many domains and may have multiple administrators and operators

for each one. The SMS controls administrative privileges through the use of UNIX groups. Normal user accounts can be assigned different levels of privileges by being members of the different administration groups. Privileges can be assigned on a per-domain basis or by a user-role basis. For example, operators can have a different administrative level than platform or domain administrators. An account that belongs to all groups has privileges over the entire platform.

TABLE 5-1 Administrative Groups

Group Name	Description
platadmn	Platform Administrator Group
platoper	Platform Operator Group
platsvc	Platform Service Group
dmn[A-R]adm	Domain Administration Group (one for each domain)
dmn[A-R]rcfg	Domain Configuration Group (one for each domain)

Platform Configuration

Platform configuration tasks performed from the SC include:

- Defining partitions and domains
- Setting the date and time for the platform and for the domains. Since the domains are completely isolated from each other, they have the ability to run at different dates or time zones.
- Setting security policies for the platform and domains. This allows different groups to share the same platform without affecting other domains.
- Setting up SNMP or `syslog` policies. The `loghost` can be set for the SC and for each domain.
- Assigning boot time diagnostic levels for the SC and for each domain.

Examples of each of these tasks are provided in Chapter 6.

Defining Domains

Domains are preconfigured for the Sun Fire system product line. Each domain has an assigned Ethernet address. For the Sun Fire 6800/4810/4800/3800 systems, two to four domains are available and for the Sun Fire 15K and 12K systems there are 18 and 9 domains, respectively.

Domains are defined by adding boards to a domain. First the desired system boards must be included in the available component list (ACL) for the domain. You do this with the `setupplatform` command.

CODE EXAMPLE 5-5 Adding Boards to the Domain A ACL With the `setupplatform` Command

```
3800-sc0:SC> setupplatform
.
.
ACLs
----
ACL for domain A [ ]: SB0 SB2 IB6 IB8
ACL for domain B [ ]:
ACL for domain C [ ]:
ACL for domain D [ ]:
.
.
.
```

The next step is to assign the system boards to the domain with the `addboard` command.

CODE EXAMPLE 5-6 Assigning Boards to Domain A With the `addboard` Command

```
3800-sc0:SC> addboard -d A SB0 SB2 IB6 IB8
```

Note – For the Sun Fire 15K/12K systems, you can also use the `addboard` command to assign and configure system boards into a running system dynamically.

After you define the domain, you can set the date, time, and security policies for console access to the domain.

Environmental Monitoring

The SC monitors the environmental conditions for the platform and domains. It monitors power, cooling, component status, and temperature.

If the SC detects a problem, it is logged and reported through SNMP or by the `syslog` program. The SC also updates the appropriate Status LEDs. If power or temperature exceeds safe limits, the SC shuts down the affected domains.

Platform Status Commands

The platform status commands differ slightly between the Sun Fire 6800/4810/4800/3800 systems and the Sun Fire 15K/12K systems, but in most cases similar information is available for both platforms.

TABLE 5-2 Platform Status Commands

Command	Description
`showboards`	Displays board state, diagnostic results and domain assignments
`showcomponent`	Displays status and POST results for CPUs, DRAMs and I/O Cards
`showenvironment`	Displays voltages, current and temperature of system components
`showplatform`	Displays the ACL for each domain and cabinet information

The following platform `status` command examples are from a Sun Fire 3800 system.

CODE EXAMPLE 5-7 `showboards` Command

```
3800-sc0:SC> showboards

Slot     Pwr Component Type                State      Status     Domain
----     --- --------------                -----      ------     ------
SSC0     On  System Controller             -          Passed     -
SSC1     On  System Controller             -          -          -
ID0      On  Sun Fire 3800 Centerplane     -          OK         -
PS0      On  A145 Power Supply             -          OK         -
PS1      On  A145 Power Supply             -          OK         -
PS2      On  A145 Power Supply             -          OK         -
FT0      On  Fan Tray                      Low Speed  OK         -
FT1      On  Fan Tray                      Low Speed  OK         -
FT2      On  Fan Tray                      Low Speed  OK         -
FT3      On  Fan Tray                      Low Speed  OK         -
RP0      On  Repeater Board (F3800)        -          OK         -
RP2      On  Repeater Board (F3800)        -          OK         -
/N0/SB0  On  CPU Board                     Active     Passed     A
/N0/IB6  On  CPCI I/O board (F3800)        Active     Passed     A
/N0/IB8  On  CPCI I/O board (F3800)        Active     Passed     A
```

CODE EXAMPLE 5-8 `showcomponent` Command

```
3800-sc0:SC> showcomponent

Component           Status   Pending  POST   Description
---------           ------   -------  ----   -----------
/N0/SB0/P0          enabled  -        pass   UltraSPARC-III, 750MHz, 8M ECache
/N0/SB0/P1          enabled  -        pass   UltraSPARC-III, 750MHz, 8M ECache
/N0/SB0/P2          enabled  -        pass   UltraSPARC-III, 750MHz, 8M ECache
```

CODE EXAMPLE 5-8 showcomponent Command *(Continued)*

```
3800-sc0:SC> showcomponent

/N0/SB0/P3          enabled  -        pass   UltraSPARC-III, 750MHz, 8M ECache
/N0/SB0/P0/B0/L0    enabled  -        pass   512M DRAM
/N0/SB0/P0/B0/L2    enabled  -        pass   512M DRAM
/N0/SB0/P0/B1/L1    enabled  -        untest empty
/N0/SB0/P0/B1/L3    enabled  -        untest empty
/N0/SB0/P1/B0/L0    enabled  -        pass   512M DRAM
/N0/SB0/P1/B0/L2    enabled  -        pass   512M DRAM
/N0/SB0/P1/B1/L1    enabled  -        untest empty
/N0/SB0/P1/B1/L3    enabled  -        untest empty
/N0/SB0/P2/B0/L0    enabled  -        pass   512M DRAM
/N0/SB0/P2/B0/L2    enabled  -        pass   512M DRAM
/N0/SB0/P2/B1/L1    enabled  -        untest empty
/N0/SB0/P2/B1/L3    enabled  -        untest empty
/N0/SB0/P3/B0/L0    enabled  -        pass   512M DRAM
/N0/SB0/P3/B0/L2    enabled  -        pass   512M DRAM
/N0/SB0/P3/B1/L1    enabled  -        untest empty
/N0/SB0/P3/B1/L3    enabled  -        untest empty
/N0/IB6/P0          enabled  -        pass   IO Controller 0
/N0/IB6/P1          enabled  -        pass   IO Controller 1
/N0/IB6/P0/B0       enabled  -        untest 66/33MHz. CPCI Bus
/N0/IB6/P0/B1       enabled  -        untest 33MHz. CPCI Bus
/N0/IB6/P1/B0       enabled  -        untest 66/33MHz. CPCI Bus
/N0/IB6/P1/B1       enabled  -        untest 33MHz. CPCI Bus
/N0/IB6/P0/B0/C0    enabled  -        untest 66/33MHz. 3.3V 3U CPCI card
/N0/IB6/P1/B0/C1    enabled  -        untest 66/33MHz. 3.3V 3U CPCI card
/N0/IB6/P0/B1/C2    enabled  -        untest 33MHz. 5V 3U CPCI card
/N0/IB6/P0/B1/C3    enabled  -        untest 33MHz. 5V 3U CPCI card
/N0/IB6/P1/B1/C4    enabled  -        untest 33MHz. 5V 3U CPCI card
/N0/IB6/P1/B1/C5    enabled  -        untest 33MHz. 5V 3U CPCI card
/N0/IB8/P0          enabled  -        pass   IO Controller 0
/N0/IB8/P1          enabled  -        pass   IO Controller 1
/N0/IB8/P0/B0       enabled  -        untest 66/33MHz. CPCI Bus
/N0/IB8/P0/B1       enabled  -        untest 33MHz. CPCI Bus
/N0/IB8/P1/B0       enabled  -        untest 66/33MHz. CPCI Bus
/N0/IB8/P1/B1       enabled  -        untest 33MHz. CPCI Bus
/N0/IB8/P0/B0/C0    enabled  -        untest 66/33MHz. 3.3V 3U CPCI card
/N0/IB8/P1/B0/C1    enabled  -        untest 66/33MHz. 3.3V 3U CPCI card
/N0/IB8/P0/B1/C2    enabled  -        untest 33MHz. 5V 3U CPCI card
/N0/IB8/P0/B1/C3    enabled  -        untest 33MHz. 5V 3U CPCI card
/N0/IB8/P1/B1/C4    enabled  -        untest 33MHz. 5V 3U CPCI card
/N0/IB8/P1/B1/C5    enabled  -        untest 33MHz. 5V 3U CPCI card
```

CODE EXAMPLE 5-9 showenvironment Command

```
3800-sc0:SC> showenvironment

Slot Device    Sensor    Value  Units     Age     Status
---- --------- --------- ------ --------- ------- ------
SSC0 SBBC 0    Temp. 0    49    Degrees C   6 sec OK
SSC0 CBH 0     Temp. 0    52    Degrees C   6 sec OK
SSC0 Board 0   Temp. 0    37    Degrees C   6 sec OK
SSC0 Board 0   Temp. 1    37    Degrees C   6 sec OK
SSC0 Board 0   Temp. 2    37    Degrees C   6 sec OK
SSC0 Board 0   1.5 VDC 0   1.49 Volts DC    6 sec OK
SSC0 Board 0   3.3 VDC 0   3.33 Volts DC    6 sec OK
SSC0 Board 0   5 VDC 0     4.98 Volts DC    6 sec OK
PS0  48 VDC 0  Current 0   4.67 Amps        5 sec OK
PS0  48 VDC 0  Temp. 0    34    Degrees C   6 sec OK
PS0  48 VDC 1  Current 0   0.36 Amps        6 sec OK
PS0  48 VDC 1  48 VDC 0   55.02 Volts DC    6 sec OK
PS0  48 VDC 0  48 VDC 0   56.33 Volts DC    6 sec OK
PS1  48 VDC 0  Current 0   4.63 Amps        5 sec OK
PS1  48 VDC 0  Temp. 0    33    Degrees C   5 sec OK
PS1  48 VDC 1  Current 0   0.36 Amps        5 sec OK
PS1  48 VDC 1  48 VDC 0   54.76 Volts DC    5 sec OK
PS1  48 VDC 0  48 VDC 0   56.07 Volts DC    5 sec OK
FT0  Fan 0     Cooling 0    Low             5 sec OK
FT1  Fan 0     Cooling 0    Low             4 sec OK
FT2  Fan 0     Cooling 0    Low             4 sec OK
FT3  Fan 0     Cooling 0    Low             3 sec OK
RP0  Board 0   1.5 VDC 0   1.48 Volts DC    3 sec OK
RP0  Board 0   3.3 VDC 0   3.33 Volts DC    3 sec OK
RP0  Board 0   Temp. 0    26    Degrees C   3 sec OK
RP0  Board 0   Temp. 1    28    Degrees C   3 sec OK
RP0  SDC 0     Temp. 0    62    Degrees C   3 sec OK
RP0  AR 0      Temp. 0    46    Degrees C   3 sec OK
RP0  DX 0      Temp. 0    54    Degrees C   3 sec OK
RP0  DX 1      Temp. 0    46    Degrees C   3 sec OK
/N0/SB0 Board 0   1.5 VDC 0   1.50 Volts DC    2 sec OK
/N0/SB0 Board 0   3.3 VDC 0   3.29 Volts DC    2 sec OK
/N0/SB0 SDC 0     Temp. 0    68    Degrees C   2 sec OK
/N0/SB0 AR 0      Temp. 0    68    Degrees C   2 sec OK
/N0/SB0 DX 0      Temp. 0    63    Degrees C   2 sec OK
/N0/SB0 DX 1      Temp. 0    61    Degrees C   2 sec OK
/N0/SB0 DX 2      Temp. 0    59    Degrees C   2 sec OK
/N0/SB0 DX 3      Temp. 0    61    Degrees C   3 sec OK
/N0/SB0 SBBC 0    Temp. 0    67    Degrees C   3 sec OK
/N0/SB0 Board 1   Temp. 0    36    Degrees C   3 sec OK
/N0/SB0 Board 1   Temp. 1    39    Degrees C   3 sec OK
/N0/SB0 Cheetah 0 Temp. 0    66    Degrees C   3 sec OK
/N0/SB0 Cheetah 0 1.8 VDC 0   1.73 Volts DC    3 sec OK
/N0/SB0 Cheetah 1 Temp. 0    62    Degrees C   3 sec OK
/N0/IB6 Board 0   1.5 VDC 0   1.50 Volts DC    2 sec OK
/N0/IB6 Board 0   3.3 VDC 0   3.33 Volts DC    3 sec OK
/N0/IB6 Board 0   5 VDC 0     4.95 Volts DC    3 sec OK
/N0/IB6 Board 0   12 VDC 0   12.11 Volts DC    3 sec OK
/N0/IB6 Board 0   Temp. 0    38    Degrees C   3 sec OK
/N0/IB6 Board 0   Temp. 1    41    Degrees C   3 sec OK
/N0/IB6 SDC 0     Temp. 0    62    Degrees C   3 sec OK
/N0/IB6 AR 0      Temp. 0    57    Degrees C   3 sec OK
/N0/IB6 DX 0      Temp. 0    55    Degrees C   3 sec OK
/N0/IB6 DX 1      Temp. 0    57    Degrees C   3 sec OK
/N0/IB6 SBBC 0    Temp. 0    58    Degrees C   3 sec OK
/N0/IB6 IOASIC 0  Temp. 0    65    Degrees C   3 sec OK
/N0/IB6 IOASIC 1  Temp. 1    54    Degrees C   3 sec OK
```

CODE EXAMPLE 5-10 showplatform Command

```
3800-sc0:SC> showplatform

Domain    Solaris Nodename    Domain Status            Keyswitch
--------  ------------------  -----------------------  -------------
A         domainA             Active - Solaris         on
B         -                   Powered Off              off
C         -                   Powered Off              off
D         -                   Powered Off              off

The system controller is configured to be on a network.
Network settings: static
Hostname: 3800-sc0
IP Address: 10.10.3.120
Netmask: 255.255.255.0
Gateway: 10.10.3.1
DNS Domain: somedomain.com
Primary DNS Server: 10.101.30.6
Secondary DNS Server:

Loghost for Platform: loggit
Log Facility for Platform: local0

SNMP Agent: enabled
Chassis Description: Sun Fire 3800
Chassis Contact: admin@west
Chassis Location:  Rm 1500

ACL for Domain A: SB0 SB2 IB6 IB8
ACL for Domain B: SB0 SB2 IB6 IB8
ACL for Domain C: SB0 SB2 IB6 IB8
ACL for Domain D: SB0 SB2 IB6 IB8

SC POST diag Level: min

Chassis is in single partition mode.


Frame: 0006ac:03480a(Sat Feb 16 09:01:58 PST 2002)
     Fail LED:OffService LED:Off

Fan Trays information

                  Left Tray    Right Tray
       State:            OK            OK
   Power LED:            On            On
   Fault LED:           Off           Off
 Service LED:           Off           Off
       Speed:          Slow          Slow
 Temperature:        Normal        Normal
       Power:            OK            OK
       Fan 0:            OK            OK
       Fan 1:            OK            OK
       Fan 2:            OK            OK

RTS information
Rear/Left- OK, Connected to the System
Rear/Right- OK
Front/Left- OK, Connected to the System
Front/Right- OK
```

Platform Maintenance Tasks

In addition to configuration and environmental monitoring, the SC is also responsible for system maintenance tasks such as virtual keyswitch, controlling power, and blacklisting and CPU/Memory board testing.

Virtual Keyswitch

The SC provides control of domain through the use of virtual keyswitches. Using the `setkeyswitch` command enables you to power and boot domains. Each domain has a virtual keyswitch. You can use the `showkeyswitch` command to check the virtual keyswitch position.

TABLE 5-3 Virtual Keyswitch Commands

Command	Description
`setkeyswitch`	Sets the keyswitch position for a domain.
`showkeyswitch`	Displays the keyswitch position.

TABLE 5-4 Virtual Keyswitch Positions

Key position	Description
`on`	Powers on all system boards and initializes and runs startup diagnostics on the domain. Puts the domain into the OpenBoot™ PROM (OBP). If the OpenBoot PROM `auto-boot?` parameter is set to `true`, the system boots to the Solaris OE.
`standby`	Powers on all system boards in the domain but the domain is not initialized and diagnostics are not run.
`off`	Puts all system boards in the domain in the low-power mode. In this state, the boards can be removed from the system.
`diag`	Similar to `setkeyswitch on`. Powers and initializes the domain and runs diagnostics with the `diag-level` set to `max` and in the verbose mode.
`secure`	Similar to `setkeyswitch on`. Ignores the `break`, `reset` and `xir` commands. Prevents PROM updates to system and I/O boards.

The following keyswitch command examples are from a Sun Fire 3800 system.

CODE EXAMPLE 5-11 `showkeyswitch` Command

```
3800-sc0:A> showkeyswitch
keyswitch is: on
```

CODE EXAMPLE 5-12 `setkeyswitch` Command Examples

```
3800-sc0:A> setkeyswitch off

This will abruptly terminate Solaris in domain A.
Do you want to continue? [no]


3800-sc0:A> setkeyswitch off
Powering boards off ...


3800-sc0:A> setkeyswitch on
Powering boards on ...
Testing CPU Boards...
Loading the test table from board SB0 PROM 0 ...
{/N0/SB0/P2} Running CPU POR and Set Clocks
{/N0/SB0/P0} Running CPU POR and Set Clocks
{/N0/SB0/P1} Running CPU POR and Set Clocks
.
.
.


3800-sc0:A> setkeyswitch standby
Powering boards on ...
```

Note – The `setkeyswitch` command will warn you if the domain is running.

Controlling Power

For maintenance or troubleshooting purposes, you can control the power for system boards. Power supplies and fan trays can be controlled with the `poweron` and `poweroff` commands. You can control power for individual components or you can power off and on the entire platform.

The following system components are accessible with the `poweron` and `poweroff` commands

- Sun Fire 6800/4810/4800/3800 systems

  CPU/Memory boards

  I/O modules

Repeater boards

Power grids

Power supplies

Fan trays

- Sun Fire 15K/12K systems

  CPU/Memory boards

  I/O modules

  Expander boards

  System Controller boards*

  Centerplane support boards

  Power supplies

  Fan trays

* Only the spare System Controller board can be powered off.

CODE EXAMPLE 5-13 Power Control Examples

```
nudan0-sc0:SC> poweroff SB0 IB6
/N0/SB0: powered off
/N0/IB6: powered off



nudan0-sc0:SC> poweron all
PS0: powered on
PS1: powered on
PS2: powered on
RP0: powered on
RP2: powered on
/N0/SB0: powered on
/N0/IB6: powered on
/N0/IB8: powered on
```

Blacklisting and CPU/Memory Board Testing

The SC can test CPU/Memory boards that are not assigned to a domain. This enables you to run diagnostics on a board before you add it to a domain.

CODE EXAMPLE 5-14 `testboard` Command

```
3800-sc0:SC> testboard SB0
{/N0/SB0/P0} Running CPU POR and Set Clocks
{/N0/SB0/P2} Running CPU POR and Set Clocks
{/N0/SB0/P3} Running CPU POR and Set Clocks
{/N0/SB0/P2} @(#) lpost 5.12.5 2001/09/26 15:47
{/N0/SB0/P3} @(#) lpost 5.12.5 2001/09/26 15:47
.
.
.
```

From the SC you can put system boards on the *blacklist***.** Boards placed on the blacklist are still assigned to the domain, but will not be tested or configured, and are unusable by the Solaris OE. You can use the `disablecomponent` and `enablecomponent` commands to maintain the blacklist**.** You can use blacklisting to remove a marginal or failed component from the domain until a replacement is available. You can also use blacklisting to make configuration changes for testing or for benchmarking purposes.

CODE EXAMPLE 5-15 `disablecomponent` Command

```
3800-sc0:SC> disablecomponent IB8
IB8: will be disabled at the next boot sequence.
```

Redundant SCs

All Sun Fire systems should be configured with redundant SCs because the SC supplies all of the system clocks. Therefore, these boards must be functional for the system to run. In the event of a failure, all domains would stop. Without a functioning SC no domains can run. The controllers self monitor for hardware and software failures or hangs. If configured with a second controller, in the event of a failure, all of the controller's functions fail over to the spare controller with no interruption of running domains. This also enables replacement of the failed controller without interruption.

Other SC Functions

Depending on site preferences, the SC logs and/or forwards all error messages and status messages the SC and domains generate. This error message handling is configurable to use SNMP or the `syslog` facility.

All updates to system board or I/O module PROMs are made through the SC operating system, or ScApp. You can make diagnostic updates to the Sun Fire 6800/4810/4800/3800 system SCs by updating the PROMs. You can make OS or SMS updates to the Sun Fire 12K/15K system SCs by applying patches or application upgrades.

CHAPTER 6

Configuring Your System

This chapter describes in detail how to configure the systems you have used so much time designing. Much of the emphasis is on I/O layout and how to best set up the domains. The descriptions in this chapter review the configuration of each of the example designs in Chapter 4 so you can see how to get them up and running.

This chapter covers the following topics:

- Laying Out the I/O
- Laying Out the Domains
- Using Sun Fire 6800 System Power Grids
- Security
- Configuration Steps
- Configuration Examples
- Conclusion

Laying Out the I/O

The following sections describe the I/O layout methodology.

Disk Layout Guidelines

When designing your system, you decided what types of RAID schemes you were going to use for each of your disk volumes. What remains is to lay out these volumes to optimize their performance and redundancy characteristics.

If you followed our advice and purchased an enclosure capable of hardware RAID, configuration for RAID 5 should amount to utilizing the hardware properties of the storage enclosure. In enclosures such as the Sun StorEdge T3 array, you can select the number of disks you want in a given RAID 5 volume, as well as some other options. Simply size each volume to match your requirements.

For RAID 0+1, configuration can get much more complicated. It will likely take some experimentation with your application to determine the best layout for your needs.

To start with, though, the first step is to lay out the I/O properly. You have designed and purchased the right elements, so now it is just a matter of putting them all in the right slots.

The easiest way to ensure the best performance and redundancy from your disk I/O is by following the rule discussed in Chapter 3—spread the volumes across as many enclosures as possible, then distribute those enclosures across as many controllers as possible, and finally distribute your controllers across as many different I/O buses as possible.

You should balance this with ease of administration, however. If you have 10 enclosures, you probably do not want your volumes striped across all 10 enclosures. because it is impossible to remove an enclosure without affecting all of your data. This means you cannot service a single enclosure without incurring substantial downtime.

Instead, you can follow a simple three-step process when designing the volumes:

1. Connect your enclosures to as many different controllers as possible.
2. Within one or more of these enclosures, stripe across separate disks.
3. Mirror that stripe to the same configuration in *different* enclosures and subsequently different controllers.

Note – You should avoid allowing different halves of a mirrored volume to occupy the same enclosure, as this can introduce a single point of failure.

How you split up the disks and volumes is dependent on the storage device, as well as the needs of your application. In a device that is “just a bunch of disks” (JBOD), you can stripe across multiple disks in a single enclosure, and then mirror that to a second enclosure. In devices that have their own hardware RAID, such as the Sun StorEdge T3 arrays, configuration can be more complicated. In these arrays, for example, you should first define the volumes within the device itself, and then use a volume manager to administer these in the Solaris OE. Subsequent examples go through some specific configurations.

Boot Device Mirroring

As you found during the design process, you almost always want to mirror your boot drives, as this is an easy way to reduce downtime. The easiest way to do so is by doubling the number of boot devices, as we addressed in the design section. During configuration, you then mirror these two separate devices to one another. For example, if you are using two Sun StorEdge S1's, you would configure your volume manager software so that each S1 is one half of the root device mirror.

If utilizing two Sun StorEdge D240 media trays is not possible, the Sun StorEdge D240 media tray should be placed in "split-bus" mode, and the boot drive for the domain should be mirrored across the two halves of the Sun StorEdge D240 media tray and connected to separate SCSI cards. FIGURE 6-1 shows an example of this setup.

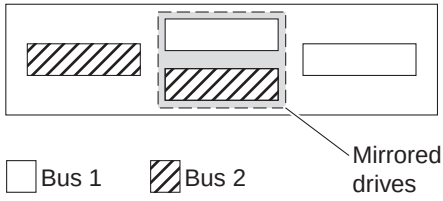


FIGURE 6-1 Mirrored Disks on Different Buses

A good configuration utilizes two Sun StorEdge D240 media trays with the boot drives mirrored between the two arrays and separate SCSI cards. This configuration is good because even in split-bus mode, the two SCSI buses of a Sun StorEdge D240 media tray share a common centerplane, which could be considered a single point of failure (SPOF). FIGURE 6-2 shows an example of this setup.

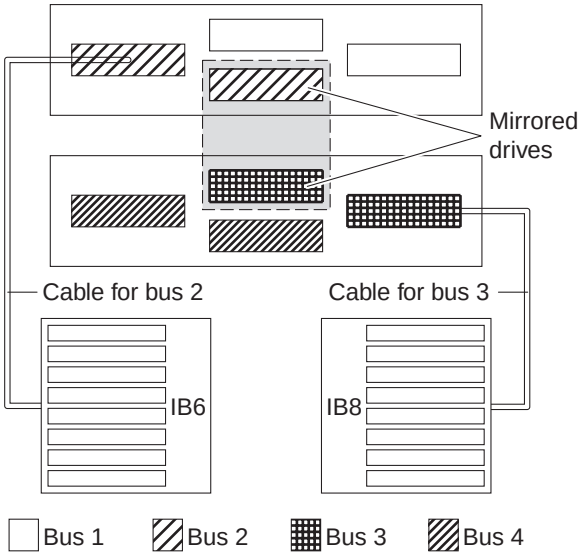


FIGURE 6-2 Mirroring Disks Between Two Sun StorEdge D240 Media Trays and Two Controllers on Two I/O Assemblies

Network Cards

You can follow much of the same philosophy on disks with network cards. Namely, each network card should be connected to a separate I/O bus. This gives you the best performance and also allows for redundancy because removing an I/O module removes only a single interface.

Dynamic Reconfiguration

To use DR for I/O applications, you should remember one simple rule:

> **After removing an I/O assembly, you must still have at least one path to each device**.

This means that each device should have at least two connections, each to a different board. The best way to ensure that this is true is to draw a diagram of the I/O parts, using a different color for the cables going to each I/O assembly. Then, selectively erase each color (as if that board were removed), and make sure you do not have any devices without connections going to them.

Even if you are not going to use DR, this is a good process to follow if uptime is critical. Doing so allows you to remove a complete I/O assembly and still have a functional system (even if you have to reboot to do it).

Standard Setup

Here is an example of the “standard setup,” which is an abstraction of how most of the systems look. This configuration, based on two I/O boards, allows you to dynamically reconfigure any single I/O assembly and still have a working system. You can also remove any single I/O device without affecting the volumes.

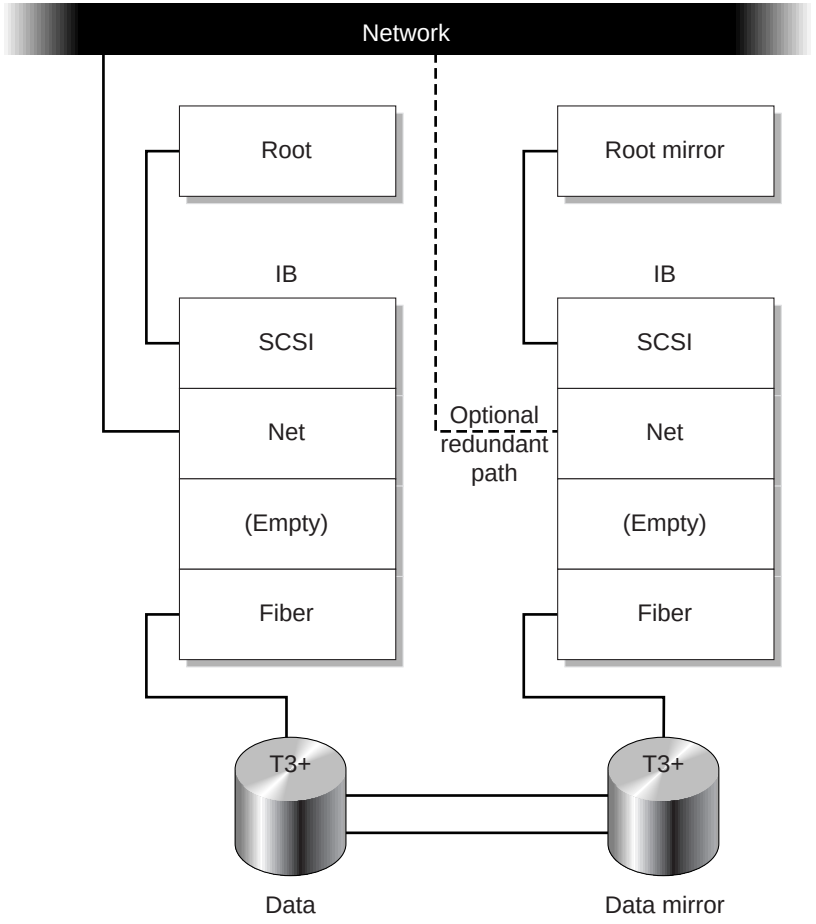


FIGURE 6-3 Standard Setup

Laying Out the Domains

Although you chose how to use domains during the design process, at that time you did not specify where the domains would be located in the box. Now that you have the system, you can decide exactly to how to divide it up.

Remember that you should split up CPU *and* I/O boards, so that each domain has at least one of each. To help you do so, here is a simple table (TABLE 6-1) you can fill in to make sure all your boards are accounted for. When filling out this table, make sure you refer to the domain layout restrictions in Chapter 1.

TABLE 6-1 Domain Layout Decision Guide

Domain	Purpose	CPU/Memory boards	I/O boards

When using multiple domains in a single system, keep in mind that they are not completely physically separated. They share many components, introducing single points of failure to both domains. For example, in the event of a Fireplane failure, all domains in a system will be rendered inoperable. While some of these concerns can be mitigated by using partitions (discussed below), you should avoid placing multiple mission-critical servers inside the same chassis.

Partitions

A partition divides the repeater boards on the Fireplane by two, effectively cutting the machine in half. By splitting them up, you are reducing the number of repeater boards a particular domain relies on, resulting in fewer single points of failure. Since a repeater board failure will crash all domains using that repeater board, this can be an advantage.

The downside to partitions is that by halving the number of repeater boards a domain can use, this means each domain gets half the data bus bandwidth it would otherwise have. Currently, this does not appear to be much of a real-world concern, as no applications have run into this limitation yet. Nonetheless, it is an important disadvantage to keep in mind. Essentially, utilizing partitioning gives you increased availability in exchange for decreased theoretical performance.

We do not recommend that you rely on partitions alone for increasing your system's RAS, hence the reason we did not mention them during the design portion of this book. Even with partitions, multiple domains can still share power, fan trays, and numerous other components, each of which is a potential single point of failure. If you think you need partitions due to their fault-isolation properties, you should instead consider separate Sun Fire systems for each domain.

Partitions should be used as follows:

- If you are designing multiple mission-critical servers, you should design them in separate Sun Fire systems when possible.
- If you are purchasing a system with an additional test domain, or where multiple domains in a single system is a clear advantage, you should utilize partitions in the configuration stage. Ignoring partitions until the configuration phase will help ensure that you design a system with the best RAS properties.

Using Sun Fire 6800 System Power Grids

If you are using multiple domains in the Sun Fire 6800 system, you should use its two separate power grids. The rule you should follow is:

Keep all devices for a domain in the same power grid if possible.

The Sun Fire 6800 system differs from all other Sun Fire models in that it has two separate internal power grids, each supplied by a different redundant transfer unit (RTU).

FIGURE 6-4 shows how the boards are separated on the internal power grids.

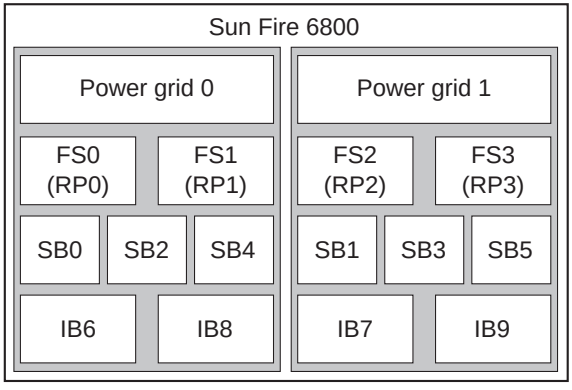


FIGURE 6-4 Sun Fire 6800 Power Grids

TABLE 6-2 lists how the boards are separated on the internal power grids.

TABLE 6-2 Separation of Boards on Sun Fire 6800 System Power Grid

Power grid 0	Power grid 1
SB0	SB1
SB2	SB3
SB4	SB5
IB6	IB7
IB8	IB9
RP0	RP2
RP1	RP3

In a dual partition configuration, domains A and B are associated with RP0 and RP1, so boards for those domains should be chosen from power grid 0. Domains C and D are associated with RP2 and RP3, so boards for those domains should be chosen from power grid 1. When configuring domains, you should avoid using boards from both power grids so that in the unlikely event of an RTU failure, the effects of the power loss will be limited. If a domain depends on both power grids, there is twice the likelihood of a power failure causing a failure of the domain. The following examples illustrate this.

Example 1:

Domain A is created using RP0, RP1, SB0, and IB6. Domain C is created using RP2, RP3, SB1, and IB7. If the RTU supplying power grid 1 fails, domain C would fail, but there would be no effect on the operation of domain A. A failure of the RTU supplying grid 0 would cause domain A to fail, but domain C would remain available.

Example 2:

Domain A is created using RP0, RP1, SB1, and IB6. Domain C is created using RP2, RP3, SB0, and IB7. Because domain A was created using SB1 and domain C was created using SB0, the domain has components in both power grids, so a failure of either RTU (or power grids) would cause all domains to fail.

Security

Sun Fire systems support numerous different security features. You can set different passwords for the platform and domain shells, and also use Access Control Lists (ACLs) to restrict the resources to which a domain has access. In addition, the Sun Fire 15K system also supports 18 different levels of access, so that you can precisely specify which users can do what tasks. Managing this level of security is beyond the scope of this book; however, Example 3 illustrates some of the features.

For best security and performance, you should configure the SCs on a switched, private network. This configuration not only reduces to traffic to the SCs (thus increasing performance), but it also enables you to restrict access to the SCs at the network level.

If possible, you should connect this private network to a separate administrative workstation, giving you direct access to the SC.

Combined with a good site-wide policy, a private SC network and separate passwords for platforms and domains should give you good security. However, it is always important to remember that the security of a network is only as good as its weakest link.

If you are serious about security, we recommend you read a book devoted to the topic, such as *Practical UNIX & Internet Security* from O'Reilly and Associates.

Logging

The SC supports several logging schemes, but does not do any logging by default. Logging can be configured during the *setupplatform* phase. You should at least configure syslog logging and send system messages to a remote host of your choice (such as an admin workstation or central logging host). The SC also supports SNMP, and has a small 4k buffer that holds recent messages and can be accessed with the command `showlog`.

When setting up syslog, one way to organize messages is to use one facility (such as local0) for the platform messages, and another facility (such as local1) for all domain messages. In doing so, you will have all of the Sun Fire systems platform messages in one log, and all domain messages in another. Alternatively, you could use a different facility for different hosts (combining platform and domain messages), or even use the standard facilities such as daemon and auth, depending on your site.

Note – Remember that you must also make changes to the log host's `/etc/syslog.conf` file to enable syslog.

The SC also supports SNMP messages, which you can use to communicate with monitoring packages such as SunMC. By default, the public and private community strings are set up as the letter of the domain followed by a hyphen and the term "public" or "private."

So, domain A would be "A-public" and "A-private", domain B would be "B-public" and "B-private," and so forth. The platform is denoted as "P-public" and "P-private." It is strongly recommended that you change these defaults for security reasons.

Finally, you should note that not all messages can be logged using syslog or SNMP, since these require that the system can communicate via the network. Critical messages such as panics and reboots will not be logged remotely. In order to catch all such messages, we recommend that you use a Solaris command such as "tail" or "script" on an open console connection. Or, if you are using a network terminal server, this may provide additional functionality to route console messages through the network.

Configuration Steps

Properly configuring a Sun Fire system is just as important as properly designing one. Conveniently, this process has a certain set of "cookbook" steps that you can follow:

1. Laying and connecting the I/O.
2. Plugging in and powering on the system.
3. Configuring each SC.
4. Configuring the ACLs and domains.
5. Booting and installing each domain like a standard Sun server.

The paragraphs that describe the process of configuring the example systems also describes each of these steps. First, briefly look at each step in a little more detail.

1. Lay out and connect your I/O.

When you have decided where to put each I/O card and which device to connect it to, rearrange your I/O appropriately. Refer to the manual for each individual device to ensure you are handling it correctly. If you need assistance, contact Sun Service.

2. **Connect and power on your system.**

   If you purchased the redundant power, make sure to connect each redundant supply to a *separate* power source. If you do not have a separate power source available, you should leave the redundant power unplugged. Otherwise, the Sun Fire system can get confused in certain brownout states, and power off the system even when adequate power is available. Note that the *redundant power units do not load share on any Sun Fire systems except the Sun Fire 15K system.*

3. **Configure each SC through a serial line and reboot it.**

   There are several differences between the Sun Fire 15K/12K SCs and the Sun Fire 6800/4810/4800/3800 SCs. The steps needed to configure each type are detailed in the following paragraphs.

   Sun Fire 6800/4810/4800/3800

   Connect each SC to an RS-232 serial line, either to an administration workstation or a terminal server. Once you are connected to the SC, perform these steps:

   a. **Type `0` to select the platform shell.**

   b. **Use the `setdate` command to set the date and time.**

   c. **Use the `password` command to set the password for the SC.**

   d. **Run the `setupplatform` command.**

   e. **Type `reboot` to cycle the SC so that you can access it through the network.**

Note – You must configure each SC individually using the serial connection.

You should name your SCs *system*`-sc0` and *system*`-sc1` (where *system* is the name of your Sun Fire system) to ease administration, although this is not required.

Sun Fire 15K/12K

Connect each SC to an RS-232 serial line, either to an administration workstation or a terminal server. Once you are connected to the SC, perform these steps:

a. **Boot the SC.**

b. **The SC runs the Solaris Operating Environment. You will be asked for network, date and password during the initial boot.**

c. **Configure the SMS software using `smsconfig` commands.**

d. **Reboot the SCs.**

4. **Connect to the SC through the network and configure the domains.**

   There are several differences between the Sun Fire 15K/12K SCs and the Sun Fire 6800/4810/4800/3800 SCs. The steps needed to configure each type are listed below.

   Sun Fire 6800/4810/4800/3800:

   Once you have configured the network with the `setupplatform` command,[1] connect to the SC as your primary means of access with `telnet`. From this point on, you only have to connect to the primary SC to administer the system.

   a. **From the Platform Shell configure the ACLs for the domains with** `setupplatform`**.**

   b. **Enable power for the platform with the** `poweron` **command.**

   c. **Type** `disconnect` **to exit the Platform Shell and choose the number from the menu that corresponds to the domain to enter the Domain Shell.**

   d. **Use the** `password` **command to set the password for the domain.**

   e. **Run the** `setupdomain` **command to make desired changes to OBP variables such as** *diag-level* **and** *auto-boot?* **and to configure message logging.**

   f. **Add boards to the domains with the** `addboard` **command.**

Note – You should set the `password` for all domains on the system, even if you do not plan to use them. This ensures that no one can alter a domain's settings surreptitiously.

   Sun Fire 15K/12K:

   After you have configured the network and rebooted the SCs, use rlogin or telnet to access the SC and administer the system. Connect to the primary SC to create the domains.

   a. **Setup the ACLs for each domain with the** `setupplatform` **command.**

   b. **Configure a domain administrator account with the** `smsconfig` **command.**

   c. **Assign boards to the domains with the** `addboard` **command.**

   d. **Add a "tag" to the domain with the** `addtag` **command (optional).**

5. **Boot and install each domain like a standard Sun server.**

   After the domain is configured to your satisfaction, boot it with the command `setkeyswitch on`. This command powers on the appropriate components, runs POST, then attempts to boot the domain. Essentially, it is just as though you turned

1. The Sun Fire 15k system requires some additional setup steps, which the examples describe.

the keyswitch to the ON position on one of the Sun Enterprise systems. At OBP, the system acts just like any other Sun server and can be installed using JumpStart™ software.

Configuration Examples

This section describes how to configure each of the real-world examples you designed in Chapter 4.

- NFS Server
- Simulation Server
- Database Server

Example 1—NFS Server

This is the system that you designed to act as the main NFS and backup server for a graphical design company. You purchased a single domain Sun Fire 3800 system with four CPUs (two per CPU/Memory board) and about two terabytes of disk. You may want to review the final specification for this system in TABLE 4-5 to refresh your memory.

The step-by-step configuration procedure is:

1. **Lay out and connect the I/O.**

   When you purchased this system, you bought the proper equipment to enable you to mirror the root device. To do so, you should configure each S1 to go to a different I/O board ensuring the proper redundancy. You also need room for two FC-AL controllers, a DWIS controller for the tape drive, and the Gigabit Ethernet card.

   Piggybacking the boot device controllers on the same I/O bus as the tape stacker is a good strategy because most boot devices have extremely low activity, This leaves a whole set of slots in the middle with their own controller/bus for future expansion. (The Sun Fire 3800 system has two of these assemblies, for a total of 12 slots.)

   Looking at the I/O slots in the Sun Fire 3800 system (TABLE 6-3), you have the following choices.

TABLE 6-3 Sun Fire 3800 System Six Slot cPCI I/O Assembly Layout

Slot	Slot capability	I/O controller/bus
0	66/33 MHz 3.3 VDC	0/A
1	66/33 MHz 3.3 VDC	1/A
2	33 MHz 5 VDC	0/B
3	33 Mhz 5 VDC	0/B
4	33 MHz 5 VDC	1/B
5	33 MHz 5 VDC	1/B

Each of the disk controller pairs should be split between the two assemblies for best redundancy. The two FC-AL controllers, which will connect to the main data storage, should be the two cards with the highest load so you should put these on their own bus. For optimum performance, you should also make sure to keep the Gigabit Ethernet card and tape drive on a separate bus from these controllers. For best performance, place the Gigabit Ethernet card in one of the 66 MHz slots.

Finally, a boot device can be placed almost anywhere.

TABLE 6-4 shows the resulting configuration.

TABLE 6-4 Sun Fire 3800 System I/O Layout

IB / slot	Card	Card	IB / slot
IB 6 / 0	FC-AL controller (data storage)	FC-AL controller (data storage)	IB 8 / 0
IB 6 / 1	Gigabit Ethernet card		IB 8 / 1
IB 6 / 2			IB 8 / 2
IB 6 / 3			IB 8 / 3
IB 6 / 4		DWIS controller (tape stacker)	IB 8 / 4
IB 6 / 5	SCSI/Ethernet card	SCSI/Ethernet card	IB 8 / 5

2. Connect and power on your system.

Connect the main power cords and apply power to the system through the circuit breakers.

3. Configure each SC with a serial line, and reboot the SCs.

You have a networked terminal server, so connect the system to it and log in to the SC. Choose 0 from the menu to enter the *platform shell*. Set the date with the `setdate` command and use the `password` command to set a password for the platform. Then run the `setupplatform` command to assign a hostname and configure network parameters. After the configuration information is entered reboot the SC for the changes to take effect. For this example we will use *skynet* as the platform name.

CODE EXAMPLE 6-1 `setupplatform` Command Example

```
#> telnet term-server
Trying 137.13.23.92...
Connected to term-server
Escape character is '^]'.

System Controller 'noname.example.com':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console

    Input: 0

Platform Shell

noname:SC> setdate -t PST 052616352002
Sun May 26 16:35:00 PDT 2002

noname:SC> password
Enter new password:
Enter new password again:

noname:SC> setupplatform

Network Configuration
---------------------
Is the system controller on a network? [yes]:
Use DHCP or static network settings? [DHCP]: static
Hostname []: skynet-sc0
IP Address []: 119.122.6.87
Netmask []: 255.255.255.0
Gateway []: 119.122.6.1
DNS Domain: bob.com
Primary DNS Server []: 119.192.5.6
Secondary DNS Server []:

Rebooting the SC is required for changes in network settings to take effect.

Loghosts
--------
Loghost [ ]:
Log Facility [local0]:
```

CODE EXAMPLE 6-1 `setupplatform` Command Example *(Continued)*

```
SNMP
----
Platform Description [Sun Fire 3800]:
Platform Contact [ ]:
Platform Location [ ]:
Enable SNMP Agent? [no]:
ACLs
----
ACL for domain A [ SB0 SB2 IB6 IB8 ]:
ACL for domain B [ SB0 SB2 IB6 IB8 ]:
ACL for domain C [ SB0 SB2 IB6 IB8 ]:
ACL for domain D [ SB0 SB2 IB6 IB8 ]:

SC POST
-------
SC POST diag Level [min]:

Partition Mode
--------------
Configure chassis for single or dual partition mode? [single]:

noname:SC> reboot
Are you sure you want to reboot the system controller now? [no] yes

Rebooting. All telnet connections closed. Reestablish any needed connections.


Software Reset...


Diagnostic and boot messages omitted......


System Controller 'skynet-sc0':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console
```

Repeat this process for the secondary SC. The only difference for the secondary SC is the hostname and IP address that you assign it. After the SC reboots make sure your network settings are correct, or you will be unable to `connect` to the SC using telnet.Verify the information is correct with the `showplatform` command.

CODE EXAMPLE 6-2 Network Configuration Verification Example

```
#> telnet term-server
Trying 137.13.23.92...
Connected to term-server
Escape character is '^]'.
```

CODE EXAMPLE 6-2 Network Configuration Verification Example *(Continued)*

```
System Controller 'skynet-sc0':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console

    Input: 0

Platform Shell

skynet-sc0:SC> showplatform

Domain    Solaris Nodename    Domain Status            Keyswitch
--------  ------------------  -----------------------  -------------
A         -                   Powered Off              off
B         -                   Powered Off              off
C         -                   Powered Off              off
D         -                   Powered Off              off

The system controller is configured to be on a network.
Network settings: static
Hostname: skynet-sc0
IP Address: 119.122.6.87
Netmask: 255.255.255.0
Gateway: 119.122.6.1
DNS Domain: bob.com
Primary DNS Server: 119.192.5.6
Secondary DNS Server:

Loghost for Platform:
Log Facility for Platform: local0

SNMP Agent: disabled
Chassis Description: Sun Fire 3800
Chassis Contact:
Chassis Location:

ACL for Domain A: SB0 SB2 IB6 IB8
ACL for Domain B: SB0 SB2 IB6 IB8
ACL for Domain C: SB0 SB2 IB6 IB8
ACL for Domain D: SB0 SB2 IB6 IB8

SC POST diag Level: min

Chassis is in single partition mode.
```

4. Connect to the SC through the network and configure the domain.

You are configuring both boards in this system into a single domain. After using telnet to connect to the primary SC, choose `0` to access the platform shell to configure the ACL and power on the system.

```
#> telnet skynet-sc0
Trying 119.122.6.87....
Connected to skynet-sc0.
Escape character is '^]'.

System Controller 'skynet-sc0':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console

    Input: 0

Platform Shell

skynet-sc0:SC> setupplatform -p acls

ACLs
----
ACL for domain A [ SB0 SB2 IB6 IB8 ]: SB0 SB2 IB6 IB8
ACL for domain B [ SB0 SB2 IB6 IB8 ]: -
ACL for domain C [ SB0 SB2 IB6 IB8 ]: -
ACL for domain D [ SB0 SB2 IB6 IB8 ]: -

skynet-sc0:SC> showplatform -p acls

ACL for Domain A: SB0 SB2 IB6 IB8
ACL for Domain B:
ACL for Domain C:
ACL for Domain D:


skynet-sc0:SC> poweron all
Sun May 26 16:39:59 skynet-sc0 Platform.SC: FT0, fan speed, Low (4,1)
Sun May 26 16:39:00 skynet-sc0 Platform.SC: FT1, fan speed, Low (4,1)
Sun May 26 16:39:00 skynet-sc0 Platform.SC: FT2, fan speed, Low (4,1)
Sun May 26 16:39:01 skynet-sc0 Platform.SC: FT3, fan speed, Low (4,1)
PS0: powered on
PS1: powered on
PS2: powered on
RP0: powered on
RP2: powered on
/N0/SB0: powered on
/N0/SB2: powered on
/N0/IB6: powered on
/N0/IB8: powered on


skynet-sc0:SC>
```

Next, assign the system boards to the domain with the `addboard` command.

CODE EXAMPLE 6-3 `addboard` Command Example

```
#> telnet skynet-sc0
Trying 119.122.6.87....
Connected to skynet-sc0.
Escape character is '^]'.

System Controller 'skynet-sc0':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console

    Input: 1

Connected to Domain A

Domain Shell for Domain A

skynet-sc0:A> addboard SB0 SB2 IB6 IB8

skynet-sc0:A> showboards

Slot     Pwr Component Type                 State      Status     Domain
----     --- --------------                 -----      ------     ------
/N0/SB0  Off CPU Board                      Assigned   Not tested A
/N0/SB2  Off CPU Board                      Assigned   Not tested A
/N0/IB6  Off CPCI I/O board (F3800)         Assigned   Not tested A
/N0/IB8  Off CPCI I/O board (F3800)         Assigned   Not tested A

skynet-sc0:A> showdomain

Domain    Solaris Nodename    Domain Status            Keyswitch
--------  ------------------  -----------------------  -------------
A         -                   Powered Off              off

diag-level = default
verbosity-level = min
error-level = max
interleave-scope = within-board
interleave-mode = optimal
reboot-on-error = false
OBP.use-nvramrc? = <OBP default>
OBP.auto-boot? = false
OBP.error-reset-recovery = <OBP default>

Loghost for Domain A:
Log Facility for Domain A: local0

SNMP Agent: disabled
Domain Description:
Domain Contact:

ACL for Domain A: SB0 SB2 IB6 IB8

skynet-sc0:A>
```

For this system we are using the default obp settings and are not enabling logging. The settings can be displayed with the `showdomain` command.

Note that all of the available boards were added to domain A because the system runs as a single domain.

5. **Boot and install the domain like a standard Sun server.**

   Once the domain is set up, run `setkeyswitch` to power on the system boards, test the hardware, and bring it up to install the Solaris OE.

CODE EXAMPLE 6-4 Power On and Domain Test Example

```
#> skynet-sc0:A> setkeyswitch on
Powering boards on ...
Testing CPU Boards ...
Loading the test table from board SB0 PROM 0 ...
{/N0/SB2/P0} Running CPU POR and Set Clocks
{/N0/SB2/P1} Running CPU POR and Set Clocks
{/N0/SB2/P0} @(#) lpost 5.12.5 2001/09/26 15:47
{/N0/SB2/P1} @(#) lpost 5.12.5 2001/09/26 15:47


Diagnostic messages omitted.....


{/N0/SB0/P0} DCB_DECOMP_OBP command succeeded
{/N0/SB0/P0} GLOBAL IO SRAM on board 6
{/N0/SB0/P0} Decompress OBP done
{/N0/SB0/P0} DCB_ENTER_OBP command succeeded
{/N0/SB0/P1} DCB_ENTER_OBP command succeeded
{/N0/SB2/P0} DCB_ENTER_OBP command succeeded
{/N0/SB2/P1} DCB_ENTER_OBP command succeeded
Entering OBP ...


Sun Fire 3800
OpenFirmware version 5.12.5 (09/26/01 15:46)
Copyright 2001 Sun Microsystems, Inc.  All rights reserved.
SmartFirmware, Copyright (C) 1996-2001.  All rights reserved.
8192 MB memory installed, Serial #xxxxxxxx.
Ethernet address x:x:xx:x:xx:xx, Host ID: xxxxxxxx.


{0} ok
```

Example 2—Simulation Server

You purchased a Sun Fire 6800 system for use in research simulations for a university chemistry department. This system has two domains—a three-board simulation server, as well as a single-board NFS server that houses the data sets and home directories.

1. **Lay out and connect the I/O.**

   Because you are configuring two separate domains, you should ensure that the I/O is properly divided so that each domain has access to the appropriate devices. First decide where to locate the domains.

You think that the simulation server could grow by one to two boards over the next year. To prevent the domains from being interwoven, you can put the simulation domain on the first three boards (slots 0, 2, and 4) and the NFS server in the last slot (5). That way the simulation domain can expand into the empty space. This method also takes advantage of the Sun Fire 6800 system's dual power grids, powering the two domains on separate grids. It also eases administration, as the domains are physically contiguous.

In terms of I/O, you have four I/O assemblies available, each with the slots listed in TABLE 6-5.

TABLE 6-5 Sun Fire 6800/4810/4800 Eight Slot PCI I/O Assembly Layout

Slot	Slot capacity	I/O controller/bus
0	33 MHz 5 VDC (short)	0/B
1	33 MHz 5 VDC (short)	0/B
2	33 MHz 5 VDC	0/B
3	66/33 Mhz 3.3 VDC	0/A
4	33 MHz 5 VDC	1/B
5	33 MHz 5 VDC	1/B
6	33 MHz 5 VDC	1/B
7	66/33 MHz 3.3 VDC	1/A
<--- Handle		Ejector levers ---------->

Note – The I/O assembly is identical for all four assemblies so the assembly is installed "upside down" in the two right locations. This means that the slots in the two right I/O assemblies are numbered from bottom-up instead of top-down.

The simulation domain does not need redundancy for its local disk, but it does need speed. You should spread out the two high-traffic cards (network card and SCSI card for data) across separate I/O buses for best performance, but you can leave them in a single I/O assembly. TABLE 6-6 lists the configuration for IB8 (the top right board)—the simulation server

TABLE 6-6 I/O Layout for Simulation Server

Card	IB / Slot
SCSI controller (D2 storage)	IB 8 / 7
SCSI/Ethernet card (media tray)	IB 8 / 6
	IB 8 / 5
	IB 8 / 4
Gigabit Ethernet card	IB 8 / 3
	IB 8 / 2
	IB 8 / 1
	IB 8 / 0

You chose this I/O assembly to keep the server physically together, and to take advantage of the dual power grids. Choosing this I/O layout means the simulation server is powered completely by grid 0.

For the NFS server, you need both good performance and redundancy, so you should spread the I/O controllers across multiple assemblies. TABLE 6-7 lists the I/0 layout for the NFS server.

TABLE 6-7 I/O Layout for NFS Server

IB / Slot	Card
IB 9 / 0	SCSI controller (media tray)
IB 9 / 1	
IB 9 / 2	
IB 9 / 3	Gigabit Ethernet card
IB 9 / 4	
IB 9 / 5	
IB 9 / 6	
IB 9 / 7	FC-AL controller (Sun StorEdge T3-1 array)
IB 7 / 0	SCSI controller (media tray)
IB 7 / 1	
IB 7 / 2	
IB 7 / 3	

TABLE 6-7 I/O Layout for NFS Server *(Continued)*

IB / Slot	Card
IB 7 / 4	
IB 7 / 5	
IB 7 / 6	
IB 7 / 7	FC-AL controller (Sun StorEdge T3-2 array)

FIGURE 6-5 shows how these cards will be connected to the I/O.

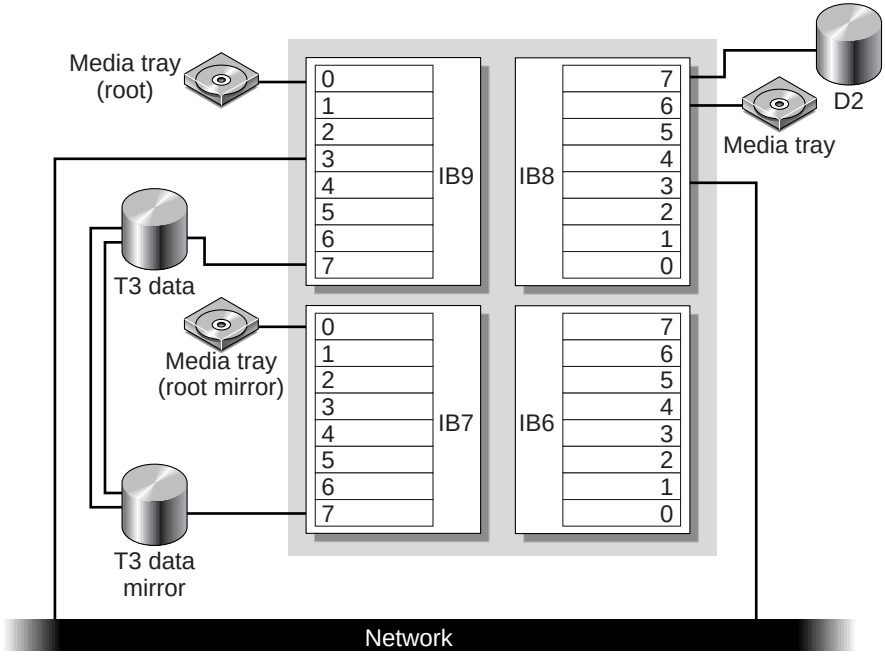


FIGURE 6-5 I/O Connections

2. **Connect and power on the system.**

   Connect the main power cords and power on the system.

3. **Configure each SC through a serial line and reboot the system.**

   This site does not have a terminal server, so instead use the network administrators laptop to connect to each SC in turn. Use the `setupplatform` command and then reboot. In this example there are two domains. We have decided to isolate the

domains and place them in separate partitions. This is accomplished by changing the *partition mode* with the `setupplatform` command. Remember to set the date and password for the platform.

CODE EXAMPLE 6-5 `setupplatform` Example

```
#> tip hardwire

Connected:

System Controller 'noname.example.com':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console

    Input: 0

Platform Shell

noname:SC> setdate -t PST 050417302002
Sat May 04 17:30:00 PDT 2002

noname:SC> password
Enter new password:
Enter new password again:

noname:SC> setupplatform

Network Configuration
---------------------
Is the system controller on a network? [yes]:
Use DHCP or static network settings? [DHCP]: static
Hostname []: simland-sc0
IP Address []: 119.122.6.45
Netmask []: 255.255.255.0
Gateway []: 119.122.6.1
DNS Domain: west.com
Primary DNS Server []: 109.132.51.6
Secondary DNS Server []:

Rebooting the SC is required for changes in network settings to take effect.

Loghosts
--------
Loghost [ ]:
Log Facility [local0]:

SNMP
----
Platform Description [Sun Fire 6800]:
Platform Contact [ ]:
Platform Location [ ]:
Enable SNMP Agent? [no]:
ACLs
```

CODE EXAMPLE 6-5 `setupplatform` Example *(Continued)*

```
----
ACL for domain A [ SB0 SB2 IB6 IB8 ]:
ACL for domain B [ SB0 SB2 IB6 IB8 ]:
ACL for domain C [ SB0 SB2 IB6 IB8 ]:
ACL for domain D [ SB0 SB2 IB6 IB8 ]:

SC POST
-------
SC POST diag Level [min]:

Partition Mode
--------------
Configure chassis for single or dual partition mode? [single]: dual
May 04 17:35:37 noname Platform.SC: Moving RP2, RP3 to partition 1
May 04 17:35:37 noname Platform.SC: Chassis is in dual partition mode.

noname:SC> reboot
Are you sure you want to reboot the system controller now? [no] yes

Rebooting. All telnet connections closed. Reestablish any needed connections.


Software Reset...


Diagnostic and boot messages omitted.....


System Controller 'skynet-sc0':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console
```

4. **Connect to the SC through the network and configure the domains.**

   Use the `setupplatform` command to manage the ACL. Power on the platform with the `poweron` command, then use `addboard` to create the domains.

CODE EXAMPLE 6-6 ACL Setup and Platform Power On Example

```
#> telnet simland-sc0
Trying 119.122.6.45....
Connected to simland-sc0.
Escape character is '^]'.

System Controller 'chem-sc0':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console

Input: 0

Platform Shell

simland-sc0:SC> setupplatform -p acls

ACLs
----
ACL for domain A [ SB0 SB2 SB4 SB5 IB6 IB7 IB8 IB9]:SB0 SB2 SB4 IB6 IB8
ACL for domain B [ SB0 SB2 SB4 SB5 IB6 IB7 IB8 IB9]:-
ACL for domain C [ SB0 SB2 SB4 SB5 IB6 IB7 IB8 IB9]:SB5 IB7 IB9
ACL for domain D [ SB0 SB2 SB4 SB5 IB6 IB7 IB8 IB9]:-

simland-sc0:SC> showplatform -p acls

ACL for Domain A: SB0 SB2 SB4 IB6 IB8
ACL for Domain B:
ACL for Domain C: SB5 IB7 IB9
ACL for Domain D:


simland-sc0:SC> poweron grid0 grid1
May 04 17:39:38 simland-sc0 Platform.SC: FT0, fan speed, Low (4,1)
May 04 17:39:39 simland-sc0 Platform.SC: FT1, fan speed, Low (4,1)
May 04 17:39:39 simland-sc0 Platform.SC: FT2, fan speed, Low (4,1)
May 04 17:39:40 simland-sc0 Platform.SC: FT3, fan speed, Low (4,1)
grid0: powered on
grid1: powered on

simland-sc0:SC>
```

When laying out the I/O, you decided how to lay out the domains, so now it is a matter of logging into each domain shell and using the `addboard` and `setupdomain` commands. In this example we are changing the OBP variable *auto-boot?* to *true* and configuring each domain to send system messages to the **syslog**

host *chemlogger*. You can check the configuration with the `showdomain` command.

CODE EXAMPLE 6-7 Board Assignment and Domain Setup Example

```
System Controller 'simland-sc0':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console

    Input: 1

Connected to Domain A

Domain Shell for Domain A

simland-sc0:A> addboard SB0 SB2 SB4 IB6 IB8

simland-sc0:A> showboards

Slot     Pwr Component Type                State      Status     Domain
----     --- --------------                -----      ------     ------
/N0/SB0  Off CPU Board                     Assigned   Not tested A
/N0/SB2  Off CPU Board                     Assigned   Not tested A
/N0/SB4  Off CPU Board                     Assigned   Not tested A
/N0/IB6  Off PCI I/O board                 Assigned   Not tested A
/N0/IB8  Off PCI I/O board                 Assigned   Not tested A

simland-sc0:A> setupdomain

Domain Boot Parameters
----------------------
diag-level [default]:
verbosity-level [min]:
error-level [max]:
interleave-scope [within-board]:
interleave-mode [optimal]:
reboot-on-error [false]:
OBP.use-nvramrc? [true]:
OBP.auto-boot? [false]: true
OBP.error-reset-recovery [<OBP default>]:

Loghosts
--------
Loghost [ ]: chemlogger
Log Facility [local0]:

SNMP
----
Domain Description [ ]:
Domain Contact [ ]:
The SNMP agent is disabled.


simland-sc0:A> showdomain
```

CODE EXAMPLE 6-7 Board Assignment and Domain Setup Example *(Continued)*

```
System Controller 'simland-sc0':

Domain    Solaris Nodename    Domain Status            Keyswitch
--------  ------------------  -----------------------  -------------
A         -                   Powered Off              off

diag-level = default
verbosity-level = min
error-level = max
interleave-scope = within-board
interleave-mode = optimal
reboot-on-error = false
OBP.use-nvramrc? = true
OBP.auto-boot? = true
OBP.error-reset-recovery = <OBP default>

Loghost for Domain A: chemlogger
Log Facility for Domain A: local0

SNMP Agent: disabled
Domain Description:
Domain Contact:

ACL for Domain A: SB0 SB2 SB4 IB6 IB8

simland-sc0:A> disconnect
Connection closed.


System Controller 'simland-sc0':

    Type  0  for Platform Shell

    Type  1  for domain A console
    Type  2  for domain B console
    Type  3  for domain C console
    Type  4  for domain D console

    Input: 3

Connected to Domain C

Domain Shell for Domain C

simland-sc0:C> addboard SB5 IB7 IB9

simland-sc0:C> showboards

Slot     Pwr Component Type                State      Status     Domain
----     --- --------------                -----      ------     ------
/N0/SB5 Off CPU Board                      Assigned   Not tested C
/N0/IB7 Off PCI I/O board                  Assigned   Not tested C
/N0/IB9 Off PCI I/O board                  Assigned   Not tested C

simland-sc0:C> setupdomain

Domain Boot Parameters
----------------------
```

CODE EXAMPLE 6-7 Board Assignment and Domain Setup Example *(Continued)*

```
System Controller 'simland-sc0':
diag-level [default]:
verbosity-level [min]:
error-level [max]:
interleave-scope [within-board]:
interleave-mode [optimal]:
reboot-on-error [false]:
OBP.use-nvramrc? [true]:
OBP.auto-boot? [false]: true
OBP.error-reset-recovery [<OBP default>]:

Loghosts
--------
Loghost [ ]: chemlogger
Log Facility [local0]: local1

SNMP
----
Domain Description [ ]:
Domain Contact [ ]:
The SNMP agent is disabled.

simland-sc0:C> showdomain

Domain    Solaris Nodename    Domain Status            Keyswitch
--------  ------------------  -----------------------  -------------
C         -                   Powered Off              off

diag-level = default
verbosity-level = min
error-level = max
interleave-scope = within-board
interleave-mode = optimal
reboot-on-error = false
OBP.use-nvramrc? = true
OBP.auto-boot? = true
OBP.error-reset-recovery = <OBP default>

Loghost for Domain C: chemlogger
Log Facility for Domain C: local1

SNMP Agent: disabled
Domain Description:
Domain Contact:

ACL for Domain C: SB5 IB7 IB9

simland-sc0:C>
```

5. **Boot and install each domain as you would a standard Sun server.**

   Use `setkeyswitch` on each domain to boot it. Because you have two domains, you must connect to the domain shell for each one and run `setkeyswitch` from there.

CODE EXAMPLE 6-8 Power On and Domain Test Example

```
simland-sc0:A> setkeyswitch on
Powering boards on ...
Testing CPU Boards ...
{/N0/SB2/P0} Running CPU POR and Set Clocks
{/N0/SB2/P2} Running CPU POR and Set Clocks
{/N0/SB2/P1} Running CPU POR and Set Clocks
{/N0/SB2/P3} Running CPU POR and Set Clocks
{/N0/SB2/P0} @(#) lpost
5.12.5
2001/09/26 15:47
{/N0/SB2/P2} @(#) lpost
5.12.5
2001/09/26 15:47
{/N0/SB2/P1} @(#) lpost
5.12.5
2001/09/26 15:47
{/N0/SB2/P3} @(#) lpost
5.12.5
2001/09/26 15:47


Diagnostic messages omitted.....


{/N0/SB0/P0} DCB_DECOMP_OBP command succeeded
{/N0/SB0/P0} Decompress OBP done
{/N0/SB0/P0} DCB_ENTER_OBP command succeeded
{/N0/SB0/P1} DCB_ENTER_OBP command succeeded
{/N0/SB0/P2} DCB_ENTER_OBP command succeeded
{/N0/SB0/P3} DCB_ENTER_OBP command succeeded
{/N0/SB2/P0} DCB_ENTER_OBP command succeeded
{/N0/SB2/P1} DCB_ENTER_OBP command succeeded
{/N0/SB2/P2} DCB_ENTER_OBP command succeeded
{/N0/SB2/P3} DCB_ENTER_OBP command succeeded
Entering OBP ...

     .
     .
     .

{0} ok

.................

simland-sc0:C setkeyswitch on
Powering boards on ...
Testing CPU Boards ...
{/N0/SB5/P0} Running CPU POR and Set Clocks
{/N0/SB5/P2} Running CPU POR and Set Clocks
{/N0/SB5/P1} Running CPU POR and Set Clocks
{/N0/SB5/P3} Running CPU POR and Set Clocks
```

CODE EXAMPLE 6-8 Power On and Domain Test Example *(Continued)*

```
{/N0/SB5/P0} @(#) lpost
5.12.5
2001/09/26 15:47
{/N0/SB5/P2} @(#) lpost
5.12.5
2001/09/26 15:47
{/N0/SB5/P1} @(#) lpost
5.12.5
2001/09/26 15:47
{/N0/SB5/P3} @(#) lpost
5.12.5
2001/09/26 15:47


Diagnostic messages omitted.....


{/N0/SB5/P0} Decompress OBP done
{/N0/SB5/P0} DCB_ENTER_OBP command succeeded
{/N0/SB5/P1} DCB_ENTER_OBP command succeeded
{/N0/SB5/P2} DCB_ENTER_OBP command succeeded
{/N0/SB5/P3} DCB_ENTER_OBP command succeeded
Entering OBP ...

     .
     .
     .

{0} ok
```

Example 3—Database Server

You purchased a 56-CPU Sun Fire 15K system as an OLTP server for an online retailer. This system had 14 boards fully populated with CPUs and memory, as well as a Sun StorEdge 9970 array with approximately six terabytes of usable storage. You should build lots of redundancy into this configuration, making it the most complicated configuration.

1. **Lay out and connect the I/O.**

   First, because there are multiple domains, you should decide how to divide them up. As with the Sun Fire 6800 system, the best thing to do is place each domain at opposite ends of the system, so that they can expand without getting interwoven. This means that the production domain will be boards 0 to 12, and the test domain will be board 15.

Next, you should lay out the I/O for the production domain. To ensure redundancy for the boot device, you should place each LVD SCSI controller and Sun StorEdge S1 array on a different I/O board. You should also do the same for the public network. TABLE 6-8 lists the configuration so far for each of the first two I/O boards:

TABLE 6-8 I/O Layout for Boot Device and Network on I/O Boards 0 and 1

I/O Slot	Device
0 (C3V 0)	Gigabit Ethernet network card (for backups—board 0 only)
1 (C5V 0)	
2 (C3V 1)	SCSI controller (for boot device)
3 (C5V 1)	Network card (for public network)

In the Sun Fire 15k/12K systems, slots 0 and 1 share an I/O controller and slots 2 and 3 are on the second I/O controller. There are two PCI slots per controller, a 66 MHz slot and a 33-MHz slot. As shown in the preceding configuration, this means you should distribute the heavily used adapters across these pairs of slots.

The Sun StorEdge 9970 array, which you use for data storage, is a fundamentally different type of storage device than the traditional “dumb” bunch of disks. This concept is beyond the scope of this book, but you should think of the Sun StorEdge 9970 array as a server of its own. You will connect it to a switch, along with the servers that want to access the data, thus creating a storage area network (SAN). You then must set up the LUNs and logical devices (similar to subdisks) on the Sun StorEdge 9970 array.

As you decided during the design phase, you are going to use 14 FC-AL controllers on the production server. The controllers required for the Sun StorEdge 9970 array must be placed in the 66-MHz slots (slots 0 and 2 in the I/O board) for best performance. So, for each of the subsequent I/O boards, you want to place two FC-AL controllers on each in the bottom two slots (0 and 2).

You can run the first seven controllers to one SAN switch, and the next seven controllers to another SAN switch. You should skip the middle slot because you would have one board that could not be dynamically reconfigured out as it would have connections to both switches. On the Sun StorEdge 9970 array, you can do the same, running half the connections to each switch. FIGURE 6-6 shows the layout.

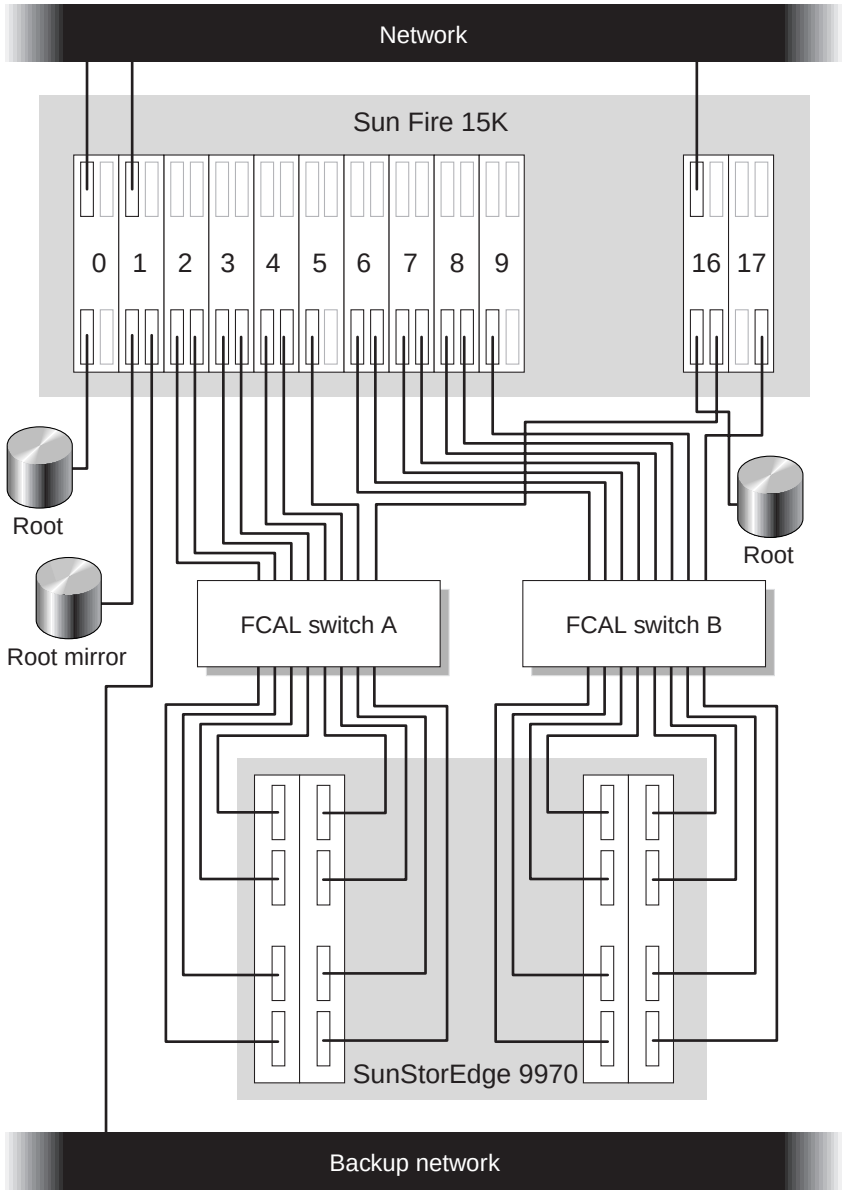


FIGURE 6-6 Sun Fire 15K I/O Layout With Sun StorEdge 9970 Array

This layout enables you to distribute the volumes across enclosures for both the boot and data storage devices, and also dual-path each enclosure across different I/O boards. The net result is that you can dynamically reconfigure out any single I/O board without having to take down the system.

I/O layout for the test domain is much simpler because no boot redundancy is required. You can use I/O boards 14 and 15 so that the I/O is located under the CPU board for the domain. The boot and network cards will go on the first board, and you will have one FC-AL controller on each board to connect to the Sun StorEdge 9970 array. TABLE 6-9 and TABLE 6-10 list the test domain I/O layouts for slots 14 and 15.

TABLE 6-9 I/O Layout for Test Domain in Slot 14

I/O Slot	Device
0 (C3V 0)	FC-AL controller (to Sun StorEdge 9970 array)
1 (C5V 0)	
2 (C3V 1)	LVD SCSI controller (for boot device)
3 (C5V 1)	Network card

TABLE 6-10 I/O Layout for Test Domain in Slot 15

I/O Slot	Device
0 (C3V 0)	FC-AL controller (to Sun StorEdge 9970 array)
1 (C5V 0)	
2 (C3V 1)	
3 (C5V 1)	

Remember that the Sun StorEdge 9970 array will be connected to the switch in a SAN configuration. You will have to make the necessary configuration changes on the Sun StorEdge 9970 array to allow the test domain to access it.

2. **Connect and power on the system.**

   This system was purchased with *redundant bulk power supplies.* Therefore, you must connect the redundant power cords to *separate* power sources at your site. When the cords are connected, turn on the breakers to power on the system.

3. **Configure each SC through a serial line and reboot the system.**

   While similar, the SC configuration on the Sun Fire 15K has some noticeable differences from the Sun Fire midrange systems. Most prominently, the Sun Fire 15K SC runs the Solaris software, so configuration for DNS, logging, and so forth is done by configuring the appropriate UNIX files (`/etc/hosts`, `/etc/resolv.conf`, `/`

`etc/syslog.conf`, and so forth). This also means that you cannot access the platform shell by choosing `0` from a menu. Instead, you must log in as a user that has platform system administration privileges.

When shipped from the factory, the SC should arrive with a loaded but unconfigured OS. The first time you power it up, it will prompt you with a series of questions similar to `setupplatform` on the Sun Fire midrange systems.[1] These questions should also take care of setting the date (like `setdate`) and password (like `password.`

Use the `smsconfig` command to configure the management network. The management network maintains internal network connections between the SCs for failover and control purposes and between the SCs and each of the domains. For the information needed to complete the management network setup, such as IP addresses, refer to the *Sun Fire 15K/12K System Site Planning Guide* that is completed prior to starting the installation process.

For complete installation procedures and examples, refer to the *System Management Services (SMS) x.x Installation Guide and Release Notes for the Sun Fire 15K/12K Systems* manual. The `man` page for `smsconfig` also has several examples.

Both SCs must be configured, one as primary and the other as a spare.

4. Connect the SC through the network and configure the domains.

After the SCs have been configured and rebooted, you are ready to create domains.

Unlike the Sun Fire midrange system SC, the Sun Fire 15K/12K SC has no platform or domain shell concepts. Instead, permissions are based on UNIX group membership. There are many different levels of platform and domain administration privileges. For complete information regarding SMS administration models refer to the SMS security chapter in the *System Management Services (SMS) x.x Administration Guide for the Sun Fire 15K/12K Systems*.

a. Set up the ACLs for each domain with the `setupplatform` command.

The default login for the platform administrator is `sms-svc`. Log in to the primary SC and set up the ACLs with the `setupplatform` command. This command differs from the Sun Fire 6800/4810/4800/*3*800 system version of the command in that it is used only to set up the ACLs.

1. This is the same set of questions you will see on any Sun server if you type `sys-unconfig` and reboot.

CODE EXAMPLE 6-9 Domain A and B ACL Population Example

```
somewhere> rlogin dboxsc0 -l sms-svc
Password:
Last login: Sat May  4 15:11:33 from dboxsc0
Sun Microsystems Inc.SunOS 5.9GenericMay 2002
dboxsc0:sms-svc:1>
dboxsc0:sms-svc> setupplatform -d A sb0 sb1 sb2 sb3 sb4 sb5 sb6 sb7 sb8 sb9 sb10 sb11
io0 io1 io2 io3 io4 io5 io6 io7 io8 io9 io10 io11

dboxsc0:sms-svc> setupplatform -d B sb17 io16 io17

dboxsc0:sms-svc> showplatform -d A

Available Component List for Domains:
=====================================
Available Component List for domain  A:
 SB0 SB1 SB2 SB3 SB4 SB5 SB6 SB7 SB8 SB9 SB10 SB11
 IO0 IO1 IO2 IO3 IO4 IO5 IO6 IO7 IO8 IO9 IO10 IO11


Domain Ethernet Addresses:
==============================
Domain ID   Domain Tag        Ethernet Address
A               -             UNKNOWN


Domain configurations:
======================
Domain ID   Domain Tag        Solaris Nodename       Domain Status
A               -                     -              Powered Off

dboxsc0:sms-svc> showplatform -d B

Available Component List for Domains:
=====================================
Available Component List for domain B:
 SB17
 IO16 IO17


Domain Ethernet Addresses:
==============================
Domain ID   Domain Tag        Ethernet Address
B               -             UNKNOWN


Domain configurations:
======================
Domain ID   Domain Tag        Solaris Nodename       Domain Status
B               -                     -              Powered Off
```

b. Configure a domain administrator account with the `smsconfig` command.

Just as you log in as `sms-svc` to administer the platform, you must log in as a user who has permission to the domain you want to administer. By default, `sms-svc` administers the platform. Unless it is added to the domain administrator group for the domain, it cannot be used to configure the domain. For this example, `sms-svc` is added to the appropriate groups; however, *any* user added to the group would have administration privileges. You need not create special

user accounts to administer the platform and domains. If you do not have security concerns, you can add `sms-svc` to the domain administrator group for each domain. In that way one account can do it all. Check the `man` page for the `smsconfig` command for several examples.

Run `smsconfig` as root to assign domain administration privileges.

CODE EXAMPLE 6-10 `smsconfig` Command Example

```
dboxsc0# /opt/SUNWSMS/bin/smsconfig -a -u sms-svc -G admn  A
All privileges to domain A have been applied.

dboxsc0# /opt/SUNWSMS/bin/smsconfig -a -u sms-svc -G admn  B
All privileges to domain B have been applied.
```

c. Assign system boards to domains with `addboard`.

As with the midrange systems, the domains are already *created*. You just need to assign the system boards to them with the `addboard` command.

CODE EXAMPLE 6-11 `addboard` Command Example

```
dboxsc0:sms-svc> addboard -d A -c assign sb0 sb1 sb2 sb3 sb4 sb5 sb6 sb7 sb8 sb9 sb10
sb11 io0 io1 io2 io3 io4 io5 io6 io7 io8 io9 io10 io11
SB0 assigned to domain: A
SB1 assigned to domain: A
SB2 assigned to domain: A
SB3 assigned to domain: A
SB4 assigned to domain: A
SB5 assigned to domain: A
SB6 assigned to domain: A
SB7 assigned to domain: A
SB8 assigned to domain: A
SB9 assigned to domain: A
SB10 assigned to domain: A
SB11 assigned to domain: A
IO0 assigned to domain: A
IO2 assigned to domain: A
IO3 assigned to domain: A
IO4 assigned to domain: A
IO5 assigned to domain: A
IO6 assigned to domain: A
IO7 assigned to domain: A
IO8 assigned to domain: A
IO9 assigned to domain: A
IO10 assigned to domain: A
IO11 assigned to domain: A

dboxsc0:sms-svc> addboard -d B -c assign sb17 io16 io17
SB12 assigned to domain: B
IO12 assigned to domain: B
IO12 assigned to domain: B
dboxsc0:sms-svc> showboards
Retrieving board information. Please wait.
......
Location    Pwr    Type of Board   Board Status  Test Status   Domain
```

CODE EXAMPLE 6-11 `addboard` Command Example *(Continued)*

```
--------    ---    -------------   ------------  -----------   ------
SB0         Off    CPU             Assigned      Unknown       A
SB1         Off    CPU             Assigned      Unknown       A
SB2         Off    CPU             Assigned      Unknown       A
SB3         Off    CPU             Assigned      Unknown       A
SB4         Off    CPU             Assigned      Unknown       A
SB5         Off    CPU             Assigned      Unknown       A
SB6         Off    CPU             Assigned      Unknown       A
SB7         Off    CPU             Assigned      Unknown       A
SB8         Off    CPU             Assigned      Unknown       A
SB9         Off    CPU             Assigned      Unknown       A
SB10        Off    CPU             Assigned      Unknown       A
SB11        Off    CPU             Assigned      Unknown       A
SB12         -     Empty Slot      Available         -         Isolated
SB13         -     Empty Slot      Available         -         Isolated
SB14         -     Empty Slot      Available         -         Isolated
SB15         -     Empty Slot      Available         -         Isolated
SB16         -     Empty Slot      Available         -         Isolated
SB17        Off    CPU             Assigned      Unknown       B
IO0         Off    HPCI            Assigned      Unknown       A
IO1         Off    HPCI            Assigned      Unknown       A
IO2         Off    HPCI            Assigned      Unknown       A
IO3         Off    HPCI            Assigned      Unknown       A
IO4         Off    HPCI            Assigned      Unknown       A
IO5         Off    HPCI            Assigned      Unknown       A
IO6         Off    HPCI            Assigned      Unknown       A
IO7         Off    HPCI            Assigned      Unknown       A
IO8         Off    HPCI            Assigned      Unknown       A
IO9         Off    HPCI            Assigned      Unknown       A
IO10        Off    HPCI            Assigned      Unknown       A
IO11        Off    HPCI            Assigned      Unknown       A
IO12         -     Empty Slot      Available         -         Isolated
IO13         -     Empty Slot      Available         -         Isolated
IO14         -     Empty Slot      Available         -         Isolated
IO15         -     Empty Slot      Available         -         Isolated
IO16        Off    HPCI            Assigned      Unknown       B
IO17        Off    HPCI            Assigned      Unknown       B
```

d. Add a "tag" to the domain with the `addtag` command (optional).

One added feature of the Sun Fire 15K/12K systems that the midrange systems do not have is the ability to custom-name domains by using the `addtag` command. So, instead of administering domains A and B, you can administer *dbserver* and *testdom*.

CODE EXAMPLE 6-12 `addtag` Command Example

```
dboxsc0:sms-svc> addtag -d A dbserver

dboxsc0:sms-svc> addtag -d B testdom

dboxsc0:sms-svc:168> showboards
Retrieving board information. Please wait.
.......
Location    Pwr    Type of Board   Board Status  Test Status   Domain
--------    ---    -------------   ------------  -----------   ------
SB0         Off    CPU             Assigned      Unknown       dbserver
```

CODE EXAMPLE 6-12 `addtag` Command *(Continued)*Example

```
SB1         Off    CPU             Assigned     Unknown       dbserver
SB2         Off    CPU             Assigned     Unknown       dbserver
SB3         Off    CPU             Assigned     Unknown       dbserver
SB4         Off    CPU             Assigned     Unknown       dbserver
SB5         Off    CPU             Assigned     Unknown       dbserver
SB6         Off    CPU             Assigned     Unknown       dbserver
SB7         Off    CPU             Assigned     Unknown       dbserver
SB8         Off    CPU             Assigned     Unknown       dbserver
SB9         Off    CPU             Assigned     Unknown       dbserver
SB10        Off    CPU             Assigned     Unknown       dbserver
SB11        Off    CPU             Assigned     Unknown       dbserver
SB12         -     Empty Slot      Available        -         Isolated
SB13         -     Empty Slot      Available        -         Isolated
SB14         -     Empty Slot      Available        -         Isolated
SB15         -     Empty Slot      Available        -         Isolated
SB16         -     Empty Slot      Available        -         Isolated
SB17        Off    CPU             Assigned     Unknown       testdom
IO0         Off    HPCI            Assigned     Unknown       dbserver
IO1         Off    HPCI            Available    Unknown       dbserver
IO2         Off    HPCI            Assigned     Unknown       dbserver
IO3         Off    HPCI            Assigned     Unknown       dbserver
IO4         Off    HPCI            Assigned     Unknown       dbserver
IO5         Off    HPCI            Assigned     Unknown       dbserver
IO6         Off    HPCI            Assigned     Unknown       dbserver
IO7         Off    HPCI            Assigned     Unknown       dbserver
IO8         Off    HPCI            Assigned     Unknown       dbserver
IO9         Off    HPCI            Assigned     Unknown       dbserver
IO10        Off    HPCI            Assigned     Unknown       dbserver
IO11        Off    HPCI            Assigned     Unknown       dbserver
IO12         -     Empty Slot      Available        -         Isolated
IO13         -     Empty Slot      Available        -         Isolated
IO14         -     Empty Slot      Available        -         Isolated
IO15         -     Empty Slot      Available        -         Isolated
IO16        Off    HPCI            Assigned     Unknown       testdom
IO17        Off    HPCI            Assigned     Unknown       testdom
```

5. Boot and install each domain as you would a standard Sun server.

One thing is the same among all Sun Fire systems. To boot a domain, you use `setkeyswitch` command.

CODE EXAMPLE 6-13 `setkeyswitch` Command Boot Example

```
dboxsc0:sms-svc:187> setkeyswitch -d A on
Powering on: CSB at CS0
Powering on: CSB at CS1
Powering on: EXB at EX0
Powering on: HPCI at IO0
Powering on: CPU at SB0
Powering on: EXB at EX1
Powering on: HPCI at IO1
Powering on: CPU at SB1
Powering on: EXB at EX2
Powering on: HPCI at IO2
Powering on: CPU at SB2
Powering on: EXB at EX3
```

CODE EXAMPLE 6-13 `setkeyswitch` Command Boot Example *(Continued)*

```
Powering on: HPCI at IO3
Powering on: CPU at SB3
Powering on: EXB at EX4
Powering on: HPCI at IO4
Powering on: CPU at SB4
Powering on: EXB at EX5
Powering on: HPCI at IO5
Powering on: CPU at SB5
Powering on: EXB at EX6
Powering on: HPCI at IO6
Powering on: CPU at SB6
Powering on: EXB at EX7
Powering on: HPCI at IO7
Powering on: CPU at SB7
Powering on: EXB at EX8
Powering on: HPCI at IO8
Powering on: CPU at SB8
Powering on: EXB at EX9
Powering on: HPCI at IO9
Powering on: EXB at EX10
Powering on: HPCI at IO10
Powering on: CPU at SB10
Powering on: EXB at EX11
Powering on: HPCI at IO11
Powering on: CPU at SB11

Significant contents of .postrc (platform)
        /etc/opt/SUNWSMS/SMS1.3/config/platform/.postrc:
# ident "@(#)postrc     1.1     01/04/02 SMI"

Reading domain blacklist file /etc/opt/SUNWSMS/config/A/blacklist ...
# ident "@(#)blacklist  1.1     01/04/02 SMI"
Reading platform blacklist file /etc/opt/SUNWSMS/config/platform/blacklist ...
# ident "@(#)blacklist  1.1     01/04/02 SMI"

Diagnostic messages omitted.....

stage final_config: Final configuration...
Creating CPU SRAM handoff structures...
Creating GDCD IOSRAM handoff structures in Slot IO0...
Writing domain information to PCD...
Configured in 333 with 48 procs, 192.000 GBytes, 19 IO adapters.
Interconnect frequency is 149.984 MHz, Measured.
Golden sram is on Slot IO0.

dboxsc0:sms-svc:188> console -d A
Trying to connect...
Connected to Domain Server.
Your console is in exclusive mode now.

{160} ok

.............

dboxsc0:sms-svc:189> setkeyswitch -d B on
Powering on: EXB at EX16
Powering on: HPCI at IO16
Powering on: EXB at EX17
```

CODE EXAMPLE 6-13 `setkeyswitch` Command Boot Example *(Continued)*

```
Powering on: HPCI at IO17
Powering on: CPU at SB17

Significant contents of .postrc (platform)
        /etc/opt/SUNWSMS/SMS1.3/config/platform/.postrc:
# ident "@(#)postrc     1.1     01/04/02 SMI"

Reading domain blacklist file /etc/opt/SUNWSMS/config/B/blacklist ...
# ident "@(#)blacklist  1.1     01/04/02 SMI"
Reading platform blacklist file /etc/opt/SUNWSMS/config/platform/blacklist ...
# ident "@(#)blacklist  1.1     01/04/02 SMI"

Diagnostic messages omitted.....

stage final_config: Final configuration...
Creating CPU SRAM handoff structures...
Creating GDCD IOSRAM handoff structures in Slot IO0...
Writing domain information to PCD...
Configured in 333 with 4 procs, 16.000 GBytes, 4 IO adapters.
Interconnect frequency is 149.984 MHz, Measured.
Golden sram is on Slot IO16.

dboxsc0:sms-svc:188> console -d B
Trying to connect...
Connected to Domain Server.
Your console is in exclusive mode now.

{544} ok
```

Conclusion

At this point, the systems are configured and functional. What remains is for you to install the appropriate version of Solaris OE for your application, as well as any other packages that you may need. Ongoing maintenance should only require fairly standard system administration practices—monitoring logs, periodically applying patches, and upgrading the system as your needs change.

Also, you should take some time to analyze your newly-installed system once it is placed into active use. This can help you evaluate your design assumptions, and use this knowledge to improve your design. You can use the tools and processes we outlined in Chapter 3 to do this analysis.

Reading this book and applying the tools and methods provided herein should have given you the following:

- Key features of the Sun Fire product line
- A process to follow to determine your needs
- Design considerations and rules of thumb

- Methods for maximizing performance and RAS properties
- Step-by-step system analysis procedures
- Methods for planning for growth and upgrading systems
- System configuration guidelines and steps
- Real-world system examples

Using these tools and knowledge to design and configure your Sun Fire systems will help you design systems that more accurately meet your application support requirements. Also, you can verify your design assumptions by analyzing the newly-installed system and using that knowledge to improve your design next time. Over time, this should lead to better-performing, more reliable systems with lower cost of ownership.

Abbreviations, Acronyms, and Glossary

A

ACL Access control list. In order to assign a system board to a domain with the `addboard` command, the board name must be listed in the Access Control List (ACL). The ACL is checked when a domain makes an `addboard` or a `testboard` request on that board.

Note – May be referred to as the available component list in some documents.

ASIC Application specific integrated circuit. ASICs are large-scale integrated circuit chips that perform specific tasks as opposed to general purpose ICs such as CPUs and memory chips. The Sun Fire architecture contains several ASICs in its design.

B

blacklist The blacklist specifies Sun Fire system components that are not to be used or configured into the system. The blacklist is maintained using the `enablecomponent` and `disablecomponent` commands.

C

cluster A cooperative collection of interconnected computer systems, each running a separate OS image, utilized as a single, unified computing resource.

Compact PCI, cPCI The compact PCI card is designed to be hot-swappable. Used in the Sun Fire 6800/4810/4800/3800 systems only.

CPU Central processing unit.

CPU/Memory board The CPU/Memory board is used on all Sun Fire 15K/12K and Sun Fire 6800/4810/4800/3800 systems. The CPU/Memory board can be configured with two or four CPUs and with 2GB to 32GB of memory.

D

DIMM Dual in-line memory module. A small printed circuit card containing memory chips and some support logic.

domain administrator The domain administrator has privileges to manage a particular domain. The domain administrator can control power, domain configuration and the virtual keyswitch position.

domain shell With the domain shell, you have access to system controller commands that you need to perform on a domain. There are up to four domain shells (A through D). The domain shell prompt is *schostname*:A> (or B>, C>, or D>. The **domain shell** applies to the Sun Fire 6800/4810/4800/3800 systems only.

domain A set of one or more system boards that act as a separate system capable of booting the operating system and running independently of other domains. Domains do not depend on each other and do not interact with each other.

domain console A virtual console that allows access to the system console. Similar to the RS232 serial console on previous Sun systems.

DR Dynamic reconfiguration. Enables you to logically attach and detach system boards to and from the operating system without causing a system interrupt. DR is used in conjunction with hot-swap, which is the process of physically removing or inserting a system board. DR can be used to add a new system board, reinstall a repaired system board, or modify the domain configuration on a Sun Fire system.

E

Ecache External cache. Fast SRAM memory local to each processor that is used to cache data and instructions.

ECC Error correction code. ECC detects and corrects single bit data errors. ECC also detects and reports multiple bit errors.

environmental monitoring All systems have a large number of sensors that monitor temperature, voltage, and current. The system controller polls devices in a timely manner and makes the environmental data available. The system controller will shut down various components to prevent damage.

Expander board Interface board that connects the CPU/Memory board and I/O Module to the centerplane. This board is only used in Sun Fire 15K/12K systems. Together the CPU/Memory board, I/O Module and Expander board are known as a *board set*.

F

FRU Field-replaceable unit.

H

High-end Systems Sun Fire 15K/12K systems.

Hot-swap PCI, HsPCI The Sun Fire 15K/12K systems use a PCI Card adapter that allows PCI Cards to be hot swapped.

HPOST Sun Fire 15K/12K power on self test (POST) control application. Runs on the Sun Fire 15K/12K System Controller. Configures and tests the hardware and prepares it to run the Solaris OE

I

IB See **I/O Module.** Also referred to as an *I/O Boat* in some documents. Used only on the Sun Fire 6800/4810/4800/3800 systems.

I/O Module A system assembly that contains slots for PCI controllers and the necessary circuitry to interface with the system interconnect. Can be configured with four to eight PCI Cards depending on the Sun Fire Model.

Note – Also referred to as an I/O board in some Sun Fire 15K/12K documentation.

K

keyswitch See virtual keyswitch.

M

MaxCPU Dual CPU board that can be configured in place of an I/O module. Used in Sun Fire 15K/12K systems only.

Mid-range Systems The Sun Fire 6800/4810/4800/3800 systems. Also referred to as midframe systems in some documents.

O

OE Operating environment.

P

partition A partition is a group of Repeater boards that are used together to provide communication between CPU/Memory boards and I/O assemblies in the same domain. You can configure the platform with one or two partitions using the system controller `setupplatform` command. Configuring domains in separate partitions offers a higher level of isolation than creating domains in the same partition. Partitions are supported on the Sun Fire 6800/4810/4800/3800 systems only.

Note – Also referred to as *segments* in some documents. Partition and segment have the same meaning.

PCI Controller Industry standard peripheral controller used in the Sun Fire systems.

platform administrator The platform administrator manages hardware resources across domains.

POST Power-on self test. Diagnostics that test the hardware prior to booting a domain.

R

Repeater board A crossbar switch that connects multiple CPU/Memory boards and I/O assemblies. Having the required number of Repeater boards is mandatory for operation. There are Repeater boards in each mid-range system except for the Sun Fire 3800. In the Sun Fire 3800 system, the equivalent of two Repeater boards is integrated into the active centerplane. Repeater boards are not used in the Sun Fire 15K/12K systems.

RTS Redundant transfer switch.

RTU Redundant transfer unit.

S

SAN Storage area network. A storage interconnect method that allows one or more storage devices to be connected to one or more servers in a network-like configuration.

SB System Board. See **CPU/Memory board.**

SC System controller.

SCPER System controller peripheral board. This board contains a 4mm tape drive, a DVD drive and the boot disk for the SC. The SCPER is used only in the Sun Fire 15K/12K systems.

ScApp Application that runs on the Sun Fire 6800/4810/4800/3800 System Controller that provides control/monitoring functions.

segment See **partition**.

SMS System management services software. The software that runs on the Sun Fire 15K/12K SC and provides control/monitoring functions for the Sun Fire 15K/12K platform and domains.

V

virtual keyswitch The SC provides a virtual keyswitch for each domain which controls the bring-up process for the domain. The `setkeyswitch` command controls the position of the virtual keyswitch for each domain. Possible positions are: `on`, `off`, `standby`, `diag`, and `secure`.

Bibliography

JHOL01 James Hsieh, “Sun Fire™ Midframe Server Configuration Best Practices,” Sun BluePrints Online, September 2001, `http://www.sun.com/blueprints`

JHOL02 James Hsieh, “Sun Fire Midframe Server Best Practices for Administration,” Sun BluePrints OnLine, October 2001, `http://www.sun.com/blueprints`

AFOA01 Æleen Frisch, *Essential System Administration, Second Edition,* O’Reilly & Associates, Inc. ISBN 1-56592-127-5

SGGSOA02 Simson Garfinkel, Gene Spafford, *Practical UNIX & Internet Security, 2nd Edition,* O’Reilly & Associates, Inc. ISBN 1-56592-148-8

Sun Fire 15K Architecture http://webhome.eng/alanc/sunfire/arch/sf15Karch.pdf

Online documents:

http://www.sun.com/processors/UltraSPARC-III/details.html

http://www.sun.com/processors/UltraSPARC-III/USIIITech.html

http://www.sun.com/processors/UltraSPARC-III/USIIICuoverview.pdf

http://www.sun.com/sun-on-net/itworld/UIR970501perf.html

http://www.sun.com/sun-on-net/itworld/UIR980801perf.html

http://www.sun.com/sun-on-net/itworld/UIR981001perf.html

http://www.sun.com/sun-on-net/itworld/UIR960401perf.html

http://www.sun.com/sun-on-net/itworld/UIR970901perf.html

Index

S

U

V

W