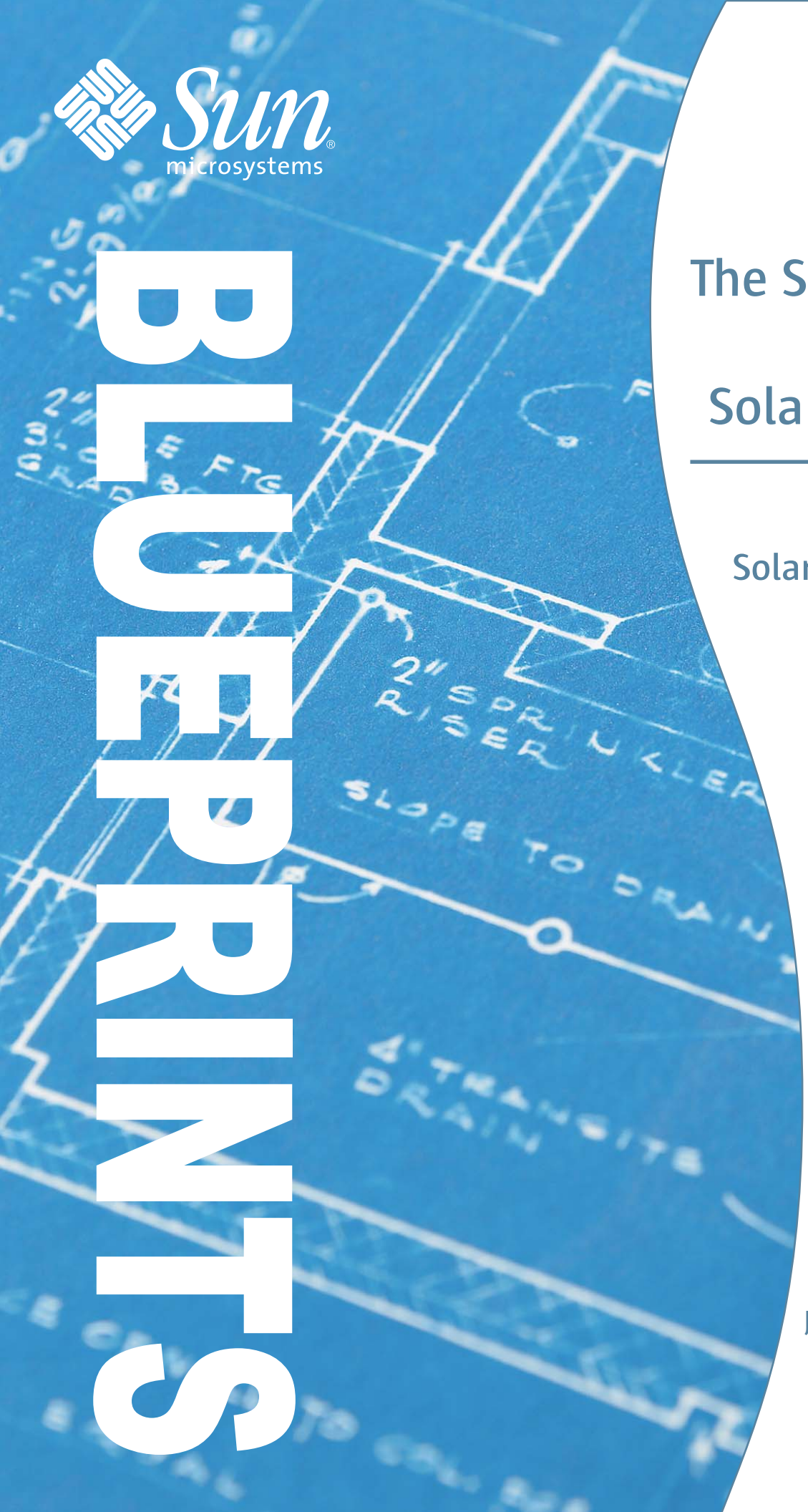

The Sun BluePrints™ Guide to Solaris™ Containers

Virtualization in the Solaris Operating System

Harry J. Foxwell
Menno Lageman
Joost Pronk van Hoogeveen
Isaac Rozenfeld
Sreekanth Setty
Jeff Victor

Part No. 820-0001-10
Revision 1.0, 10/11/06

Please
Recycle

Contents

Chapter 1: Introduction. 1
Solaris Containers Technology. 2
Scope 2

Chapter 2: Resource Management Concepts. 3
Resource Containment 3
Workload Classification 4
Resource Controls 6
Differentiated Services 6
Virtualization and Containment 7
Application Isolation 8

Chapter 3: An In-Depth Look at Containment and Virtualization. 11
Containing Services 12
Approaches to Containment 13
Software-Based Containment 15
Solaris Containers. 16
Containment and Virtualization Trade-Offs. 18
Guidelines for Deploying Solaris Containers 20

Chapter 4: Managing Workloads 23
Projects 24
Using Projects to Define Workloads 24
The Project Database 25
Commands 26
Extended Accounting 26
Commands 27

The Fair Share Scheduler 27
CPU Shares 28
CPU Shares Configuration 29
Resource Controls 30
Administering Resource Controls 30
Available Resource Controls 31
Determining Thresholds 33
Commands 33
Managing Workloads — An Example 33
Requirements 34
Defining the Projects 34
Installing Oracle and Creating the Databases 37
Running Oracle Instances in Different Projects 38
Controlling CPU Consumption 40
Using Extended Accounting 44

Chapter 5: Managing Resources 49

Processor Sets 49
Resource Pools 50
Binding Processes To Pools 51
Fair Share Scheduler and Processor Sets 52
Dynamic Resource Pools 52
Automated Resource Allocation 53
Configuration Objectives 54
Monitoring Resource Pools 55
Commands 55
Resource Pools — An Example 55
Creating a Pool 58
Binding to a Pool 61
Transferring CPUs 62
Adapting to Load 62
Saving the Dynamic Configuration 64

Chapter 6: Isolating Applications 65

Zones Overview 66
Administering Zones 67
Zone Configuration 68

Installing Zones 68
Virtual Platform Management 69
Zone Login 69
Commands 70
Zone Administration 71
File Systems 71
Resource Management 72
Resource Pools 72
Extended Accounting 72
Fair Share Scheduler and Zones 73
Resource Controls 73
Using Zones — An Example 73
Requirements 74
Preparation 74
Creating the First Zone 75
Creating the Second Zone 79
Controlling CPU Consumption of Zones 83
Controlling CPU Consumption Inside Zones 85
Halting Zones 87

Chapter 7: Creating Solaris Containers 89

Container Construction 91
Creating the Pools 91
Binding Zones to Pools 94
Creating Development Zones 94
Creating Development Users and Projects 96
Verifying the Configuration 97

Chapter 8: Integrating Solaris Containers into the Environment 101

Storage Configuration 102
File System Structure 102
File Systems versus Raw Devices 103
Selecting Direct Attached Storage, Network Attached Storage, and Storage Area Networks 103
File System Types 103
General File System Considerations 109
Ability to Mount Read-Write versus Read-Only 109
Backup and Restore 112

Tape Backup . . . 113
Disk Snapshot . . . 115
Network Configuration . . . 115
Dynamic Host Configuration Protocol . . . 116
Changing the IP Address for a Zone . . . 116
Routing . . . 117
Firewalls and Filters . . . 118
Internet Protocol Multi-Pathing and Sun Trunking . . . 118
Subnet Masks . . . 118
Printing . . . 119
Security Risks . . . 119
Resource Management . . . 119
Resource Capping . . . 120
Resource Management Using Kernel Parameters . . . 123
Provisioning and Installation . . . 125
Sparse versus Whole Root Models . . . 126
Package Management and Solaris Containers Technology . . . 126
Patch Management and Solaris Containers Technology . . . 127
Flash Archives . . . 127
Security . . . 127
Process Rights Management . . . 127
Auditing and Access Control . . . 128
Namespace Isolation and Naming Services . . . 129

Chapter 9: Managing the Environment . . . 131

Sun Management Center Software . . . 131
Solaris Container Manager Software . . . 131
Consolidation Tool for Sun Fire Servers . . . 132
Predictive Self-Healing Technology . . . 132
Solaris Containers and Predictive Self-Healing Technology . . . 133
Solaris Service Manager Software . . . 134
Working with SMF . . . 137
WorkingTogether . . . 144

Chapter 10: Troubleshooting . . . 147

Methods to Access a Troubled Zone . . . 147
Telnet and Shells . . . 147

User Login with the zlogin Command 148
Zone Console Login 149
Safe-Mode Login 149
Boot Single User. 150
Network Troubleshooting. 150

Chapter 11: Putting It All Together—Deploying Sun Java Enterprise System 2005-Q4 on the Sun Fire T2000 Server Using Solaris Containers 151

Deploying Java ES 2005-Q4 on a Sun Fire T2000 Server Using Solaris Zones 152
Configuring Solaris Zones 154
Monitoring and Managing Zones. 157
Overview of Deploying Sun Java Enterprise System 2005-Q4 on Solaris Zones 158
Deploying Directory Server in Zone-1 158
Deploying Access Manager in Zone-2. 160
Deploying Portal Server in Zone-3 162
Preparing Directory Server for Messaging and Calendar Server Installations 164
Configuring Delegated Admin and Communications CLI for Creating Users 165
Deploying the Messaging Server in Zone-4. 167
Deploying Calendar Server in Zone-5 170
Installing Communications Express and Messenger Express in Zone-6 171
Deploying Messenger Express in Zone-6 173
Deploying Communications Express in Zone-6 175
Configuring Single Sign-On for Communications Services Products. 178
Tuning Sun Java Enterprise System Software for Improved Performance 180
Tuning the Directory Server. 180
Tuning a Web Container 181
Tuning Access Manager. 182
Tuning the Portal Server 184
Tuning the Messaging Server. 185
Tuning the Calendar Server 186
Tuning Communications Express 186
Tuning the Solaris Operating System 186
Sun Java Enterprise System Performance Test Case 187
Overview of the JESMark Benchmark 188
Calendar Workload 188
Logical Architecture. 189

Testing Scenario 190
Performance Results 190

Chapter 12: About the Authors 193

Chapter 13: Glossary 195

Chapter 14: References 199

Chapter 15: Index 203

Acknowledgments

A great deal of effort goes into any book. While it is impossible to name everyone who contributed to, or influenced, the content of this book, it is important to thank those individuals who most directly impacted this significant body of work. In particular, Glenn Brunette, James Carlson, David Collier-Brown, David Comay, Mark de Groot, Mark Huff, Paul Kraus, Holger Leister, John Meyer, Dan Price, and Ray Voight provided keen insight and careful review of the material presented.

In addition, the authors would like to recognize the several groups at Sun, including the Performance, Availability, and Architecture Engineering (PAE) group, Portal Server Performance team, Messaging Server QA team, Java Performance group, Ireland Performance group, Information Products group (IPG), Communications Marketing team, and Solutions Deployment Engineering (SDE) group for their contributions to this book.

Lastly, a special thanks is due to Margaret Bierman and Kemer Thomson, for without them, this book would not exist.

Preface

How This Book is Organized

- Chapter 1, “Introduction” describes the challenges organizations are facing as data centers become critcally important, and introduces Solairs Containers technology as an important tool that can help increase operational efficiency while reducing costs.
- Chapter 2, “Resource Management Concepts,” provides an introduction to the resource management tools provided as part of Solaris Containers technology in the Solaris Operating System.
- Chapter 3, “An In-Depth Look at Containment and Virtualization,” provides an overview of hardware- and software-based containment and technologies, discusses consolidation and virtualization trade-offs, and presents guidelines for deploying Solaris Containers technology.
- Chapter 4, “Managing Workloads,” explains how to use the resource management tools in the Solaris OS to manage workloads and ensure resource utilization requirements are met.
- Chapter 5, “Managing Resources,” explains how to partition available resources.
- Chapter 6, “Isolating Applications,” explains how Solaris Zones technology can be used to isolate applications from one another.
- Chapter 7, “Creating Solaris Containers,” puts all the pieces together and explains how resource management and application isolation techniques can be used in combination to create Solaris Containers on a system.
- Chapter 8, “Integrating Solaris Containers into the Environment,” provides guidelines and suggestions for designing system configurations using Solaris Containers technology.
- Chapter 9, “Managing the Environment,” identifies several tools that can help ease management and ensure reliability and availability requirements are met.
- Chapter 10, “Troubleshooting,” provides an overview of items to consider should a virtual environment not behave as expected.

- Chapter 11, “Putting It All Together—Consolidating Enterprise Applications with Solaris Containers,” discusses how to consolidate enterpise applications onto a single Sun Fire™ T2000 server using Sun Java Enterprise System software and Solaris Containers technology.

Typographic Conventions

TABLE P-1 describes the typographic conventions used in this book.

TABLE P-1 Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Use `ls -a` to list all files. `% You have mail`.
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	The names of files and directories	Edit your */etc/system* file.
AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this.
AaBbCc123	Command-line placeholder text; replace with a real name or value	To delete a file, type `rm` *filename*.

TABLE P-2 shows the typographic conventions that are used when describing individual software options and commands.

TABLE P-2 Typographic Notations for Options and Commands

Notation	Meaning	Example: Text or Instance
[ ]	Square brackets contain arguments that are optional.	`–progress[off]` `–progress off, –progress help[subcommand]`
{ }	Curly brackets contain a set of choices for a required option or command.	`–filter {id| id:id}` `step {major|minor}`
:	The colon, like the comma, is sometimes used to separate arguments or show a range of arguments.	`–filter [id:id]`
|	The “pipe” or “bar” symbol separates arguments, either of which may be specified.	`–filter {id|id:id}` `autosave {on|off}`
...	The ellipsis indicates omission in a series.	`–filter` *id1*`[,...`*idn*`]` `–filter 5000,5005`

Shell Prompts

TABLE P-3 shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P-3 Shell Prompts

Shell	Prompt
C shell	`%`
Bourne shell and Korn shell	`$`
C shell, Bourne shell, and Korn shell superuser	`#`

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. Readers living in the United States, Canada, Europe, or Japan, can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` Web site enables users to access Sun technical documentation online. Users can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is *http://docs.sun.com/*

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: *http://www.sun.com/blueprints/online.html*

1

Introduction

In recent years, the nature of computing has changed in fundamental ways. The explosive growth of corporate intranets and the Internet has created new and challenging demands. As the number of users and devices accessing services over the network grows, IT organizations are being forced to rethink how they create, manage, extend, and ultimately deliver information technology (IT) services with greater functionality and reduced cost. At the same time, these advancements create massive opportunities for operational efficiency, cost reductions, and innovations in service and functionality.

IT organizations are focused on obtaining additional availability or scalability in the most efficient way possible, with users taking for granted that IT services will deliver the performance and predictability they need. Traditionally, companies deployed large-scale information systems consisting of pools of redundant servers in a distributed architecture in an effort to support rapid growth and adapt to rapidly changing business demands. However, as systems are replicated throughout IT infrastructures to give them greater resiliency and throughput, the result is often a complicated network of systems that are often over- or under-utilized, as well as costly and difficult to manage.

Deployment models requiring each application to run on its own system are costly to build and maintain—and the escalating costs associated with managing vast networks of servers and software components are forcing organizations to find new ways to reduce IT infrastructure costs and better manage end user service levels. To help this effort, many organizations are beginning to turn to consoliation and virtualization techniques that can help deploy complex applications while raising resource utilization rates. While server consolidation and virtualization techniques help by enabling systems within data centers to be visualized and managed as interconnected computing resources rather than as individual systems, better ways must be found to provision applications and ensure shared resources are not compromised. For these resource management techniques to be effective, companies must be able to manage applications effectively.

Solaris Containers Technology

With the release of the Solaris™ 10 Operating System (OS), Sun has taken a big step towards delivering functionality that can help address many of the challenges IT organizations face as they look to consolidate and virtualize the environment. Sun's next advancement in server virtualization is a concept called Solaris Containers technology.

Solaris Containers consist of a set of technologies that help system administrators increase resource utilization by consolidating multiple applications onto a single system. With Solaris Containers technology, administrators can specify the percentage of physical system resources each application receives, as well as isolate each application in its own virtual environment with its own hostname, IP address(es), users, file system, and more. By providing isolation between software applications or services using flexible, software-defined boundaries, Solaris Containers create an execution environment within a single instance of the Solaris OS and provide:

- *Full resource containment and control* for more predictable service levels
- *Software fault isolation* to minimize fault propagation and unplanned downtime
- *Security isolation* to prevent unauthorized access as well as unintentional intrusions

The primary benefits of Solaris Containers are:

- *Reduced management costs* through server consolidation and a reduced number of operating system instances
- *Increased resource utilization* with dynamic resource reallocation among Containers
- *Increased service availability* by minimizing fault propagation and security violations between applications
- *Increased flexibility* because software-based Containers can be dynamically created and reconfigured
- *Increased accuracy and flexibility of accounting* based on workloads rather than systems or processes

Scope

This book provides an overview of the resource management concepts and technologies that comprise Solaris Containers, and explains how to create, use, and integrate Solaris Containers within a system and infrastructure. Emphasis is placed on explaining each concept and providing detailed examples that can be used to create more effective environments and affect better resource utilization.

2

Resource Management Concepts

While server consolidation provides the opportunity to lower costs by reducing the hardware and system administration required to run a set of applications, the problem of provisioning applications with the appropriate resources on a shared system can be a difficult task. The ability to minimize cross-workload performance compromises, combined with facilities that monitor resource usage and utilization, are collectively referred to as *resource management*. Resource management facilities permit administrators to modify the default behavior of the system with respect to different workloads.

In a consolidation effort, resource management has three roles:

- A method to classify workloads, ensuring the system knows which processes belong to which workload
- The ability to measure the workload and quantify how many system resources the workload is really using
- The ability to control the workload, ensuring it does not interfere with other workloads, yet gets enough system resources to meet its service level requirements

Resource Containment

No two organizations have the same kind of workload or use system resources in the same way. Some may utilize batch compute servers and application or database servers, while others may employ complex timesharing systems. Regardless of environment, a vast amount of potential computing capacity often remains untapped. Users are continually searching for more computing resources to help solve problems, resulting in systems that are alternately over-loaded or under-utilized. At the same time, a lack of trust hinders the sharing of computing resources—users are afraid their computing resources will be overtaken by other applications. System administrators must find ways to gain control and establish isolation mechanisms in order to improve resource utilization. With a finer granularity of control, administrators can ensure all workloads have access to an appropriate amount of resources, and that no workload consumes the entire system.

To address these issues, administrators need to prioritize application services and control their resource usage. Today, system administrators can affect how applications impact the system, including defining memory usage and determining disk storage location. CPUs, on the other hand, remain the most sought after, and the least controlled, resource.

Resource containment enables administrators to establish resource boundaries and create isolated computing environments. In the Solaris OS, administrators can control system resources through the creation of a *resource pool*, a collection of resources, known as *resource sets*, such as CPUs, physical memory, or network I/O bandwidth, that are reserved for exclusive use by an application or set of applications. Resource pools enable system administrators to partition a system into a set of smaller virtual environments, each providing resources for a fixed workload consisting of one or more applications. These partitions provide fixed boundaries between workloads, ensuring each has access to a consistent set of resources regardless of resource usage on the rest of the machine. As a result, administrators can separate workloads to eliminate the competition for resources, helping achieve predictable application and system performance, as well as user trust.

Resource pools have the following characteristics:

- A resource pool consists of one or more physical resource sets. For example, a CPU resource set consists of one or more CPUs that will execute processes assigned to the resource pool. A physical memory set consists of a specific amount of physical memory reserved for the process(es) bound to the set.
- A resource pool may include different types of resource sets, such as CPU and memory. Only one set of each resource type is permitted in a given resource pool.
- Multiple resource pools can share resources, enabling the system to be partitioned among applications.
- If a process is bound to a resource pool that includes only a CPU resource set, the process runs within the confines of the specified CPUs, but utilizes a general pool of physical memory that is assigned to all processes not assigned to a specific memory set. Similarly, if a process is bound to a pool that includes only a memory resource set, the process runs within the confines of the specified memory, but utilizes a general pool of CPUs set aside for processes not assigned to a specific CPU set.

Workload Classification

System administrators are often challenged to consolidate systems, classify application services, and track resource usage in an effort to better utilize resources. To optimize workload response, system administrators must be able to classify the workloads running on the consolidated system, information which historically has been difficult to obtain. *Workload classification* enables administrators to identify application services based on business rules—and the Solaris OS incorporates two facilities to help separate and identify workloads and track resource usage: *projects* and *tasks*.

Projects are used to label *workloads*, or *services*, and distinguish them from one another. A *project* is a workload tag that is used to classify a service, such as a database instance, and indicate which users or groups are allowed to join it. A *task* may be a specific job within a project, or a collection of processes performing a single job. A task enables users to identify a specific job within a project, such as a query to a specific database instance. Administrators can place related processes in a task, enabling the generation of detailed accounting records for the resources consumed to perform the task. Each process is a member of a task, and each task is associated with one project. A project may consist of one or more tasks.

Figure 2-1 illustrates the relationship between processes, tasks and projects. The Database resource pool contains two projects, named Oracle and Engineering. The Oracle project is an instance of an Oracle database. Two tasks are running in the project for the database: Task 1 is a database query, while Task 2 is writing new information into the database. The Engineering project is a CAD program, and Task 3 is rendering a new image of a circuit design.

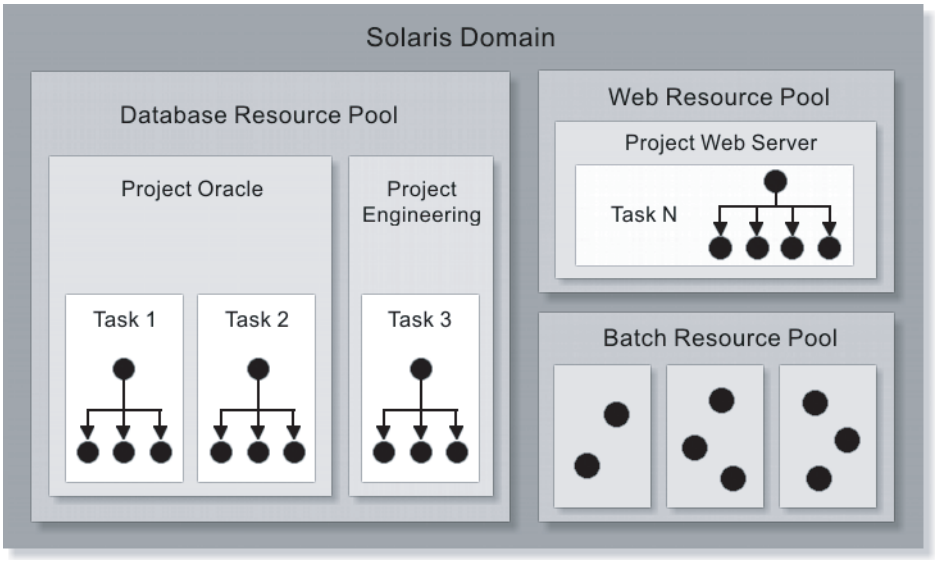

FIGURE 2- 1 Each project contains one or more tasks, each of which contains one or more processes

A user, or a group of users, can be a member of multiple projects. While a user must have a default project assigned, the processes launched by the user can be associated with any of the projects in which that user is a member.

Resource allocation policies are stored in a project database in the form of a local file, or in a Network Information Server (NIS) or Lightweight Directory Access Protocol (LDAP) database on a central server. As a result, the resource consumption of related workloads tagged under the same project ID that run on multiple machines can ultimately be analyzed across all machines. This ability to centrally manage distributed systems results in reduced administrative costs.

Resource Controls

System administrators need to be able to allocate and control resource usage for applications and users. The Solaris OS has long had the ability to establish limits on a per-process basis (rlimits), including the CPU time used, per-process core file size, and per-process maximum heap size. This concept has been extended to support tasks and projects. Now, the Solaris OS provides a comprehensive *resource control* facility that gives administrators the ability to place bounds on resource usage and prevent workloads from over-consuming resources.

A basic building block, resource controls enable administrators to establish resource limits on a per-process, per-task, and per-project basis, and change resource limits on a running process. In response, the system can perform the following actions on behalf of administrators:

- Send a signal when a threshold is reached
- Deny a resource request when a threshold is exceeded

Resource controls are configured through the project database. Process controls affect each process in a project, while task controls affect each task within a given project. Project controls affect all processes associated with a given project.

Differentiated Services

While resource pools provide the ability to partition a system, they do not specify how resources are to be shared by applications and services. Indeed, without an allocation scheme, resource contention may result. System administrators need a fine-grained method for controlling resource usage between different services within a resource pool. The Solaris OS incorporates an enhanced *Fair Share Scheduler (FSS)* that gives administrators the ability to specify that certain processes be given more resources than others. Now integrated into the Solaris kernel, the FSS software can be used on individual resources, or those within a resource pool, enabling administrators to modify the allocation policy for CPU resources to ensure important applications obtain more resources and affect better utilization.

With the FSS software, CPU resources are allocated on a per-project basis using project resource controls. Administrators control the allocation of available CPU resources among projects based on their importance. The relative importance of applications is expressed by allocating CPU resources based on *shares*—a portion of the system's CPU resources assigned to a project. The larger the number of shares assigned to a project, the more CPU resources it receives from the FSS software relative to other projects. The number of shares a project receives is not absolute—what is important is how many shares it has relative to other projects, and whether those projects will compete with it for CPU resources.

The FSS software guarantees the fair dispersion of CPU resources among projects based on allocated shares, independent of the number of processes attached to a project. Fairness is achieved by reducing a project's entitlement for heavy CPU usage and increasing its entitlement for light usage with respect to other projects. As a result, administrators ensure resources are not allocated to users and applications that are not entitled to them. As users and applications log in or out, the FSS software automatically recalculates the proportion of resources allocated to each active user. Resources can be varied based on previous usage.

With the FSS software, administrators can keep rogue processes from consuming all available processing power. Because CPU time is allocated based on an assigned number of shares rather than on a flat percentage basis, users can take advantage of additional processing power when applications are idle or are consuming less than their designated CPU allotment. As a result, administrators can protect key applications and ensure resources are fully utilized.

Virtualization and Containment

Part of an emerging family of containment technologies, *server virtualization* is designed to help reduce *server sprawl*—the proliferation of individual hardware servers and accompanying management and resource allocation problems. Server virtualization allows large servers to be flexibly partitioned into independent execution environments that provide total isolation within the same server. As a result, data centers can be visualized and managed as a fabric of interconnected computing resources rather than as a room filled with individual systems. To effectively implement server virtualization, organizations must be able to manage applications independently, control resource utilization, isolate faults, and ensure security between multiple applications.

Sun's approach to server virtualization centers on a concept called Solaris Containers technology. Conceptually, a Solaris Container is like a box. Each wall of the box provides an aspect of resource management, including:

- *Resource containment and control*, enabling fine-grained control of system resources through multiple, virtual environments that all share a common operating system kernel. For example, since virtual environments do not require assignment to physical processors, a minimum percentage of CPU resources can be assigned.
- *Virtualization*, providing a virtualized environment that hides hardware details from applications, such as physical device names, the primary IP address of the system, and the host name. In addition to providing security and application isolation, virtualization can be used to accelerate application provisioning.
- *Application isolation*, ensuring applications in one area cannot interact with applications in other areas. Application interaction is permitted for network Internet protocol (IP) communication, or when granted specifically by the administrator.

- *Security isolation*, ensuring that if intruders break into one area they do not have access to other areas on the system.

Because Solaris Containers are independent from the underlying hardware environment, application services can be re-created on other systems as needed. Each application runs in its own private environment—without dedicating new systems—and many application resources can be tested and deployed on a single server without fear that they will impact one another. System and network resources can be allocated and controlled on a fine-grained basis, helping simplify computing infrastructures and improving resource utilization. As a result, companies can better consolidate applications onto fewer servers without concern for resource constraints, fault propagation, or security breaches, simplifying service provisioning.

Solaris Containers are designed to be transparent—they do not present a new application programming interface (API) or application binary interface (ABI) to which applications must be ported. Standard Solaris OS interfaces and application environments are provided. Some restrictions are imposed, and primarily affect applications attempting to perform privileged operations. These restrictions are part of the security model implementation.

Application Isolation

A key inhibitor to consolidating applications is the lack of logical isolation between applications. Solaris Containers technology allows many private execution environments to be created within a single instance of the Solaris OS. Each virtualized environment, called a *Solaris Zone*, has its own identity that is separate from the underlying hardware. These virtual environments appear to be a separate system to applications and users. However, every zone has its own namespace, and therefore has its own users, root user, files, IP addresses, IP ports, hostname, and much more. It has everything it needs to act like an independent system from the application perspective (Figure 2-2). As a result, each zone behaves as if it is running on its own system, making consolidation simple, safe, and secure.

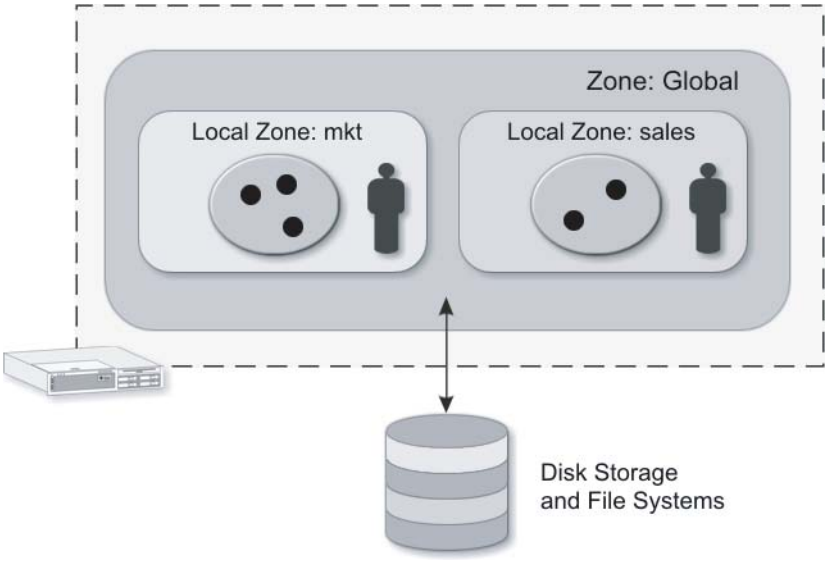

FIGURE 2-2 Solaris Zones effectively create separate environments on one physical system

The underlying original operating system, called the *global zone*, remains, and has its own namespace. The global zone is the place where the kernel runs, and from where the system is controlled and configured, and where the other *non-global zones* are created. Non-global zones (sometimes referred to as *local zones*) are isolated from each other. Not only do they have a separate namespace, non-global zones cannot *see* one another, their processes, or their attributes, such as IP addresses. Non-global zones also cannot share memory through mechanisms like IPC, and even have their own user level operating system services, such as `inetd`, `telnetd`, `sshd`, and so on. Because every zone is isolated in this way, zones can be independently booted and rebooted at will in order to start and stop a set of constrained processes without disturbing the other environments on the system. Figure 2-3 illustrates the relationships between the global zone and non-global zones, as well as the underlying resource management facilities used to control resource allocation and utilization.

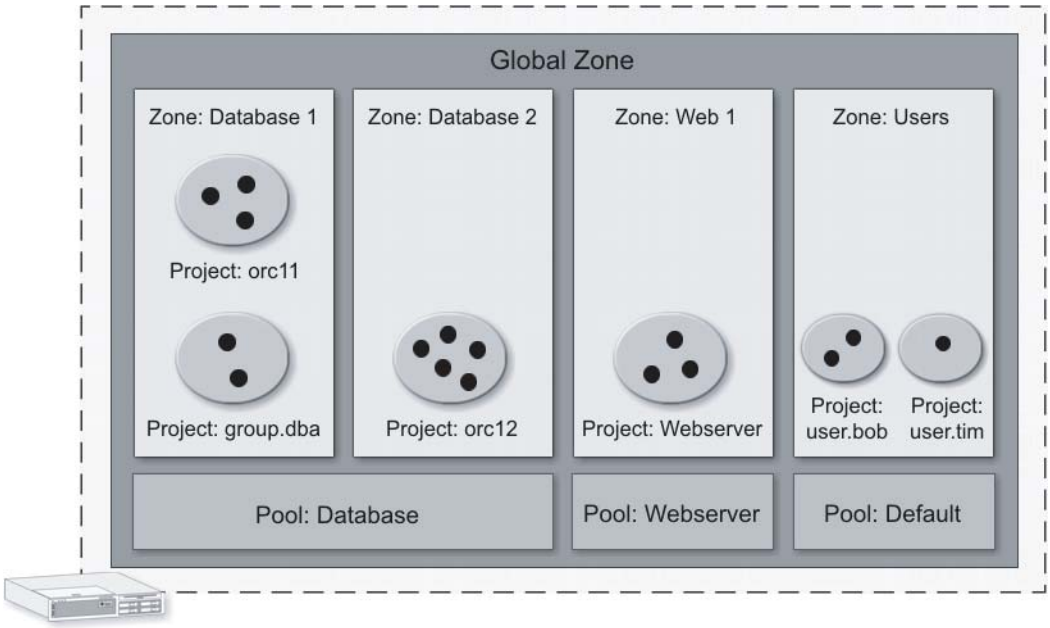

FIGURE 2-3 A system with several projects running in Solaris Zones that are assigned to resource pools.

3

An In-Depth Look at Containment and Virtualization

Deploying applications, selecting appropriate server resources to support them, and managing the resulting environment is a complex problem. Many IT managers take a simple approach—assign each application its own server. Why? They do not want applications to interfere with each other in any way, and perceive this can only be accomplished through dedicated, application-specific hardware servers. This belief may be motivated by mistrust of the application, mistrust of other users or applications that could potentially share the same server, not wanting to put too many eggs in one basket, or other technical and organizational reasons.

Figure 3-1 depicts a typical non-virtualized server. A single operating system per server directly initializes and controls all hardware. This simplistic approach often results in a large number of servers that are typically under-utilized, are difficult to manage effectively, and increase requirements for data center floor space, cooling, and power. Server utilization problems are common when expected application demand and peak loads placed on the server are variable and uncertain. As a result, organizations tend to over compensate. Most servers are considerably oversized—and are therefore significantly under-utilized, with utilization rates as low as five to 15 percent—for much of their deployment life. In addition, if a server is overcommitted, spare resources from other systems cannot be easily transferred to ease the problem. Complicating this situation are additional requirements for data backup, high availability, and keeping both applications and operating systems up to date.

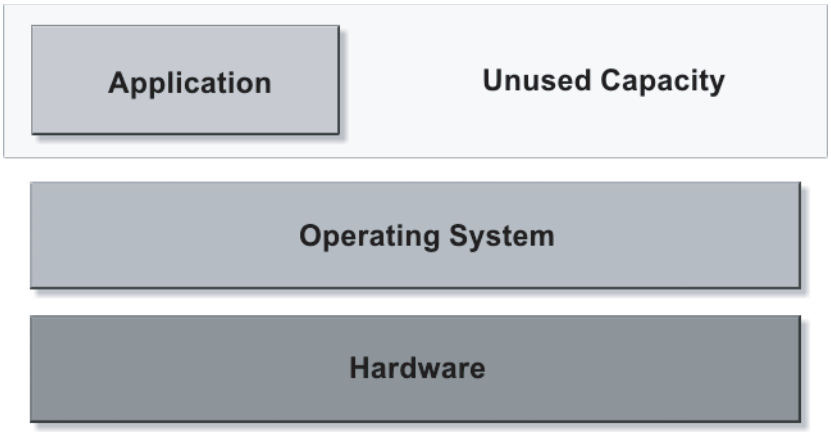

FIGURE 3-1 A non-virtualized environment

A variety of hardware and software technologies has evolved to help address these problems. Nearly all solutions involve some form of *containment*, techniques that help prevent the spreading of material or effects beyond a barrier or boundary. A *container* is a specific implementation of the containment concept. For example, a bottle is a container for liquid; it prevents liquid from spreading into places in which it is not wanted, and at the same time protects the liquid from contaminants. When applied to computing environments, a server can be thought of as one type of container for an application environment. The boundaries of the server prevent the application from affecting other systems, and protect the applications running on it from most external effects. These boundaries may be physical (hardware-based) or implemented in software. Examples of containment technologies include physical server partitioning, such as Sun's Dynamic System Domains, and software-based solutions such as Solaris Containers technology.

Containing Services

In computing environments, it may be important to contain applications, processes, groups of users, and possibly complete operating systems. Each of these categories can be thought of as a *service*, a long-lived set of software objects with well-defined states, error boundaries, start and stop mechanisms, and dependency relationships to other services. A service must be viewed and managed— that is, *contained*—as a single entity. A container is therefore a *bounded environment* for a service; such environments can be implemented and managed using a wide variety of hardware and software technologies.

Ideally, container solutions should provide:

- *Resource containment*, the ability to allocate and manage resources assigned to the container, such as CPUs, memory, network and I/O bandwidth
- *Security containment*, the bounding of user, namespace, and process visibility, hiding activity in each container from other containers, limiting unwanted process interaction, and limiting access to other containers
- *Fault containment*, hardware errors (failed components) and software errors (such as memory leaks) in one container should not affect other containers
- *Scalability*, the ability to exploit enterprise class systems by creating and managing a potentially large number of containers without significant performance overhead
- *Flexible resource allocation*, the ability to share resources from a common pool or to dedicate resources to specific containers
- *Workload visibility*, the ability to view system activity both from within the container and from a global system perspective
- *Management framework*, tools and procedures to create, start, stop, re-create, restart, reboot, move, and monitor containers, as well as provisioning and versioning
- *Hardware independence*, where possible, containment technologies should not require special hardware
- *Native operating system support*, solutions should not require a custom or ported kernel, as this has an impact on ISV supportability

Approaches to Containment

The first general purpose mainframe computers were very large, extremely expensive, and relatively few in number. As a result, utilization and efficiency were critical. These systems were designed to run many tasks simultaneously using more than one operating system at very high (over 90 percent) utilization rates. To provide multiple independent and contained execution environments, mainframes used hardware configuration managers (HCM) or virtual machine monitors (VMM), also called hypervisors. These programs interact directly with the system hardware, and sometimes run on a dedicated component known as a service processor (SP).

The HCM and VMM software enable system hardware to be partitioned into multiple containers. The term *virtual* is used to indicate that access to the physical hardware is abstracted to hide implementation details. For example, a VMM-controlled container accesses system resources, such as disks and network cards, through interfaces presented by the VMM rather than direct communication with the device. When running on specialized hardware, these containers can be fully isolated and independent environments capable of being separately powered, configured, booted, and administered. Figure 3-2 illustrates the different approaches taken by HCM- and VMM-based techniques.

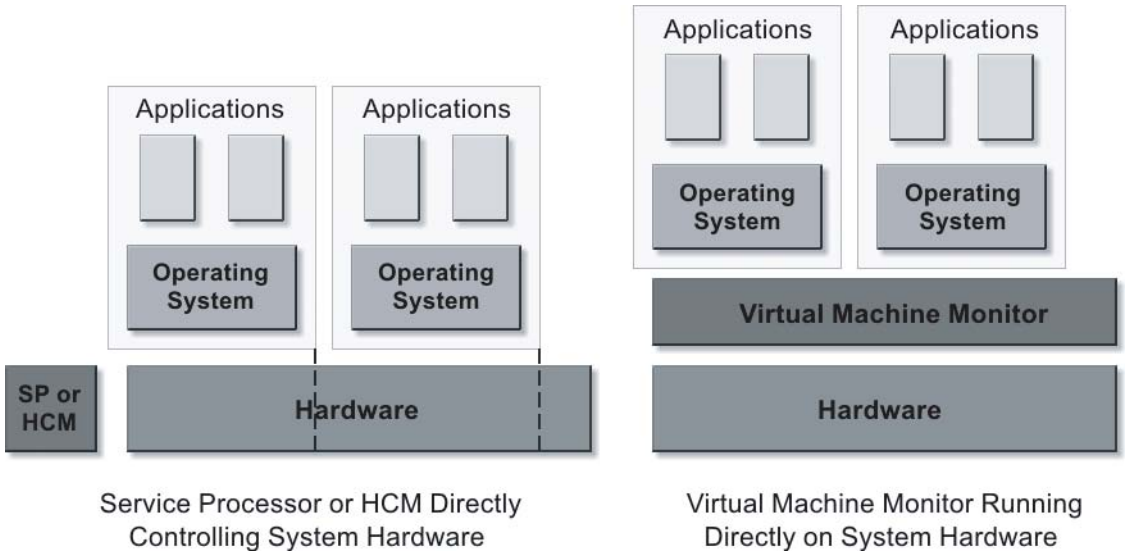

FIGURE 3-2 Hardware configuration managers and virtual machine monitors take different approaches to controlling hardware resources

Many vendors utilize a similar approach, including Sun's Dynamic System Domains. Containers constructed in this manner are called *hardware domains* or *partitions*, and may support different operating systems, or different releases of the same operating system, in each partition. Creating these partitions typically requires specialized hardware, and the number and size of partitions supported on a given server may be limited. For example, the Sun Fire™ E25K Server supports a maximum of 18 domains; the smallest domain must use at least a system board with two or four processors, and each domain requires its own boot device and network connection.

The purpose of this and other containment techniques is to enable safe and efficient workload management on a single, physically shared resource. Safe—because any solution should, by definition, have the means of being secure. Efficient —because sufficient tools and technologies that aid in the effort already exist. Technologies like Sun's Dynamic System Domains are designed to work in conjunction with resource management and partition reconfiguration services. These services permit the reallocation of CPU and other resources from one partition to another in order to balance loads and improve utilization. Reallocation is dynamic, meaning the reassignment of a resource from one partition to another partition typically does not require shutting down the operating system or applications running in the partitions. Operating systems and applications are fully contained within their respective partitions, and effectively run on separate hardware servers, albeit within the same physical system enclosure. What happens within one partition—resource consumption, application misbehavior, security issues, and hardware faults—generally has no effect on other partitions. When deploying multiple containers on a shared resource, any single point of failure must be recognized and addressed with redundant hardware and software components, such as clustering or other high availability solutions.

The popular blade server architecture can be viewed as a kind of hardware partitioning for large numbers of similar or identical applications, such as Web servers. Each blade is an individual hardware server running a complete operating system and application set. While blade servers can address a variety of scalability and redundancy issues, individual blades are constrained by their size, typically one or two processors with limited memory. These constraints can reduce the flexibility needed to allocate sufficient CPU and memory resources for demanding applications.

Software-Based Containment

Hardware containment methods originated the 1960s and 1970s on early mainframe systems and continue today on modern, enterprise-class servers. However, they almost always require specialized systems capable of hardware partitioning, like the Sun Fire E25K server. In recent years, however, several commercial and open source software-based containment solutions have emerged. These solutions generally do not require specialized hardware, and can run on a wide range of systems, from laptops and desktop workstations, to mid-range and enterprise-class servers. Some of these software-based containment solutions operate as shown in Figure 3-2 but do not require special hardware to run the VMM.

Figure 3-3 describes the general architecture of software-based containment solutions that use a *hosted* VMM. In these solutions, a primary operating system runs directly on the system hardware, and a VMM runs as an application under the host operating system. The hosted VMM permits multiple guest operating systems, such as Linux or the Solaris OS, along with their applications, to run simultaneously in a contained manner on the host system. Administrative tools are provided to allocate and change resources among the guest operating systems. Additionally, applications can be run directly on the primary operating system, ignoring the VMM entirely.

Not all server containment technologies require a VMM. In fact, VMMs can consume significant CPU resources as they rewrite or redirect guest operating system code, especially when they need to intercept and redirect privileged guest operating system instructions.

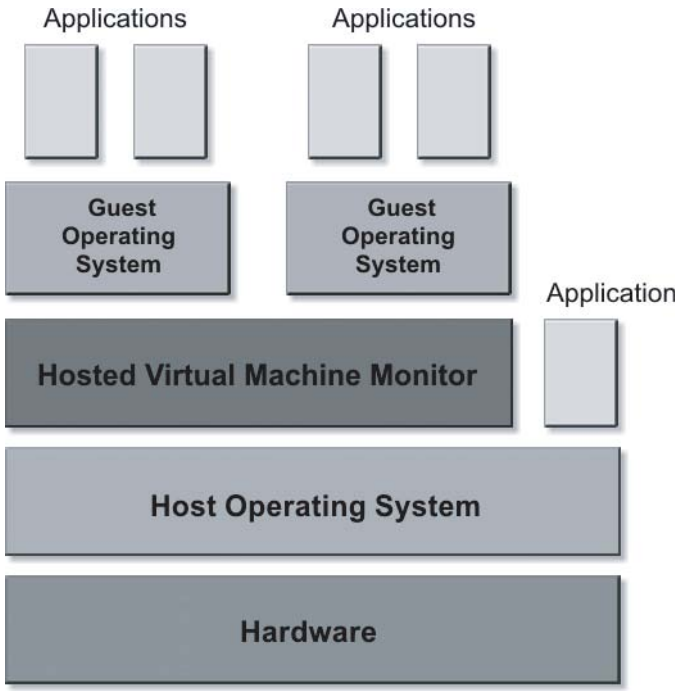

FIGURE 3-3 Software partitioning using a hosted VMM

Solaris Containers

An operating system's primary task is the efficient management of processes. The operating system allocates shares of system resources, such as CPUs, memory, and I/O, and sets minimum guaranteed boundaries for the execution of the processes that use them. If a collection of processes and resources can be defined and bounded to match the requirements of a contained server environment, server virtualization can be accomplished efficiently without the use of a separate VMM. This approach to containment, often described as operating system virtualization, is the approach taken with Solaris Containers technology. In the Solaris OS, virtual server environments are implemented using a type of container called a Solaris Zone. Other types of containers exist in the Solaris OS, such as projects and limit nodes. Much of the current discussion and literature about Solaris Zones treats a Zone and Container as if they were equivalent. To be clear, a Zone is one type of container, one that encapsulates a server environment, limits the effects of that environment on other system activities (including other active zones), and protects the environment from outside influence.

Since a container is defined as a bounded environment for a service, and a service is a group of processes managed as a whole, then a zone is a container for the service or group of processes that implements a virtual server.

Figure 3-4 illustrates the concept of Solaris Containers technology. The Solaris OS runs directly on the hardware, manages the boot process, and initializes interfaces to the CPUs, memory, host bus adapters, network interface cards (NICs), storage, and device drivers in the system. Only *one* instance of the Solaris OS runs on the hardware, and it is referred to as the *global zone*. The administrator defines one or more non-global zones that contain virtual server environments. A non-global zone appears to all users—end users, applications, developers, and the zone administrator—as a fully realized server with its own host name, IP address, process and name space, root and user names and passwords, and network devices and file systems.

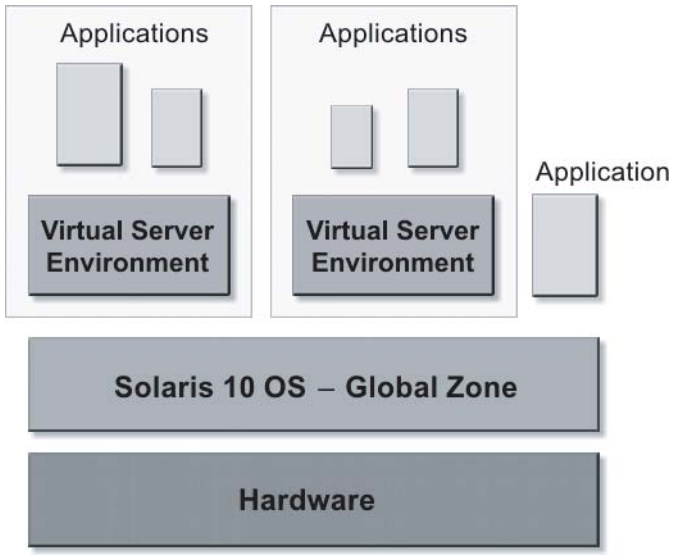

FIGURE 3-4 Solaris Containers and the Global Zone

Note that Solaris applications are not required to be contained in zones. Similar to a traditional approach, applications can run directly in the global process and name space of a single operating system instance. In such a scheme, however, the benefits of being able to contain the application in a non-global zone cannot be realized.

As multiple non-global zones are defined and implemented on a system running the Solaris OS, workload management of the non-global zones must also be considered in order to achieve the goal of secure and efficient resource allocation among multiple active zones. When defining a non-global zone, it is important to determine how resources should be allocated. Resources may be simply shared with the global zone, or dedicated to a specific zone. For example, a server with multiple network interfaces can be configured so that one non-global zone is assigned exclusive access to one interface, while all other zones share

access to the remaining interfaces. The resource management facilities incorporated into the Solaris OS permit these resource assignments and Quality of Service (QoS) parameters to be defined for CPU, memory, and network usage so that application demands in one zone do not affect the performance of other zones. For example, resource management tools may be used to guarantee that database queries are assigned at least 50 percent of available system resources, or to limit compiler tasks to no more than 10 percent of available resources.

The Solaris 10 OS contributes to secure architectures by providing secure execution containers for applications. Solaris Containers run with reduced privileges within the global zone. Non-global zone processes cannot modify these privileges, load kernel modules, or alter shared read-only file systems provided by the global zone. Processes in non-global zones are fully observable and can be audited from the global zone. As a result, Solaris Containers technology provides an important building block for creating a secure IT infrastructure. This can be accomplished through the use of standardized operating environment configurations, which can help promote more consistent security and predictability while aiding organization compliance efforts.

Containment and Virtualization Trade-Offs

It is important to contain application environments in some way to ensure security and stability. Virtualization technologies can help:

- Manage the task of resource allocation among these environments
- Enable consolidation of server workloads
- Allow for hosting of untrusted or hostile applications,
- Provide flexible development, testing, and production environments.

Additionally, some virtualization technologies provide a repeatable way to troubleshoot a contained application service from a global view of the operating system, without affecting other application services. For example, the Solaris Dynamic Tracing (DTrace) facility available in the Solaris 10 OS provides built-in instrumentation and observability of the kernel and applications, reducing the time it takes to identify and correct performance and stability problems.

The following questions may prove helpful when considering virtualization solutions:

- Are existing servers over-provisioned and under-utilized?
- How long does it take the organization to install, configure, and deploy a new server or application environment?
- Does the number and variety of hardware servers in use make it difficult to manage them all effectively?
- Is there a need to run a variety of operating systems?
- Is there a need to run different versions or patch levels of one operating system family?

- Is there a need run multiple instances of the same operating system?
- Is there a need to quickly create and reconfigure server environments for testing, development, and production?
- Is the expertise and staff available to support multiple, virtualized instances of several operating systems?
- Are in-house and commercial applications supported in a virtualized environment? What are the licensing costs and pitfalls?
- Do the virtualization solutions include tools for monitoring and managing contained applications and operating systems?
- What is the cost of the virtualization solution?
- What is the performance impact or overhead of the solution?
- How can virtualized systems be backed up?
- How are redundant or highly available configurations using virtualized systems created?
- How can multiple virtual operating systems and applications be updated and maintained?

Virtualization technologies are maturing, enabling them to provide contained environments for quickly creating and testing applications and operating systems, guarantee application quality of service (QoS), and increase overall system utilization and return on investment (ROI). At the same time, some virtualization technologies can add complexity to the overall IT infrastructure, increase licensing and administration costs, potentially add system overhead, and make diagnosis of system problems more difficult.

Solaris Containers technology addresses many, but not all, of these questions and issues. The Solaris OS includes Containers at no additional cost, as well as resource allocation and management tools.With the Solaris OS, there is only one operating system image to update and maintain. All application processes in non-global zones can be observed, controlled, and audited, and secure workload management capabilities are built in to the operating system. Moreover, exploiting Solaris Containers on Sun hardware that supports Dynamic System Domains can provide even greater flexibility in deploying fully contained and manageable application environments while getting maximum utilization from hardware investments. Additional tools, such as the Solaris Container Manager 1.1 software, can help system managers deploy Solaris Containers technology with greater effectiveness.

Guidelines for Deploying Solaris Containers

Solaris Containers technology provides the foundation for the consolidation recommendations presented. Several rules should be considered, including those common to any consolidation plan utilizing Solaris Container technology.

- Start with simple, low risk consolidations. Doing so decreases the impact of unplanned downtime, and enables staff to get through the rather shallow learning curve of this new technology.
- Start with systems that are early in the software life cycle, as such systems tend to be simpler and the user base is usually more tolerant of change. In addition, this results in a simple mapping of software revision and operating system version as changes move downstream. These first two rules justify beginning consolidation efforts with sandbox and development environments.
- Many instances of a software package are easier to consolidate than different applications. Different applications can potentially require multiple operating system patch levels. While this can be accommodated, it decreases the benefits of reduced administration typically achieved by consolidation.
- Develop naming conventions and standards for Solaris Containers, network addresses, and file systems. Solaris Containers permit flexibility in each of these areas. Local conventions make it easier to deploy and administer Solaris Containers and the applications they contain, and may help to reduce human error.
- Consolidating multiple tiers of an application benefits from better network latency and bandwidth. Network traffic between Solaris Containers does not leave the system—only memory transfers are involved. This can also be used to provide secure network communication without encrypted network connections, providing security without the cost of CPU cycles or special network offload devices.
- Consolidating backup servers should involve careful analysis of the data traffic to avoid creating an unacceptable bottleneck, such as PCI channel bandwidth congestion.
- Look for opportunities to combine lightly loaded systems, applications with different load types (such as one using I/O but few CPU resources, and one using CPU and few I/O resources), or applications with peak loads that occur at different times. Note that consolidating systems increases CPU, memory, and I/O utilization. Each resource should be sized appropriately, and the side effects of a shortfall should be considered. For example, two identical systems which use 75 percent of CPU and memory capacity could be consolidated. However, the performance impact of insufficient memory will dwarf the impact of insufficient CPU power.
- Look for opportunities to consolidate software licenses from lightly loaded systems. Software that is priced based on the number of CPUs and runs on many low utilization CPUs can represent a significant software expense that can be reduced by minimizing licensed CPU counts.

- Maintenance windows should be taken into account when consolidating. Consolidating multiple systems with non-overlapping maintenance windows may lead to the inability to perform system maintenance.
- Take special care with security boundaries. Although the security isolation of Containers enables a system to include both external and internal zones, the increased risk (probability of downtime) of an external zone may make consolidation with an internal mission-critical server inappropriate.
- Servers that provide static content, such as static Web servers, are excellent candidates for consolidation. Content can be contained in the global zone and mounted read-only by Web servers located in Containers. If a Web server is compromised, the Web content cannot be modified.
- Avoid consolidating systems with different availability requirements. For example, do not consolidate servers used to test software with 7x24 production servers. However, exceptions may exist that permit such a consolidation.
- Systems which occasionally require different versions of the operating system should not be consolidated.
- High availability solutions should be architected and implemented to eliminate single points of failure.

4

Managing Workloads

Running multiple applications on a single computer system without a means to control how applications use system resources can lead to unpredictable service levels. By default, the Solaris OS treats every resource request with equal priority. If there is enough of the resource available the request is granted. If the demand for the resource exceeds the total capacity available, the Solaris OS adapts by restricting access to the resource. The action taken to restrict access depends on the type of resource. For example, should demand for CPU time exceed the CPU time available, the scheduler reacts by adjusting the priorities of processes in order to change the distribution of the CPU time. The scheduler operates on threads and has no concept of applications, let alone their relative importance from a business perspective. An unimportant CPU-bound application can victimize other, more important applications by placing high demand for CPU resources on the system.

Other resources, such as the total number of processes on the system, have a fixed upper bound. Once the limit is reached, no more of this resource can be used. A runaway process that keeps creating new processes can prevent new useful work from being started. Other than specifying the system-wide upper limit, there is no way to limit the number of processes that may be created by an application or a set of applications.

What is needed is a way to control resource usage based on workloads. A *workload* is an aggregation of all processes of an application, or group of applications, that makes sense from a business perspective. Instead of managing resource usage at the process level, it should be possible to manage resource usage at the workload level. This allows the implementation of policies such as “the Sales application shall be granted at least 30% of CPU resources” as part of a service level agreement. The Solaris OS resource management features make it possible to treat workloads in this way by:

- Restricting access to specific resources
- Offering resources to workloads on a preferential basis
- Isolating workloads from each other

The first step in managing resource usage by workloads is identifying or classifying the components, such as processes, that make up the workload. The next step is measuring the resource consumption of these workloads. Finally, by applying constraints on the use of resources the workloads can be controlled. The constraints applied follow from the policies defined for the workloads based on business requirements.

A possible policy could be that an important workload should always be granted a minimum amount of CPU time even on an overloaded system. Another policy could be that a workload is only granted access to the CPU if there are no other workloads requiring CPU resources.

Projects

The first step in managing resource usage involves identifying the workloads running on the system. Possible approaches include identifying workloads by user name or process name. While simple, this poses a challenge when multiple instances of the same application are running on the system for different workloads, such as a sales application database and a marketing application database. Unless the database application provides a way to run the instances as different users, it is imposible to attribute resource usage to a specific workload based solely on userid. In addition, aggregation of multiple related applications, such as database servers, application servers and Web servers for a business application on one system is not possible.

The Solaris OS provides a facility called *projects* to identify workloads. The project serves as an administrative tag used to group related work in a manner deemed useful by the system administrator. System administrators can, for example, create one project for the sales application and another project for the marketing application. By placing all processes related to the sales application in the sales project and the processes for the marketing application in the marketing project, the administrator can separate, and ultimately control, the workloads in a way that makes sense to the business.

A user that is a member of more than one project can run processes in multiple projects at the same time, making it possible for users to participate in several workloads simultaneously. All processes started by a process inherit the project of the parent process. As a result, switching to a new project in a startup script runs all child processes in the new project.

Using Projects to Define Workloads

In the example of the sales and marketing applications, the system administrator can create two new projects, one for the sales application and one for the marketing application. The application startup scripts must be modified to switch to the desired project as part of the application startup. The sales application startup script switches to the sales project, and the marketing application switches to the marketing project. This results in both applications

running in different projects while still using the same userid. Adding another application, such as a Web server, to the sales application workload requires adding the Web server user to the sales project and modifying the Web server startup script to switch to the sales project. With the introduction of the Service Management Facility (SMF) in the Solaris 10 OS, administrators can assign the project in which to run the application or service through service properties in the SMF repository.

The Project Database

Projects are defined in the project database. The project database can be a local file or in a name service such as NIS or LDAP. By putting the project database in NIS or LDAP, the project definition can be shared across multiple systems. Each entry in the project database consists of the following fields:

- **name**, the name of the project
- **id**, the project's unique numerical ID
- **comment**, the description of the project
- **user list**, a list of users allowed in the project
- **group list**, a list of groups allowed in the project
- **attributes**, a list of project attributes, such as resource controls

A freshly installed system always contains a local project database */etc/project* containing five standard projects:

- `system`, used for all system processes and daemons
- `user.root`, used for all processes run by root
- `noproject`, for processes specific to IP quality of service (IPQoS)
- `default`, for users not matching any other project (a catch-all project)
- `group.staff`, for all users in the group `staff`

A user or group can be a member of one or more projects. The user and group lists in the project database determine in what projects a user or group of users can execute processes. These lists can contain wildcards to allow for flexible definitions, such as 'all members of group staff excluding user bob'. Users can switch to any project of which they are a member. Until the user changes the project in which to execute a process, all processes run in the user's default project. The user and group lists only define the project(s) in which a user or group is allowed to execute processes. It does not define a default project for the user or group. The default project for a user is determined by the system at login time. See the `man` page for `getprojent(3C)` for the exact algorithm used.

Commands

The following commands are available to administer projects:

Command	Description
`projadd(1M)`	Adds a new project to the local project database
`projmod(1M)`	Modifies a project entry in the local project database
`projdel(1M)`	Deletes a project entry from the local project database
`projects(1)`	Displays project membership for a user
`newtask(1)`	Switches to a project

Several standard Solaris OS commands include project related options, and can be used to view or manipulate processes based on their project membership:

Command	Option
`id(1M)`	`-p`
`ipcs(1)`	`-J`
`pgrep(1)`	`-J -T`
`pkill(1)`	`-J -T`
`poolbind(1M)`	`-i` *project*
`prctl(1)`	`-i` *project*
`priocntl(1M)`	`-i` *project*
`prstat(1M)`	`-j -J -k -T`
`ps(1)`	`-o` *projid project taskid*
`useradd(1M)`	`-p`

For example, the `prstat -J` command lists all processes and projects on the system and displays a per project total. See the `man` pages for more information on these commands and the options related to projects.

Extended Accounting

Once workloads are identified and labeled using projects, the next step in managing resource usage involves measuring workload resource consumption. While current consumption can be measured using the `prstat(1M)` command to obtain real-time snapshot of resource usage, it does not provide the capability to look at historical data.

The traditional accounting mechanism is process based and predates the introduction of projects. It is therefore unable to provide resource usage statistics based on workloads. The extended accounting facility allows collection of statistics at the process level, the task level or both. Accounting at the task level aggregates the resource usage of its member processes, thereby reducing the required disk space for accounting data. A *task* is a group of related processes executing in the same project as a result of a `newtask(1)` command. An accounting record is written at the completion of a process or task. Interim accounting records can be written for tasks, and can be used to provide accurate daily accounting for long running jobs that span multiple days.

Every process that runs in the system is associated with a project and a task. By labeling all resource usage records with the project for which the work was done, the extended accounting facility can provide data on the resource consumption of workloads. This data can be used for reporting, capacity planning or charge back schemes.

Unlike the traditional System V accounting mechanism that is based on fixed size, fixed semantic records, the extended accounting facility uses a flexible and extensible file format for accounting data. Files in this format can be read or written using the C language API provided by `libexacct(3LIB)`. This API abstracts the accounting file and offers functions to read and write records and fields in the file without the need for knowledge of the physical layout. This makes it possible to add new record or field types to the file between releases, even during system operation, without impacting existing applications that use extended accounting files. A Perl interface for `libexacct` is available to ease the creation of custom reporting tools.

Commands

The following commands are available to administer the extended accounting facility.

Command	Description
`acctadm(1M)`	Configure extended accounting
`wracct(1M)`	Write extended accounting records for active processes and tasks

The Fair Share Scheduler

Running multiple workloads on the same system can lead to a situation where one workload monopolizes CPU resources and impacts other workloads. This may result in important workloads not receiving sufficient CPU resources to complete their work. It is desirable to have a mechanism by which system administrators can prioritize access to CPU resources based on the importance of the workload.

The policy of the default scheduler in the Solaris OS is to give every process relatively equal access to CPU resources. Since it has no knowledge of workloads, the default scheduler cannot prioritize CPU allocation based on workload importance. The Solaris OS offers an

alternative scheduler that is aware of workloads and can prioritize CPU allocation with respect to workload importance.

CPU Shares

The *Fair Share Scheduler* (FSS) controls allocation of CPU resources using *CPU shares*. The importance of a workload is expressed by the number of shares the system administrator allocates to the project representing the workload. The Fair Share Scheduler ensures that CPU resources are distributed among active projects based on the number of shares assigned to each project (Figure 3-1).

A CPU share defines a relative entitlement of the CPU resources available to a project on the system. It is important to note that CPU shares are *not* the same as CPU percentages. Shares define the *relative importance* of projects with respect to other projects. If project A is deemed twice as important as project B, project A should be assigned twice as many shares as project B. The actual number of shares assigned is largely irrelevant — two shares for project A versus one share for project B yields the same results as 18 shares for project A versus nine shares for project B. In both cases, Project A is entitled to twice the amount of CPU resources as project B. The importance of project A relative to project B can be increased by assigning more shares to project A while retaining the same number of shares for project B.

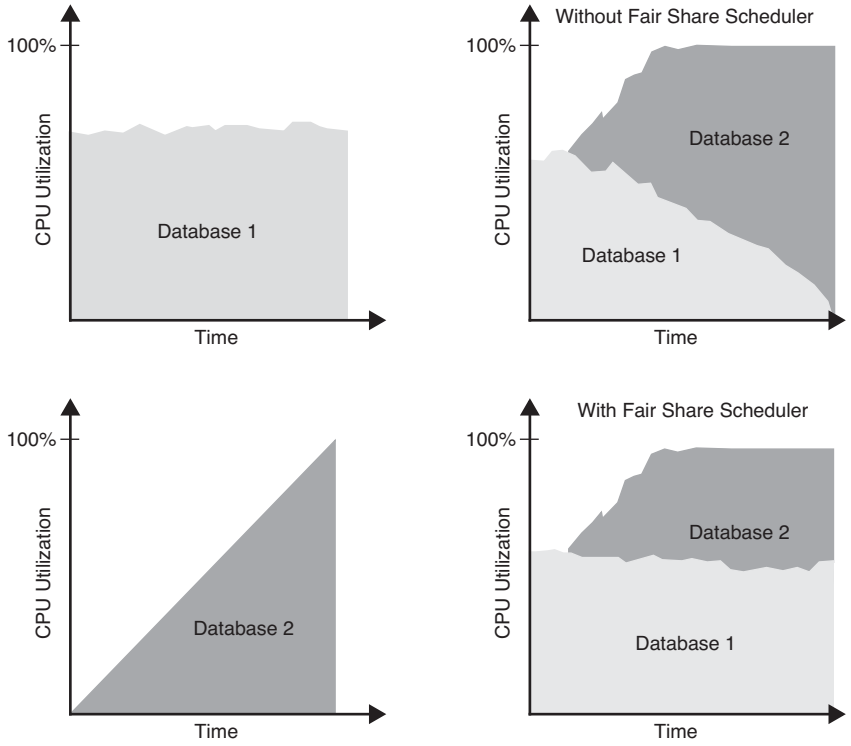

FIGURE 4-1 The Fair Share Scheduler ensures applications get the CPU resources to which they are entitled.

The Fair Share Scheduler calculates the proportion of CPU resources allocated to a project by dividing the shares for the project by the total number of shares of active projects. An *active project* is a project with at least one process using CPU resources. Shares for idle projects, such as those without active processes, are not used in the calculations. For example, consider projects A, B and C with two, one and four shares respectively. If projects A, B and C are active, then project A is entitled to $\frac{2}{7}$, project B is entitled to $\frac{1}{7}$, and project C is entitled to $\frac{4}{7}$ of CPU resources. If project A is idle, project B is entitled to $\frac{1}{5}$ of CPU resources, and project C is entitled to $\frac{4}{5}$ of CPU resources (Figure 3-2). Note that even though the actual CPU entitlement for project B and C increases, the proportion between project B and C stays the same (1:4).

It is important to note that the Fair Share Scheduler only limits CPU usage if there is competition for CPU resources. If there is only one active project on the system, it can use 100% of CPU resources, regardless of the number of shares it holds. CPU cycles are never wasted. If a project does not use all the CPU resources it is entitled to because it has no work to do, the remaining CPU resources are distributed between other active projects.

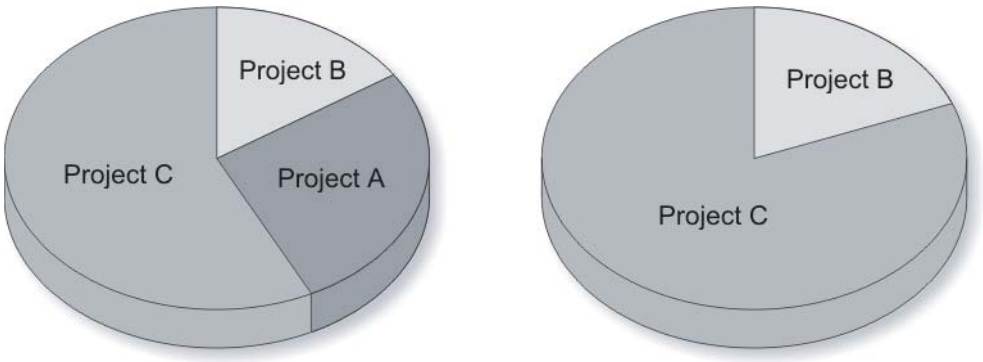

FIGURE 4-2 The Fair Share Scheduler distributes CPU resources among active projects based on the number of CPU shares

CPU Shares Configuration

CPU shares are configured through the `project.cpu-shares` resource control in the project database. Every project can be assigned a `project.cpu-shares` resource control. Projects without this resource control are assigned one share by the system. The `system` project is used for all system processes and daemons, and is special in that it has unlimited shares. Projects with zero shares assigned are only allowed to run when no other projects with non-zero shares are active.

Users can be a member of multiple projects and CPU usage is controlled by the number of shares of the project in which the user executes. As a result, a user can be entitled to different amounts of CPU resources at the same time. Note that a process can only be in one project at a time, so having different amounts of CPU resources at the same time means that processes owned by this user reside in different projects.

To place a CPU usage limit on a single user, create a project with the appropriate number of shares that contains only that user. This project should be the default project for this user, and the user should not be a member of any other projects to prevent the user from switching to another project.

The CPU shares can be adjusted dynamically using the `prctl(1M)` command. These changes are valid until the next system boot. To make the changes permanent, update the `project.cpu-shares` resource control in the project database.

Resource Controls

Resource usage of workloads can be controlled by placing bounds on resource usage. These bounds can be used to prevent a workload from over-consuming a particular resource and interfering with other workloads. The Solaris OS provides a resource controls facility to implement constraints on resource consumption. This facility is an extension of the traditional UNIX resource limit facility (*rlimit*). The rlimit facility can be used to set limits on the resource usage of processes, such as the maximum CPU time used, the maximum file size, the maximum core file size, and more. However, as the rlimit facility is process-based, its use for constraining workloads is rather limited. The resource controls facility in the Solaris OS extends process-based limits by adding resource limits at the task and project level. The number of resource limits that can be set is also expanded to give system administrators more control over resource consumption by processes, tasks and projects on the system.

Administering Resource Controls

Resource controls are configured through the project database. The last field of the project entry is used to set resource controls. A resource control in the project entry is a name-value pair. The name denotes the type of limit, while the value is a list of attributes for the control. Multiple resource controls can be added to a single project entry by separating the resource controls with a semicolon. The list of attributes for a resource control consists of a privilege level, a threshold, and an action.

The privilege level determines which users can modify the threshold value. Three privilege levels are provided:

- **basic**, the owner of the calling process can change the threshold
- **privileged**, only privileged (superuser) users can change the threshold
- **system**, the threshold is fixed for the lifetime of the operating system instance

Every resource control has at least a system value, which represents how much of the resource the current implementation of the operating system is able to provide. A resource control can have at most one basic value and any number of privileged values.

The action defines the steps to be taken when the threshold is exceeded. Three actions are possible:

- **deny**, deny resource requests for an amount that is greater than the threshold
- **signal**, send the specified signal to the process exceeding the threshold value
- **none**, perform no action when the threshold is exceeded

Note – Changes made in the project database are only applied when a new process, task or project starts. Existing processes, tasks and projects do not see these changes. The `prctl(1M)` and `rctladm(1M)`commands can be used to change resource controls on active entities.

Available Resource Controls

The following table identifies the resource controls available in the Solaris 10 OS.

Resource Control	Description
process.max-port-events	Maximum allowable number of events per event port
process.max-msg-messages	Maximum number of messages on a message queue
process.max-msg-qbytes	Maximum number of bytes of messages on a message queue
process.max-sem-ops	Maximum number of semaphore operations allowed per semop call
process.max-sem-nsems	Maximum number of semaphores allowed per semaphore set
process.max-address-space	Maximum amount of address space available to this process
process.max-file-descriptor	Maximum file descriptor index available to this process
process.max-core-size	Maximum size of a core file created by this process
process.max-stack-size	Maximum stack memory segment available to this process
process.max-data-size	Maximum heap memory available to this process
process.max-file-size	Maximum file offset available for writing by this process
process.max-cpu-time	Maximum CPU time available to this process
task.max-cpu-time	Maximum CPU time available to this task's processes
task.max-lwps	Maximum number of LWPs simultaneously available to tasks's processes
project.max-contracts	Maximum number of contracts allowed in a project
project.max-device-locked-memory	Total amount of locked memory allowed in a project
project.max-port-ids	Maximum allowable number of event ports
project.max-shm-memory	Total amount of shared memory allowed for a project
project.max-shm-ids	Maximum number of shared memory IDs allowed for a project
project.max-msg-ids	Maximum number of message queue IDs allowed for a project
project.max-sem-ids	Maximum number of semaphore IDs allowed for a project
project.max-crypto-memory	Total amount of kernel memory that can be used by libpkcs11 for hardware crypto acceleration
project.max-tasks	Maximum number of tasks allowable in a project
project.max-lwps	Maximum number of LWPs simultaneously available to a project
project.cpu-shares	Number of CPU shares granted to a project for use with the FSS
zone.max-lwps	Maximum number of LWPs simultaneously available to zone's processes
zone.cpu-shares	Number of CPU shares granted to a zone for use with the FSS

Determining Thresholds

The resource consumption of processes is often unknown, so choosing a useful and safe threshold for a resource control can be a difficult task. Selecting an arbitrary threshold can lead to unexpected application failure modes. While some required information could be extracted from extended accounting information, there is a simpler way. The resource controls facility provides a global log action that sends a message to syslog when a threshold is exceeded.

First, a resource control with the threshold value to be verified must be set. The action should be set to ‘none’ to ensure the resource is not denied if the threshold is exceeded. This allows the process to run unconstrained. Next, the global syslog action for the resource control must be enabled. When the application exceeds the threshold for that resource control, a message that the resource control threshold has been exceeded is logged to syslog. By changing the threshold until the warning no longer appears during *normal* use of the application, a reasonable setting for the resource control can be determined. After determining the value for the resource control, the action should be changed to ‘deny’, to ensure the threshold is enforced by the system.

Commands

The following commands are available for administering resource controls. More information can be found in the `man` pages for each command.

Command	Description
`prctl(1M)`	Get or set resource controls on a running process, task or project
`rctladm(1M)`	Display or modify global state of system resource controls

Managing Workloads — An Example

To demonstrate the concepts explained in this chapter, this example uses the Solaris OS resource management facilities to manage workloads on an example system. The system is shared by several business units and is running two workloads: two database instances, one for a marketing application and one for a sales application.

A project is defined for each workload, enabling the Fair Share Scheduler to be used to manage CPU allocation between the workloads. A resource control is added to limit the amount of shared memory for each workload. To account for all activity of the `oracle` user that is not related to either of these workloads, a third project is created. This project is the default project for the `oracle` user.

Requirements

The following minimum requirements are needed to run this example:

- Oracle 9i media (version 9.2.0.1.0)
- 6 GB disk space for the Oracle binaries and databases

Defining the Projects

To keep things simple, a local */etc/project* database is used. The project entry in the */etc/nsswitch.conf* file should be defined as follows:

```
# cat /etc/nsswitch
...
project:    files
...
```

By convention, Oracle instances are run as the user `oracle` in group `dba`. As a result, the group `dba` and user `oracle` are created:

```
# groupadd dba
# mkdir -p /export/home
# useradd -g dba -d /export/home/oracle -m -s /bin/bash oracle
```

A project named `group.dba` is created to serve as the default project for the user `oracle`. The system uses the rules described in the `getprojent(3C)` man page to determine the default project when a user logs in. Since the default group of user `oracle` is the `dba` group, the `group.<groupname>` rule matches and the `group.dba` project is set as the default project for user `oracle`. A comment describing the project is added using the `-c` option:

```
# projadd -c “Oracle default project” group.dba
```

The `id(1M)` command can be used to verify the default project for the `oracle` user:

```
# su - oracle
$ id -p
uid=100(oracle) gid=100(dba) projid=100(group.dba)
$ exit
```

To manage each Oracle instance as a separate workload, a project is created for each Oracle instance to run in: project `ora_mkt` for the marketing Oracle instance, and project `ora_sales` for the sales Oracle instance.

```
# projadd -c "Oracle Marketing" -U oracle ora_mkt
# projadd -c "Oracle Sales" -U oracle ora_sales
```

The `-U oracle` option specifies that the `oracle` user is allowed to run processes in these projects. Once these steps are complete, the */etc/project* file contains the following information:

```
# cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
group.dba:100:Oracle default project:::
ora_mkt:101:Oracle Marketing:oracle::
ora_sales:102:oracle Sales:oracle::
```

The first five projects are projects that are created during system installation. Note that the system assigned project IDs for the last three projects since they were not explicitly specified on the `projadd` command.

System V IPC Resource Controls

The System V IPC resource limits in the Solaris 10 OS, such as the maximum shared memory size, are no longer set in the */etc/system* file, but instead are project resource controls. As a result, a system reboot is not longer required to put changes to these parameters in effect. This also allows system administrators to set different values for different projects. A number of System V IPC parameters are obsolete with the Solaris 10 OS, simply because they are no longer necessary. The remaining parameters have more

reasonable defaults to enable more applications to work out-of-the-box, without requiring these parameters to be set. The following table identifies the values recommended by the Oracle Installation Guide and the corresponding Solaris OS resource controls.

Parameter	Oracle Recommendation	Resource Control	Default Value
SEMMNI (semsys:seminfo_semmni)	100	project.max-sem-ids	128
SEMMNS (semsys:seminfo_semmns)	1024	obsolete	
SEMMSL (semsys:seminfo_semmsl)	256	project.max-sem-nsems	512
SHMMAX (shmsys:shminfo_shmmax)		project.max-shm-memory	1/4 physical memory
SHMMIN (shmsys:shminfo_shmmin)	1	obsolete	
SHMMNI (shmsys:shminfo_shmmni)	100	project.max-shm-ids	128
SHMSEG (shmsys:shminfo_shmseg)	10	obsolete	

Since the default values are higher than Oracle recommended values, the only resource control that must be set is `project.max-shm-memory`. To set the maximum shared memory size to 2 GB, add the `project.max-shm-memory=(privileged,2147483648,deny)` resource control to the last field of the project entries for the three Oracle projects.

```
# projmod -sK "project.max-shm-memory=(privileged,2G,deny)" group.dba
# projmod -sK "project.max-shm-memory=(privileged,2G,deny)" ora_mkt
# projmod -sK "project.max-shm-memory=(privileged,2G,deny)" ora_sales
```

Once these steps are complete, the */etc/project* file should contain the following. Note that changes are shown in italics.

```
# cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
group.dba:100:Oracle default project:::project.max-shm-
memory=(privileged,2147483648,deny)
ora_mkt:101:Oracle Marketing:oracle::project.max-shm-memory=(privileged,2147483648,deny)
ora_sales:102:oracle Sales:oracle::project.max-shm-memory=(privileged,2147483648,deny)
```

To verify that the resource control is active, the `id(1M)` and `prctl(1)` commands can be used.

```
# su - oracle
$ id -p
uid=100(oracle) gid=100(dba) projid=100(group.dba)
$ prctl -n project.max-shm-memory -i process $$
process: 5754: -bash
NAME    PRIVILEGE       VALUE    FLAG   ACTION
RECIPIENT
project.max-shm-memory
        privileged      2.00GB      -   deny
```

Logging in as the `oracle` user creates a new task in the `group.dba` project, causing the entry in the project database to be read and the resource control to be set. As can be seen in the fifth line of output from the `prtcl` command, a resource control limiting the maximum shared memory size for the project to 2 GB is present.

Installing Oracle and Creating the Databases

Oracle installation consists of a series of steps, including software installation and the creation of `smf(5)` services for the Oracle instances. In this example, a directory */u01* with at least 6 GB of free space is required for the Oracle software and databases. A simple database is created for each workload. Use the database identifiers listed in the table below.

Database	Database Identifier (ORACLE_SID)
Marketing	MKT
Sales	SALES

Running Oracle Instances in Different Projects

The Oracle instances must run in separate projects in order to control them as separate entities using the Solaris Resource Manager. The processes of the marketing database instance should run in project `ora_mkt`, and the processes of the sales database instance should run in the `ora_sales` project. Since the Oracle provided start scripts are not project-aware, the processes of both instances run in the default project of the Oracle user `group.dba`. To run the instances in different projects, the Oracle start scripts must be made project-aware by issuing `/usr/bin/newtask -p ora_sales` as part of the startup of the sales database instance. This moves the current process and its childeren to the `ora_sales` project.

The Service Management Facility (SMF) in the Solaris 10 OS replaces the traditional way of managing application startup and shutdown through run control scripts. SMF uses a concept called *services* to accomplish this task. An SMF service consists of a set of methods and properties that describe service behavior. Examples of methods include the start and stop methods that `smf(5)` calls to start or stop the service. Properties are used to describe the service, such as dependencies on other required services, the user to run the service as, and the project in which to run the service. Through a set of `smf(5)` commands, services can be managed in a consistent manner. See the *System Administration Guide: Basic Administration* for more information on the Service Management Facility.

To run the example Oracle database instances in separate projects, two simple SMF services must be created: a `salesdb` service and `mktdb` service.

The service for the sales database is created by importing the manifest for the service into the SMF repository. By convention, manifests for site-specific services are placed in the */var/svc/manifest/site* directory. A manifest is an XML file that defines service properties and methods. One of the properties of an SMF service is the user under which the service should run. In this example, the user is `oracle`. The project in which the service should run is also a service property. In this example, the project is `ora_sales`. The relevant part of the manifest is shown below.

```
# cd /var/svc/manifest/site
# cat salesdb.xml
[...]
        <exec_method
                type='method'
                name='start'
                exec='/u01/app/method/ora start SALES'
                timeout_seconds='0'>
                <method_context
                        project='ora_sales'>
                        <method_credential user='oracle' />
                </method_context>
        </exec_method>
[...]
```

The project attribute of the `method_context` element determines the project in which the service runs. The user attibute of the `method_credential` element determines the user under which the service runs. The manifest for the marketing database service is equivalent except that its project attibute is set to `ora_mkt`.

The start and stop methods for both services are implemented in a single shell script (*/u01/app/method/ora*). The start method calls the script with `start` as the first argument, while the stop method calls the script with `stop` as the first argument. The Oracle database identifier is passed as the second argument.

```
# cat /u01/app/method/ora
#!/bin/sh
#
# Usage: ora ‘start’ | ‘stop’ db_id
#
ORACLE_SID=$2
ORACLE_HOME=/u01/app/oracle/product/9.2.0.1.0
export ORACLE_SID ORACLE_HOME

case “$1” in
                'start')
                $ORACLE_HOME/bin/sqlplus "/ as sysdba" <<START_EOF
startup
START_EOF
                ;;
                'stop')
                $ORACLE_HOME/bin/sqlplus "/ as sysdba" <<STOP_EOF
shutdown immediate
STOP_EOF
                ;;
esac
exit 0
```

The services are created by importing the manifest and subsequently enabling the services. Note that enabling a service implies a start of the service. The `ps(1)` command can be used to verify the intances are running in different projects.

```
# svccfg import salesdb.xml
# svccfg import mktdb.xml
# svcadm enable salesdb
# svcadm enable mktdb
# ps -u oracle -o user,project,comm
    USER  PROJECT COMMAND
  oracle ora_sales ora_lgwr_SALES
  oracle ora_sales ora_smon_SALES
  oracle ora_mkt   ora_smon_MKT
  oracle ora_sales ora_pmon_SALES
  oracle ora_sales ora_dbw0_SALES
  oracle ora_mkt   ora_ckpt_MKT
  oracle ora_sales ora_ckpt_SALES
  oracle ora_mkt   ora_lgwr_MKT
  oracle ora_mkt   ora_pmon_MKT
  oracle ora_mkt   ora_dbw0_MKT
  oracle ora_sales ora_reco_SALES
  oracle ora_mkt   ora_reco_MKT
```

The processes for the marketing database instance run in the `ora_mkt` project, the processes for the Sales database instance run in the `ora_sales` project.

Controlling CPU Consumption

Now that the Oracle instances are running in different projects, the Fair Share Scheduler can be used to control CPU consumption by the instances. Because the Fair Share Scheduler is not the default scheduler, it must be enabled using the `dispadmin(1M)` command:

```
# dispadmin -d FSS
```

The `dispadmin` command configures the Fair Share Scheduler (FSS) as the default scheduler to be enabled on the next reboot. It is possible to change to the Fair Share Scheduler without a reboot by moving all processes in the TS scheduler class and the `init(1M)` process to the FSS scheduler class using the `prioctl(1M)` command. This change persists only until the next reboot, and the `dispadmin -d FSS` command is required to make the change permanent.

```
# priocntl -s -c FSS -i class TS
# priocntl -s -c FSS -i pid 1
```

The change of the scheduler class can be verified using the `ps(1)` command with the `-cafe` options. In the output below, the fourth column (marked CLS) shows that the Fair Share Scheduler (FSS) is now the scheduler for the processes:

```
# ps -cafe
     UID   PID  PPID  CLS PRI    STIME TTY         TIME CMD
    root     0     0  SYS  96   Dec 01 ?           0:01 sched
    root     1     0  FSS  29   Dec 01 ?           0:00 /etc/init -
    root     2     0  SYS  98   Dec 01 ?           0:00 pageout
    root     3     0  SYS  60   Dec 01 ?           9:45 fsflush
    root   556     1  FSS  29   Dec 01 ?           0:00 /usr/lib/saf/sac -t 300
...
  oracle  1967     1  FSS  29 11:03:35 ?           0:00 ora_dbw0_MKT
  oracle  1971     1  FSS  29 11:03:36 ?           0:00 ora_ckpt_MKT
  oracle  2002     1  FSS  29 11:03:47 ?           0:01 ora_smon_SALES
  oracle  1973     1  FSS  29 11:03:36 ?           0:01 ora_smon_MKT
  oracle  1965     1  FSS  29 11:03:35 ?           0:00 ora_pmon_MKT
  oracle  1996     1  FSS  29 11:03:46 ?           0:00 ora_dbw0_SALES
  oracle  1975     1  FSS  29 11:03:36 ?           0:00 ora_reco_MKT
  oracle  1998     1  FSS  29 11:03:47 ?           0:00 ora_lgwr_SALES
  oracle  1969     1  FSS  29 11:03:36 ?           0:00 ora_lgwr_MKT
  oracle  2000     1  FSS  29 11:03:47 ?           0:00 ora_ckpt_SALES
  oracle  1994     1  FSS  29 11:03:46 ?           0:00 ora_pmon_SALES
  oracle  2004     1  FSS  29 11:03:47 ?           0:00 ora_reco_SALES
....
```

The final step involves assigning CPU shares to the projects to control CPU consumption. Assuming that the sales database is twice as important as the marketing database, and should therefore be entitled to twice the amount of CPU resources, the number of CPU shares for the `ora_sales` project is set to twice the number of shares for the `ora_mkt` project. The other projects are assumed to be less important, and their shares remain at system assigned default values. To give the `ora_sales` and `ora_mkt` projects a higher proportion of CPU resources with respect to these projects, the CPU shares are chosen to be much larger than those for the other projects. These values entitle the `ora_sales` project to twenty times more CPU resources than the `group.dba` project, and twice as many as the `ora_mkt` project.

Project	CPU Shares
ora_sales	20
ora_mkt	10
group.dba	1 (default)
system	Unlimited
user.root	1 (default)
default	1 (default)
group.staff	1 (default)

The CPU shares are set using the `prctl(1M)` command:

```
# prctl -n project.cpu-shares -r -v 10 -i project ora_mkt
# prctl -n project.cpu-shares -r -v 20 -i project ora_sales
```

The current value of the `project.cpu-shares` resource control for a project can be checked as follows:

```
# prctl -n project.cpu-shares -i project ora_mkt
project: 101: ora_mkt
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
project.cpu-shares
        privileged         10       -   none                             -
        system          65.5K     max   none                             -
# prctl -n project.cpu-shares -i project ora_sales
project: 102: ora_sales
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
project.cpu-shares
        privileged         20       -   none                             -
        system          65.5K     max   none                             -
```

To make these values persistent, the `project.cpu-shares` resource controls must be added to the project database.

```
# projmod -sK “project.cpu-shares=(privileged,10,none)” ora_mkt
# projmod -sK “project.cpu-shares=(privileged,20,none)” ora_sales
```

```
# cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
group.dba:100:Oracle DBA:::project.max-shm-memory=(privileged,2147483648,deny)
ora_mkt:101:Oracle Marketing:oracle::project.cpu-
shares=(privileged,10,none);project.max-shm-memory=(privileged,2147483648,deny)
ora_sales:102:Oracle Sales:oracle::project.cpu-
shares=(privileged,20,none);project.max-shm-memory=(privileged,2147483648,deny)
```

Note – A project entry must be on one line. The above lines are wrapped for readability. They should be on one line.

For demonstration purposes, the `nspin` utility is used to create enough CPU demand to show the Fair Share Scheduler in action[1]. The `nspin` utility is part of the Solaris Resource Manager 1.x software, and is available for download at *http://www.sun.com/bigadmin/software/nspin/nspin.tar.gz*. To create more demand for CPU resources than are available on the 4 CPU machine used here, four copies of `nspin` are run in both the `ora_mkt` and `ora_sales` projects.

```
$ id -p
uid=100(oracle) gid=100(dba) projid=100(group.dba)
$ newtask -p ora_mkt
$ nspin -n 4 &
[1] 2059
$ newtask -p ora_sales
$ id -p
uid=100(oracle) gid=100(dba) projid=102(ora_sales)
$ nspin -n 4 &
[1] 2066
```

The `newtask(1)` command is used to switch from the default `group.dba` project to the `ora_mkt` and ora_sales projects to run `nspin`. The `prstat(1M)` command can be used to show CPU utilization per project and verify that the Fair Share Scheduler is distributing CPU resources to the projects according to their CPU shares.

1. Any application that consumes large quantities of CPU resources can be used.

```
$ prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  2069 oracle   1064K  592K run      1    0   0:01:57  25% nspin/1
  2066 oracle   1072K  664K run     18    0   0:01:31  17% nspin/1
  2067 oracle   1072K  600K cpu1    30    0   0:01:05  12% nspin/1
  2068 oracle   1072K  600K run     28    0   0:01:06  12% nspin/1
  2061 oracle   1072K  600K run     17    0   0:01:31 8.7% nspin/1
  2059 oracle   1072K  664K run     17    0   0:01:07 8.3% nspin/1
  2060 oracle   1072K  600K cpu0    24    0   0:01:06 8.2% nspin/1
  2062 oracle   1064K  592K cpu3    18    0   0:01:13 7.9% nspin/1
  2058 root     6056K 5040K cpu2    59    0   0:00:00 0.0% prstat/1

PROJID    NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
   102       11 1011M  712M    36%   0:05:40  66% ora_sales
   101       11 1011M  703M    36%   0:04:58  33% ora_mkt
     1        5   14M 9064K   0.4%   0:00:01 0.0% user.root
   100        1 2760K 1952K   0.1%   0:00:00 0.0% group.dba
     0       28   84M   23M   1.1%   0:00:23 0.0% system

Total: 56 processes, 196 lwps, load averages: 7.30, 3.09, 1.21
```

The top portion of the `prstat` display shows active processes sorted by CPU utilization. The bottom portion shows the statistics aggregated by project. The `ora_sales` project is receiving 66% of CPU resources, and the `ora_mkt` project is receiving 33%, even though both projects requested the same amount of CPU (four runnable `nspin` processes in each project). The Fair Share Scheduler allocates CPU resources according to the proportion of CPU shares of the active projects (using CPU time). The only active projects at the time are `ora_mkt` and `ora_sales`. As a result, the CPU entitlement for the `ora_sales` project equals (20/(20 + 10)) * 100 = 67%, while `ora_mkt` is entitled to (10/(20 + 10)) * 100 = 33%. This matches the actual CPU usage observed using `prstat(1M)`.

Using Extended Accounting

Resource usage per project can be obtained using the Extended Accounting facility of the Solaris OS. Accounting records can be written per process, per task or both. To obtain resource usage per project, task accounting is sufficient. Rather than summarizing all process termination records from the process accounting file, task accounting files can be used instead. This involves substantially fewer records since the task accounting files consolidate multiple process records into one task record. Because tasks usually have a long life span and task accounting records are only written at the end of a task, *interval records* can be used to obtain accurate daily accounting. An interval record writes the current task usage to the

accounting file and resets the task usage to zero. The total task usage is the sum of all interval records plus the termination record. Examples of common long running tasks include HPC jobs and database processes.

Extended Accounting is turned off by default, and must be turned on using the `acctadm(1M)` command. In this example, the accounting files are named `task`*yyymmdd*. A `cron(1M)` job is used to switch files every night at midnight. To start the extended accounting facility at system boot time, a link to */etc/init.d/acctadm* must be created in */etc/rc2.d*:

```
# acctadm -e extended task
# acctadm -f /var/adm/exacct/task`date '+%y%m%d'` task
# ln -s /etc/init.d/acctadm /etc/rc2.d/S01acctadm
```

The following script writes interval records for all tasks and then switches to a new accounting file:

```
# cat /opt/local/bin/switchexacct
#!/bin/sh
#
# Write interval record for all active tasks and switch accounting file
#
PATH=/usr/bin:/usr/sbin
wracct -i "`ps -efo taskid= | sort -u`" -t interval task
acctadm -f /var/adm/exacct/task`date '+%y%m%d'` task
```

Add the following line to the crontab of the root user to execute the `switchexacct` script at 00:00:

```
0 0 * * * /opt/local/bin/switchexacct > /dev/null 2>&1
```

The following script uses the Perl interface to `libexacct` to extract resource usage information from the extended accounting files. More information on the Perl interface to `libexacct` can be found in the *Solaris 10 Resource Manager Developer's Guide*.

The script processes the file(s) given on the command line and summarizes the CPU usage per project by selecting all task and task interval records in the file(s). Assuming that the extended accounting files conform to the */var/adm/exacct/task<yymmdd>* naming convention, a monthly report for February 2005 can be generated by running the following script.

```
# cpureport.pl /var/adm/exacct/task0502*
PROJECT           USR+SYS
default                 0
group.dba               0
ora_mkt             76945
ora_sales          116620
system                342
user.root              59

# cat cpureport.pl
#!/usr/perl5/5.6.1/bin/perl
# cpureport.pl - extract CPU usage per project from extended
# accounting files (CPU time in seconds)
use strict;
use warnings;
use Sun::Solaris::Exacct qw(:EXACCT_ALL);
use Sun::Solaris::Project qw(:ALL);

my %proj = ();

die("Usage: $0 file [file ...]\n") unless ($#ARGV >= 0);

# Process all files given on the commandline
foreach my $arg (0 .. $#ARGV) {

  my $ef = ea_new_file($ARGV[$arg], &O_RDONLY) || die(ea_error_str());

  while (my $obj = $ef->get()) {
    if ( $obj->catalog()->id() == &EXD_GROUP_TASK ||
         $obj->catalog()->id() == &EXD_GROUP_TASK_INTERVAL ) {

      my $h = $obj->as_hash();  # returns all items in this group

      my $projid = $h->{EXD_TASK_PROJID};
      $proj{$projid}{CPU_SEC} += $h->{EXD_TASK_CPU_SYS_SEC};
      $proj{$projid}{CPU_NSEC} += $h->{EXD_TASK_CPU_SYS_NSEC};
      $proj{$projid}{CPU_SEC} += $h->{EXD_TASK_CPU_USER_SEC};
      $proj{$projid}{CPU_NSEC} += $h->{EXD_TASK_CPU_USER_NSEC};
    }
  }

  if (ea_error() != EXR_OK && ea_error() != EXR_EOF) {
    printf("\nERROR: %s\n", ea_error_str());
    exit(1);
  }
}
```

```
# Calculate total CPU time (usr + sys) and round to whole seconds
# and lookup project names (invent name if lookup fails).
for my $key ( keys %proj ) {
  my $one_second = 10 ** 9;     # ns per second

  if ( $proj{$key}{CPU_NSEC} >= $one_second ) {
    my $seconds = $proj{$key}{CPU_NSEC} / $one_second;
    $proj{$key}{CPU_SEC} += $seconds;

    if ( $proj{$key}{CPU_NSEC} % $one_second >= ($one_second / 2) ) {
      $proj{$key}{CPU_SEC}++;
    }
  }

  my $name = getprojbyid($key);
  if ( defined($name) ) {
    $proj{$key}{PROJECT} = $name;
  }
  else {
    $proj{$key}{PROJECT} = "<" . $key . ">";
  }
}

# Print the CPU usage for the projects sorted by project name
printf("PROJECT           USR+SYS\n");
for my $key ( sort { $proj{$a}{PROJECT} cmp $proj{$b}{PROJECT} } keys
%proj ) {
  printf("%-16s %8d\n", $proj{$key}{PROJECT}, $proj{$key}{CPU_SEC});
}

exit(0);
```

5

Managing Resources

Some situations may be best served by partitioning available system resources, such as processors, into a number of discrete resource partitions. There are several reasons why such partitioning may be useful:

- *Enforcing hard limits on the use of a resource.* For instance, by creating a processor set and binding a process, project or zone to it, the CPU usage of the bound processes is effectively limited to the CPUs in the processor set. These processes cannot use processors outside of their set.
- *Providing a guaranteed quantity of a resource.* If an application requires a certain amount of CPU resources at all times, a processor set can be created for use by the application, thereby reserving the CPUs for application processes. Processes not bound to the set are unable to run on the processors in that set.
- *Setting expectations.* When deploying applications on a large server in phases, users may become accustomed to having fast response times as all resources are available to the application. As more applications are deployed, users may perceive a performance degradation. By partitioning the system so that the application received only the resources it needs, expectations can be set correctly from the start
- *Partitioning by function*, such as creating a partition for interactive users and a partition for batch jobs.

Processor Sets

Every system has at least one processor set, the system or default processor set that contains all of the processors in the system. Additional processor sets can be dynamically created and removed on a running system using the `psrset(1M)` command, provided that at least one CPU remains for the system processor set. Processes are bound to the default processor set by default, and can be bound to other processor sets on-the-fly. It is important to note that partitioning a system using processor sets may lead to under utilization of the server since

only processes bound to the processor set may use the processors in the set. If these processes do not use all of available CPU resources, the remaining CPU capacity in the set remains unused.

While processor sets are very useful, managing them can be a little cumbersome. System administrators must specify the physical CPU ID of the processor to add to a processor set. Since the physical ID of a CPU is hardware dependent, it varies between different hardware platforms, creating a close coupling between the processor set definition and the underlying hardware. Also, on systems that support Dynamic Reconfiguration, processors can be added and removed while the system is on-line. If a processor to be removed is used in a processor set, the system administrator must manually remove that processor from the set before the processor can be removed from the system. This requires the system administrator to have intimate knowledge of the configured processor sets and the hardware. Processor sets are referenced by a system generated ID, making it hard to remember what a specific set is used for, especially when multiple processor sets are present.

Resource Pools

The introduction of resource pools significantly enhances the ability to partition the system. Resource pools provide a mechanism to create a persistent configuration of resource sets such as processor sets. The resource pools framework removes the link between the intention of the system administrator and the underlying hardware. Instead of creating a processor set by specifying physical CPU IDs, system administrators can now create a processor set with a chosen name by specifying the number of processors required, rather than their physical IDs. As a result, the definition of the processor set is no longer tied to a particular type of hardware.

System administrators can also specify a minimum and maximum number of processors for a set. The system assigns a number of processors between these values when creating the processor set on a specific system. This allows for more generic definitions that can be shared between systems. A configuration defining a set with at least one CPU and a maximum of three CPUs could be instantiated on a two-way system as well as on a larger server with more processors. Moving the definition to the larger server does not require any adjustment by the system administrator. The number of processors in the set on the larger server could be higher, depending on other processor sets defined in the system. The resource pools framework balances the number of processors in the set within the constraints set by the administrator.

On systems that support Dynamic Reconfiguration, the framework ensures that constraints are still met when removing processors from the system. If the total number of processors drops below the minimum number required for the active configuration, the Dynamic Reconfiguration operation is denied. If one of the processors being removed is part of a

processor set, the system reconfigures all processor sets in the system so that the processor is no longer in a set. Adding CPUs to a running system also causes a reconfiguration of processor sets, depending on the constraints set by the administrator.

Multiple configurations can be defined to adapt to changing resource requirements such as seasonal workloads or different daily and nightly workloads. The appropriate configuration can be instantiated by invoking the `pooladm(1M)` command manually or from a `cron(1M)` job.

Binding Processes To Pools

Instead of binding a process to a processor set directly, a process is bound to a resource pool using the [2]`poolbind(1M)` command. A resource pool (or pool) is a logical collection of resource sets such as processor sets. While the processor set is the only type of resource set available in the Solaris OS, the resource pool abstraction allows other types of resource sets, such as memory sets, to be added in later Solaris OS versions.

A pool can optionally be associated with a scheduling class such as the Fair Share Scheduler (FSS) or the Real Time (RT) scheduling class. Processes bound to the pool are subject to that pool's scheduler, allowing the system to use different schedulers for different types of workloads. A server can be partitioned into two pools, one pool using the Fair Share Scheduler for applications, and a second pool using the Time Share scheduler (TS) for interactive users.

Multiple pools can be linked to the same resource set. As a result, it is possible to have a system with one processor set and several pools associated with the same processor set. This may not seem useful in a world with only processor sets. However, when other types of resource sets become available, it will be possible to let pools share a common processor set while giving each pool its own memory set, for instance.

The `poolbind(1M)` command allows administrators to bind processes, tasks, projects and zones to pools. A default pool binding for projects can be established by adding the `project.pool` attribute to the project entry in the project database. All processes started in the project are bound to the pool automatically. While the `project.pool` attribute designates only the default pool to bind to, specific processes in a project can still be bound to other pools if desired.

2.

Fair Share Scheduler and Processor Sets

When processor sets are present, the Fair Share Scheduler treats every processor set as a separate partition. CPU entitlement for a project is based on CPU usage in that processor set only. The CPU usage of a project in a processor set does not influence its entitlement in a different processor set. The Fair Share Scheduler calculates the proportion of CPU resources allocated to a project in a processor set by dividing the shares of the project by the number of shares of active projects in the processor set.

For example, consider a system with two processor sets, each containing one processor. Project A has two shares, and project B has one share. Both projects have enough processes to use all available CPU resources. Project B is the only one running in the first processor set. Since it is the only project in this set, project B is entitled to all CPU resources in the set. Both projects run in the second processor set. The number of active shares in this processor set is three (two from project A and one from project B). As a result, project A is entitled to $\frac{2}{3}$ of the processor set and project B is entitled to $\frac{1}{3}$. Project B's CPU use in the first processor set does not influence its entitlement in the second processor set.

Dynamic Resource Pools

In the Solaris 10 OS the resource pools facility is further extended to provide automated resource allocation based on resource demands in the system and usage objectives set by the system administrator. This relieves system administrators from deciding how to optimally partition available resources for the current workload. Previously system administrators had to manually reassign resources to adapt to changing workloads. While fairly easy for relatively static workloads, this task may be challenging in an environment with highly variable resource demands.

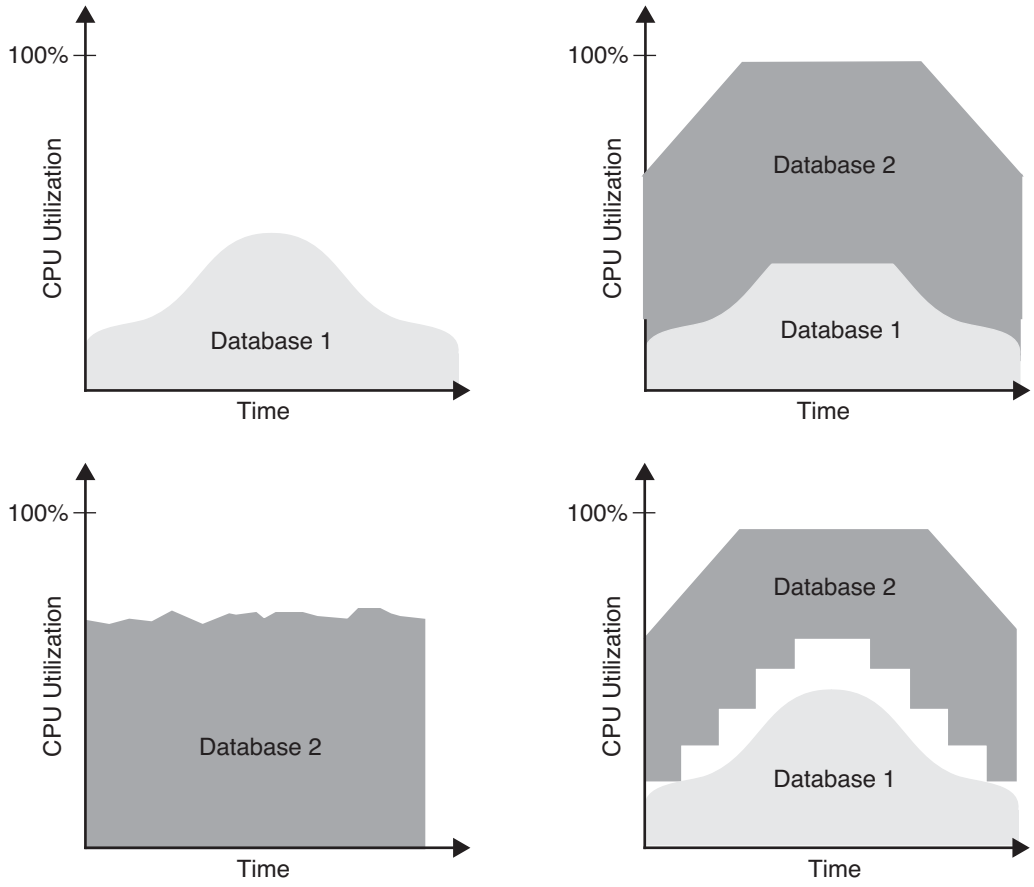

FIGURE 5-1 Dynamic resource pools let the system adapt to changing workloads

Automated Resource Allocation

The dynamic resource pools resource controller daemon `poold(1M)` is responsible for maintaining the resource allocation objectives set by system administrators. Toward this end, it creates an inventory of all available resources in the system. It continually monitors the active workloads in the system to determine if usage objectives can be met. If the resource controller detects that an objective is no longer being met, it evaluates possible alternative resource configurations to see if they can meet the objectives. If a viable alternative configuration exists, the resource controller reconfigures the resources accordingly. For processor sets, this is accomplished by moving processors between processor sets. If no alternative configuration exists that can meet objectives, no reconfiguration occurs. An appropriate message is logged, and the resource controller resumes workload monitoring.

Adding or removing resources using Dynamic Reconfiguration can also trigger a reconfiguration by the resource controller as the amount of available resource changes. Adding CPU capacity to a constrained system may create the opportunity for the resource controller to create a configuration that can meet objectives. Likewise, removing CPU capacity from the system may lead to the objectives no longer being met by the configuration.

Changes made to the objectives themselves by system administrators can also cause the resource controller to re-evaluate the configuration. The resource controller keeps a history of decisions made in the past, enabling it to rule out configuration changes that did not lead to improvement.

Even if the process of reconfiguration is automatic, system administrators can still directly manipulate the active configuration by transferring processors from one set to another. Note that doing so may or may not trigger actions by the resource controller.

Configuration Objectives

The resource controller offers several configuration objectives to influence decisions regarding possible resource configurations. These objectives can be combined and objectives can be assigned a precedence over each other. System administrators can choose from a number of different configuration objectives:

- `wt-load`
  This objective favors configurations that match resource allocations to resource demands. When this objective is in effect, a resource set that uses more resources is given more resources (within the minimum and maximum properties for the set).
- `locality`
  This objective is used to instruct the resource controller to take resource locality into consideration when allocating resources. On large servers such as the Sun Fire™ 15K server, the latency between resources on the same board and on a different board can vary. Depending on the application, latency may or may not be important. The locality objective allows the administrator to express the need for resource locality.
- `utilization`
  This objective favors configurations that allocate resources to partitions that are not meeting their utilization objective. System administrators can set target utilizations on the set using “less than”, “greater than” and “about” operators. The “less than” and “greater than” objectives can be combined to specify a target utilization range, such as between 50% and 80% utilization.

The configuration objectives are detailed in the `libpool(3LIB)` manual page.

Monitoring Resource Pools

System resource utilization can be monitored using the `poolstat(1M)` utility. This utility shows statistical data for every pool in the system. Data displayed includes the minimum, maximum and current size of the resource set, a measure of how much of the resource set is currently in use, as well as the load on the resource set.

The decisions made by the resource controller can be observed by consulting the */var/log/pool/poold* log file .

Commands

The following commands are available to administer resource pools:

Command	Description
`pooladm(1M)`	Activate and deactivate the pools facility
`poolcfg(1M)`	Create and modify resource pool configuration files
`poold(1M)`	Monitors resource usage and adjusts resource allocation
`poolbind(1M)`	Bind processes, tasks, projects and zones to a pool
`poolstat(1M)`	Report active pool statistics

Resource Pools — An Example

This section presents an example of using resource pools to partition the available CPU resources on a system. Partitioning enables minimum and maximum amounts of CPU resources to be guaranteed to applications. Continuing with the sales and marketing database example presented earlier, assume the following policies. The sales database instance should always have at least 2 CPUs available to ensure a minimum level of service. Extra CPU capacity could increase service levels and the sales business unit is willing to pay extra for increased service levels. The marketing database requires at least one CPU, and a maximum of two CPUs, to achieve business objectives. The marketing business unit is not willing to be charged for more than two CPUs. These policies should require no manual intervention by the system administrator to adjust the number of CPUs in the processor sets.

Dynamic resource pools can be used to implement these requirements by creating a large processor set with at least two CPUs for the sales database, and a small processor set with at least one CPU and at most two CPUs for the marketing database. All remaining CPUs remain in the default processor set present on every system and which contains at least one CPU. When implemented on a system with six CPUs, the following configurations are possible:

Default Processor Set	Small Processor Set	Large Processor Set
2	2	2
1	2	3
1	1	4
2	1	3
3	1	2

The number of CPUs in each processor set can be dynamically adjusted to current system load according to allocation objectives set by the system administrator. For example, if high demand is experienced for CPU resources in the large processor set, the system might move processors from the small or default processor sets to the large processor set. When demand in the large processor set decreases, the system may move processors to the small or default processor sets.

Because the pools facility is disabled by default, pools must first be enabled using the `-e` (enable) option of the `pooladm(1M)` command. This creates a configuration with a processor set with all processors in the system and a default pool. The following output illustrates the configuration of a system with 6 CPUs after the `pooladm -e` command is run, and shows the default pool named `pool_default` and the default processor set `pset_default`.

```
# pooladm -e
# pooladm
system blondie
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 611

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 447
                uint    pset.size 6
                string  pset.comment

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 11
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 10
                        string  cpu.comment
                        string  cpu.status on-line
```

While the set currently contains six CPUs, the minimum (one) and maximum (65,536) number of CPUs are also set. Note the `system.bind-default`, `pool.default` and `pset.default` properties. These properties ensure that processes that do not bind to a specific pool are bound to the `pool.default` pool.

Creating a Pool

For the sales database, a processor set named `large` with at least two CPUs, and no upper bound on the number of CPUs, is created. Next, a processor set named `small` with at least one CPU and a maximum of two CPUs is created. A pool named `sales` is created and associated with the large processor set. A second pool named `marketing` is created and associated with the small processor set. Changes to the pools configuration can be made in two ways: to the active in-kernel configuration or to the */etc/pooladm.conf* configuration file. The configuration contained in the */etc/pooladm.conf* file can be instantiated by running the `pooladm -c` command. If desired, an alternate filename can be specified using the `-f` option. To save the currently active in-kernel configuration to a file, the `pooladm -s` command can be used. In this example, changes are made to the */etc/pooladm.conf* configuration file, ensuring the changes persist across system reboots.

The initial configuration file is created from the running configuration, after which the processor sets and pools are added.

```
# poolcfg -c 'create pset large (uint pset.min=2;uint pset.max=65536)'
# poolcfg -c 'create pset small (uint pset.min=1;uint pset.max=2)'
# poolcfg -c 'create pool sales'
# poolcfg -c 'create pool marketing'
# poolcfg -c 'associate pool sales (pset large)'
# poolcfg -c 'associate pool marketing (pset small)'
```

These commands update the configuration contained in the */etc/pooladm.conf* file, and have no effect on the active in-kernel configuration. This can be verified by displaying the active in-kernel configuration using the `poolcfg(1M)` command with the `-d` option.

Next, the configuration file is instantiated on the system. The processor set and the pool are created, and the system moves processors into the created processor set according to the available processors on the system and the `pset.min` and `pset.max` attributes of the configured processor sets. The in-kernel configuration now contains the following.

```
# pooladm -c
# poolcfg -dc info
system blondie
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 611

        pool marketing
                int     pool.sys_id 1
                boolean pool.active true
                boolean pool.default false
                int     pool.importance 1
                string  pool.comment
                pset    small
        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default
        pool sales
                int     pool.sys_id 2
                boolean pool.active true
                boolean pool.default false
                int     pool.importance 1
                string  pool.comment
                pset    large
        pset large
                int     pset.sys_id 1
                boolean pset.default false
                uint    pset.min 2
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 0
                uint    pset.size 2
                string  pset.comment
                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line
                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line
```

```
 pset small
                int     pset.sys_id 2
                boolean pset.default false
                uint    pset.min 1
                uint    pset.max 2
                string  pset.units population
                uint    pset.load 0
                uint    pset.size 2
                string  pset.comment

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line

pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 4
                uint    pset.size 2
                string  pset.comment

                cpu
                        int     cpu.sys_id 11
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 10
                        string  cpu.comment
                        string  cpu.status on-line
```

Binding to a Pool

The sales database project is bound to the `sales` pool by adding the `project.pool` attribute to the project entry for the `ora_sales` project. Every new process started in this project is bound to the `sales` pool by default.

```
# projmod -sK "project.pool=sales" ora_sales
# projmod -sK "project.pool=marketing" ora_mkt
# cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
group.dba:100:Oracle DBA:::project.max-shm-memory=(privileged,2147483648,deny)
ora_mkt:101:Oracle Marketing:oracle::project.cpu-
shares=(privileged,10,none);project.max-shm-
memory=(privileged,2147483648,deny);project.pool=marketing
ora_sales:102:Oracle Sales:oracle::project.cpu-
shares=(privileged,20,none);project.max-shm-
memory=(privileged,2147483648,deny);project.pool=sales
```

Existing processes in the project are still bound to the default pool; they can be moved to the `sales` pool using the `poolbind(1M)` command. The following command binds all processes currently running in the project `ora_sales` to the `sales` pool. Start a new process in the `ora_sales` project to verify the pool binding.

```
# poolbind -p sales -i project ora_sales
# su - oracle
$ newtask -p ora_sales
$ id -p
uid=100(oracle) gid=100(dba) projid=100(ora_sales)
bash-2.05b
$ poolbind -q $$
1520    sales
```

Transferring CPUs

The system creates processor sets on a particular system based on the pool configuration and the number of CPUs in the system. In this example using a six CPU system, all three processor sets are created with two CPUs. The system administrator can manually move processors from one processor set to another to shrink or enlarge a processor set depending on the CPU requirements of applications. For example, end of month processing may require the `large` pool to contain four CPUs. The extra CPUs can be moved from the `small` and `default` processors sets using the `poolcfg(1M)` command:

```
# poolcfg -dc 'transfer 1 from pset pset_default to large'
# poolcfg -dc 'transfer 1 from pset small to large'
```

Adapting to Load

So far, the pool configuration is static. Changes in system load do not lead to configuration changes. The system administrator must manually move processors between sets to react to changes in utilization. By setting an objective, the system administrator tells the system to adapt the number of processors in a set to system demand. In this example, the utilization objective is used to ensure utilization of the `large` and `small` processor sets is kept below 75 percent to allow for spikes in the load.

```
# poolcfg -dc 'modify pset large (string pset.poold.objectives="utilization<75")'
# poolcfg -dc 'modify pset small (string pset.poold.objectives="utilization<75")''
# poolcfg -dc info
[...]
        pset large
                int     pset.sys_id 1
                boolean pset.default false
                uint    pset.min 2
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 182
                uint    pset.size 2
                string  pset.comment
                string  pset.poold.objectives utilization<75
[...]
```

Note – Until a patch for bug 6232648 is available, a workaround is needed for utilization objectives. Each processor set should have at least one 'pinned' CPU to prevent the issue described in the bug from occuring. The following command can be used to pin a CPU in a processor set. (Replace ID with the appropriate CPU ID.)

```
# poolcfg -dc 'modify cpu ID (boolean cpu.pinned=true)'
```

To see how the system adapts to varying demand for CPU resources, load is generated in the `small` processor set. It currently contains only one CPU since the second CPU was moved by the administrator in *Transferring CPUs* earlier. When the load exceeds the 75 percent utilization objective, the system attempts to move a processor from another processor set into the `small` processor set.

```
$ id -p
uid=100(oracle) gid=100(dba) projid=101(ora_mkt)
$ /usr/sbin/poolbind -q $$
666 marketing
$ nspin -n 4
```

The */var/log/pool/poold* file can be observed to see the actions taken by the resource controller, such as moving a processor from one processor set to another:

```
Mar 22 15:28:33 Monitoring INFO: all evaluated objectives satisfied
Mar 22 15:28:48 Monitoring INFO: all evaluated objectives satisfied
Mar 22 15:29:03 Monitoring INFO: pset small utilization objective not satisfied (1,
utilization, '<', 75) with utilization 85.99 (control zone bounds exceeded)
Mar 22 15:29:03 Monitoring INFO: reconfiguration required
Mar 22 15:29:03 Optimization INFO: from pset large to pset small components [cpu 2]
not applied due to poor past results
Mar 22 15:29:03 Optimization INFO: applying move from pset large to pset small
components [cpu 1]
Mar 22 15:29:03 Configuration INFO: reconfiguring...
Mar 22 15:29:03 Configuration INFO: configuration complete
```

As shown the above output, the system decides to move processor 1 from the `large` processor set to the `small` processor set to satisfy utilization objectives. Stopping the load in the `small` processor set and adding load in the `large` processor set causes another reconfiguration after some time to satisfy the utilization objective on the `large` processor set.

This example only shows a tiny fraction of the possibilities enabled by dynamic resource pools. More complex objectives can be created, such as combining different types of objectives and setting the importance of these objectives in relation to each other. See the `libpool(3LIB)` `man` page for more information on setting objectives.

Saving the Dynamic Configuration

The last few changes have been made to the in-kernel configuration. To keep these changes across reboots, the in-kernel configuration must be saved to a file using the `pooladm -s` command. This command saves the configuration to the */etc/pooladm.conf* file. The system automatically instantiates the configuration from this file at boot time.

6

Isolating Applications

Solaris Zones provide a means to create one or more virtual environments on a single operating system instance, shielding applications from details of the underlying hardware. Applications in a zone run in isolation from applications in other zones. They cannot see, monitor or affect processes running in another zone. Zones provide the following features:

- *Security*

  Network services can be run in a zone, limiting the damage that can be done to the system and other zones in case of a security violation. An intruder that is able to exploit a security hole in an application running in a zone can only do limited damage. The actions possible in a zone are restricted to subset of what is allowed in a normal system. For instance, it is not possible to load custom kernel modules, access kernel memory or create device nodes inside a zone.

- *Isolation*

  Applications requiring exclusive access to global resources, such as specific usernames or network ports, can run on the same machine using zones. Each zone has its own namespace, completely separate from other zones. Users in a zone are unable to monitor other zones, such as viewing network traffic or the activity of processes.

- *Virtualization*

  Zones present a virtualized environment to applications, removing the physical details of the hardware from view. This eases redeployment of applications on a different physical machine.

- *Granularity*

  Since zones are implemented in software, zones are not limited to granularity defined by hardware boundaries. Instead, zones offer sub-CPU granularity. Zones do not require dedicated CPU resources, dedicated I/O devices such as HBAs and NICs, or dedicated physical memory. As a result, even a system with a single processor can be used to host several zones.

- Transparency

  The environment presented to the application in a zone is nearly identical to the standard Solaris OS environment. There are no new, zone-specific APIs or ABIs to which applications must be ported. Some restrictions do exist due to security and isolation requirements. These restrictions mainly affect applications that perform privileged operations or need access to physical devices.

Zones Overview

The *global zone* encompasses the entire system and is comparable to a normal Solaris OS instance. It has access to the physical hardware and can see and control all processes. The administrator of the global zone can control the system as a whole. The global zone always exists, even when no other zones are configured. Inside the global zone are *non-global zones*. These zones are isolated from the physical hardware characteristics of the machine by the virtual platform layer. This layer provides the zones with a virtual network interface, one or more file systems and a virtual console.

Even though the virtual network interfaces may map to the same physical network interface, applications in different zones are prevented from seeing traffic from applications in other zones. Every zone has its own process environment and runs its own set of core Solaris OS services, including `inetd(1M)`, `syslogd(1M)`, `rpcbind(1M)`, and more. Applications running in a zone are unable to see applications running in other zones because of this private process environment. Zones are confined to their own subtree in the file system hierarchy and cannot access file systems of other zones or the global zone. All zones share the same operating system instance and therefore run the same Solaris OS version.

The virtual platform layer is not an emulation layer that translates requests from a zone into some other form or executes them on the zone's behalf. The role of the virtual platform layer is to instantiate and to connect virtualized resources to a zone. For instance, in the case of a virtual network interface, the virtual platform layer creates a logical interface on top of the physical network interface specified in the zone configuration. The IP address from the zone configuration is configured on the logical interface and it is made available to the zone.

One of the attributes of the logical interface is the zone in which it is configured. The kernel uses this attribute to virtualize the network interface by passing packets to the appropriate zone based on this attribute. A zone only sees packets that are destined for its own logical interfaces. Broadcast or multicast packets are replicated and sent to all zones as appropriate.

Virtualization of file systems in a zone is achieved via a restricted root similar to `chroot(2)`. A process in a zone is limited to files and file systems that can be accessed from the restricted root. Unlike chroot, a zone is not escapable. The virtual platform layer is responsible for creating the restricted root and mounting the file systems defined in the zone configuration on it.

Process isolation is accomplished by adding a reference to the zone to the process credentials. The kernel has been extended to use the zone ID as a means to restrict visibility of other processes. Only processes with the same zone ID are visible to a process in a zone. This selection is made *inside* the kernel and not available in utilities such as `ps(1)` or `kill(1)`, as that would make it possible to subvert the isolation by writing a `ps(1)` replacement.

As the zone ID is part of the credentials, the user ID namespace is also virtualized in zones. Every zone has its own user ID namespace. As a result, users in different zones with the same uid are in fact distinct users, even though they share the same numerical id. The virtualized user ID namespace also implies that passwords are unique to the zone.

The introduction of Least Privilege in the Solaris OS provides a set of fine-grained privileges to replace the concept of the omnipotent root user. Instead of performing checks against uid 0 to allow privileged operations, the kernel now checks for specific privileges required to perform privileged operations. In the past, it was sufficient to be the superuser to perform mount operations. Now, even the root user must have a specific privilege to perform mount operations. By restricting the privileges of root in the local zone to a set of privileges that are safe in a zone, the root user in a local zone can be given enough power to manage the zone without the ability to affect the system as a whole, such as rebooting the system. Restricting privileges by itself is not sufficient for isolation. Privileges only restrict the operations that can be performed, not the objects on which they are performed. This is accomplished by the isolation that zones provide.

It is therefore possible to delegate local zone administration to users by giving them access to the root account in a local zone. Since a user with uid 0 in one zone is different from a user with uid 0 in another zone, a local zone root user cannot compromise any other zone. However, the global zone root user should still be closely guarded as it has control over the system as a whole, and as such has access to all zones.

Administering Zones

Zone administration tasks can be divided into two parts, *global* zone administration tasks such as creating a zone, and *non-global* zone administration tasks such as performing configuration *within* a zone. The four primary global zone administration tasks are:

- *Configuration* — the global administrator defines the virtual platform properties, such as the required file systems and network interfaces
- *Installation* — the global administrator creates the zone on the system by creating and populating the part of the file system hierarchy reserved for the zone
- *Virtual Platform Management* — the global administrator uses zone tools to boot, halt or reboot the zone

- *Zone Login* — the global administrator can move in and out of the local zone from the global zone to assume the role of the non-global zone administrator

Zone Configuration

The first step of creating a zone on a system is defining its configuration using the `zonecfg(1M)` command. The configuration describes resources and properties that are required for the zone to operate, including:

- *Zone name* — A unique name for the zone. This name is only of interest in the global zone, it is distinct from the nodename of the zone. The name `global` and names starting with `SUNW` are reserved.
- *Zone path* — Every zone has a path to its root relative to the global zone's root directory. The zone's root path will be one level deeper; the `root` directory under the zone path. To prevent unprivileged users in the global zone from traversing into the file system hierarchy of the zone, the zone path must be owned by root with mode 700. The zone `root` directory should be owned by root and have mode 755 like a regular root directory.
- *File systems* — a zone may have file systems that should be mounted when the zone is booted
- *Network interfaces* — a zone may have one or more virtual network interfaces. The configuration specifies the IP address and the physical interface in the global zone on which it is to be created.
- *Devices* — a zone may require devices that need to be configured when the zone is booted. A default set of devices required in every zone is provided and can be imported into the configuration
- *Resource controls* — a zone may have resource controls that should be enabled when the zone is booted.
- *Properties* — a zone can have properties that control some aspect of the zone, such as the `autoboot` property. The `autoboot` property is comparable to the `auto-boot?` OBP parameter and determines if the zone is to be booted automatically when the global zone is booted.

Installing Zones

The zone configuration process is only concerned with the syntactic correctness of the configuration: it determines if the configuration could be created on *some* system, not necessarily *this* particular system. This is verified when the zone is installed on a system using the `zoneadm(1M) install` command. This command checks to see if all resources, such as the physical network interface specified in the configuration, are available. It then installs the files needed for the zone's root file system in the correct location under the zone

path, and creates the mount points for additional file systems specified in the configuration. When the installation is complete, the zone is ready to be booted. The root file system of a typical zone requires approximately 100 MB of disk space.

Virtual Platform Management

The virtual platform layer is implemented by the `zoneadmd(1M)` daemon. When the global administrator boots a zone using the `zoneadm boot` command, an instance of `zoneadmd` is created for that zone. The `zoneadmd` instance for the zone consults the zone configuration and creates a new zone. It then creates the virtual network interfaces, mounts the file systems, and creates the zone virtual console. Finally it starts an instance of `init(1M)` in the zone to further bringup the zone using SMF.

The global zone administrator can halt the zone using `zoneadm halt` and reboot the zone using `zoneadm reboot`. These commands do not perform an orderly shutdown when bringing down the zone. These operations can be considered the equivalent of the `setkeyswitch` operations on the system controller of larger Sun Fire systems. If an orderly shutdown of the zone is required, an `init 0` command should be done from inside the zone to ensure the stop methods of `smf(5)` services are executed, just like a domain on a Sun Fire server would be shutdown prior to running the `setkeyswitch off` command.

The `svc:/system/zones` `smf(5)` service in the global zone is used for zone startup and shutdown. All zones with the autoboot property set are started automatically when the global zone boots. When the global zone is stopped, the zones service performs an `init 0` in each running zone to do an orderly shutdown of services in the zone.

Zone Login

Since zones are implemented in software and require no dedicated hardware, the virtual platform provides a virtual console for each zone. The virtual console can be accessed from the global zone using the `zlogin(1M)` command with the `-C` option. The console for a zone cannot be shared by multiple users at the same time. The first user to connect to the console of a zone has exclusive access until disconnecting from the console. Access to the consoles of *other* zones is still possible.

When using the `zlogin` command in interactive, non-console mode an arbitrary number of `zlogin` sessions to the same zone may be open concurrently. Non-interactive `zlogin` is also possible, for example to allow the global administrator to perform scripted administration in zones. The use of zlogin requires a specific privilege.

Traditional remote login facilities such as `telnet(1)`, `rlogin(1)` and `ssh(1)` can still be used, provided they have not been disabled by the local zone administrator.

Commands

The following commands were added to the Solaris OS for zones:

Command	Description
`zonecfg(1M)`	Create, update and view zone configuration
`zoneadm(1M)`	Administer zones
`zlogin(1)`	Login to a zone from the global zone
`zonename(1)`	Print the name of the current zone
`zoneadmd(1M)`	Zones administration daemon

Several existing Solaris OS commands are enhanced to support zones:

Command	**Description**
`prstat(1M)`	`-Z`, `-z` options added
`ps(1)`	`-o` *zone*, `-o` *zoneid* options added
`pgrep(1)`	`-z` option added
`pkill(1)`	`-z` option added
`priocntl(1)`	`-i` option added
`renice(1)`	`-i` option added
`ipcs(1)`	`-Z`, `-z` option added
`ipcrm(1)`	`-z` option added
`ppriv(1)`	`-z` option added
`ifconfig(1M)`	`-Z` option added
`poolbind(1M)`	`-i` option added
`df(1)`	`-Z` option added
`coreadm(1M)`	`%z` token added

Zone Administration

Administering zones is largely the same as administering a normal Solaris-based system. Some of the administrative facilities are not available in a zone because they do not apply to zones, such as device configuration.

After the initial install of the zone by the global administrator, the system is in an unconfigured state. The normal sysid tools are run on the first zone boot to finish zone configuration. The prompts from these tools can be bypassed completely by creating a */etc/sysidcfg* onfiguration file with all required parameters in the zone. Each zone runs its own core Solaris OS services, and therefore has its own name service. Consequently, one zone can use NIS, while another zone can use LDAP, DNS, or local files for the name service.

File Systems

A number of file systems are mounted in the zone by the virtual platform layer when the zone is booted. In the default configuration, the following file systems are mounted in a zone:

- /, the zone root file system is mounted on <*zonepath*>*/root* in the global zone
- */sbin*, */usr*, */lib* and */platform* file systems are read-only loopback mounts from the global zone to enable text page sharing to reduce memory requirements. This also reduces the required disk footprint of the zone.
- */dev* for the zone is mounted on <*zonepath*>*/dev* in the global zone
- */proc*, */dev/fd*, */system/contract, /etc/svc/volatile, /etc/mnttab*, */var/run* and */tmp*

Additional file systems can be mounted in a zone if required, either by adding these file systems to the zone configuration using the `zonecfg(1M)` command, mounting them from within the zone through the local zone's */etc/vfstab* file, mounting them by hand, or triggered by autofs.

UFS file systems to be mounted in the zone are configured using the `zonecfg(1M)` command by adding file system resources of type `ufs`. When the zone is booted, the system automatically mounts these file systems in the zone. This is the recommended and safest way to add UFS file systems to a zone.

Mounting UFS file systems from within the zone requires that the character and block device for the file system be explicitly exported to the zone by the global administrator. The file system can then be created by the local zone administrator. It is important to note that this may expose the system as a whole to the risk of failure as the local administrator has access to the raw device and can possibly induce a panic by corrupting the UFS metadata on the disk. It is, therefore, not recommended to mount file systems in this manner.

Alternatively, the global administrator can create a UFS file system somewhere in the global zone and export this file system as a lofs mount to the zone. This prevents the local administrator from exposing the system to the risk described above.

NFS file systems can be mounted in the local zone, provided that they do not originate from the global zone, since it is not possible for a NFS server to mount its own file systems. See the `mount_nfs(1M)` manpage for more information. Therefore, zones can be NFS clients of any server except the global zone. Currently zones cannot be NFS servers.

Resource Management

The regular resource management facilities such as resource controls, projects, and more are available inside zones. Because projects are also virtualized inside a zone, each zone has its own project database. This allows a local zone administrator to configure projects and resource controls local to that zone.

Resource Pools

A zone can be bound to a resource pool through the `pool` property of the zone configuration. The zone is bound to the specified pool upon creation of the zone. The pool configuration can only be changed from the global zone. A zone cannot change the pool to which it is bound. Zones cannot span multiple pools, therefore, all processes in a zone run in the same pool. It is however possible to bind multiple zones to the same resource pool.

Extended Accounting

The Extended Accounting framework has been extended for Zones. Each zone has its own extended accounting files for task- and process-based accounting that contain accounting records exclusively for the zone. The extended accounting files in the global zone contain accounting records for the global zone and all local zones. Since the Extended Accounting framework uses a flexible and extensible format, the accounting records are now tagged with the zone name to which they apply. This allows the global zone administrator to extract per zone accounting data from the accounting files in the global zone.

Fair Share Scheduler and Zones

To prevent a local zone from monopolizing the system, the global zone administrator can set zone-wide resource controls. The `zone.cpu-shares` resource control limits the amount of CPU resources a zone is entitled to in the same way that the Fair Share Scheduler does this for projects. A zone with a higher number of `zone.cpu-shares` is allowed to use more CPU resources than a zone with a low number of shares. The `zone.cpu-shares` resource control is configured using the `zonecfg(1M)` command. Note that this requires the Fair Share Scheduler to be the default scheduler for the system, or that the zones be bound to a pool with a processor set with FSS as the scheduler.

Combined with the regular Fair Share Scheduler inside a zone, this leads to a two-level distribution of CPU resources. First, the `zone.cpu-shares` configured by the global zone administrator determine the amount of CPU resources to which a zone is entitled. The amount of CPU resources available to the zone is then further distributed across the active projects in the zone according to the `project.cpu-shares` defined by the local zone administrator.

Resource Controls

All standard resource controls are available inside the local zone and can be used by the local zone administrator to perform resource management in the zone. The global zone administrator can limit the number of lightweight processes (LWPs) created by a zone by setting the `zone.max-lwps` resource control on a zone.

Using Zones — An Example

The following example demonstrates the features provided by zones that facilitate consolidation. It shows how to run the two Oracle workloads from the *Managing Workloads* example in a Solaris Container using zones. In that example, both workloads shared the same physical system as well as the file system namespace, name service, network port namespace, user and group namespaces, and more. The sharing of these namespaces can lead to undesirable and sometimes difficult to manage situations, such as when the databases are managed by two different DBA groups. The fact that there is only one `oracle` user requires close coordination between the DBA groups, since changes made to that user's environment by one DBA group may impact the other database instance. The same holds true for the sharing of the file system namespace, where a single */var/opt/oratab* file is used by multiple Oracle instances.

Sharing namespaces can also inhibit the consolidation from a large number of servers onto fewer systems. Existing procedures and scripts may, for example, assume the system is dedicated to the application. Making changes to these procedures and scripts may be difficult, costly or even impossible.

Solaris Zones help resolve these issues because each zone is a virtualized environment with its own private namespaces that can be managed independently of other zones on the system. For instance, the `oracle` user in one zone is a completely different user from the `oracle` user in another zone — they can have different uids, passwords, login shells, home directories, etc. By running each Oracle instance in its own zone, the instances can be completely isolated from each other, simplifying their management. As far as the Oracle instance is concerned, it still runs on a dedicated system.

Requirements

Two zones each running their own Oracle instance are created. The zones require approximately 100 MB of disk space, and the Oracle software and a database each require about 4 GB of disk space.

Note – In this chapter, the prompt is set to the zone name to distinguish between the different zones.

Preparation

The Oracle instances for the sales and marketing databases are recreated in Zones in this example. Consequently, the instances previously created should be stopped and the associated user, projects, and file system should be deleted. The pool configuration previously built should be disabled.

```
global # svcadm disable salesdb
global # svcadm disable mktdb
global # svccfg delete salesdb
global # svccfg delete mktdb
global # userdel -r oracle
global # projdel ora_sales
global # projdel ora_mkt
global # projdel group.dba
global # pooladm -x
global # pooladm -d
```

Creating the First Zone

The zone used for the marketing database is named `mkt`. To show how a file system is added to a zone, a separate file system is created on a SVM soft partition (`d200`). The file system may, of course, also be created on a standard disk slice. The virtual network interface for the zone with IP address 192.168.1.14 is configured on the physical interface `hme0` of the system. The directory for the zone is created in the global zone by the global zone administrator. The directory used for the zone must be owned by `root` and have mode `700` to prevent normal users in the global zone from accessing the zone's file system.

```
global # mkdir -p /export/zones/mkt
global # chmod 700 /export/zones/mkt
global # newfs /dev/md/rdsk/d200
```

Configuring the Zone

The zone is created based on the default template that defines resources used in a typical zone.

```
global # zonecfg -z mkt
mkt: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:mkt> create
zonecfg:mkt> set zonepath=/export/zones/mkt
zonecfg:mkt> set autoboot=true
```

The virtual network interface with IP address 192.168.1.14 is configured on the `hme0` interface of the global zone.

```
zonecfg:mkt> add net
zonecfg:mkt:net> set address=192.168.1.14/24
zonecfg:mkt:net> set physical=hme0
zonecfg:mkt:net> end
```

The file system for the Oracle binaries and datafiles in the `mkt` zone is created on a soft partion named `d200` in the global zone. Add the following statements to the zone configuration to have the file system mounted in the zone automatically when the zone boots:

```
zonecfg:mkt> add fs
zonecfg:mkt:fs> type=ufs
zonecfg:mkt:fs> set type=ufs
zonecfg:mkt:fs> set special=/dev/md/dsk/d200
zonecfg:mkt:fs> set raw=/dev/md/rdsk/d200
zonecfg:mkt:fs> set dir=/u01
zonecfg:mkt:fs> end
zonecfg:mkt> verify
zonecfg:mkt> commit
zonecfg:mkt> exit
```

The zone configuration is now complete. The `verify` command verifies that the current configuration is syntactically correct. The `commit` command writes the in-memory configuration to stable storage.

Installing the Zone

The zone is now ready to be installed on the system.

```
global # zoneadm -z mkt install
Preparing to install zone <mkt>.
Checking <ufs> file system on device </dev/md/rdsk/d200> to be mounted
at </export/zones/mkt/root>
Creating list of files to copy from the global zone.
Copying <2584> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <916> packages on the zone.
Initialized <916> packages on zone.
Zone <mkt> is initialized.
The file </export/zones/mkt/root/var/sadm/system/logs/install_log>
contains a log of the zone installation.
```

Booting the Zone

The zone can be booted with the `zoneadm boot` command. Since this is the first time the zone is booted after installation, the standard system identification questions must be answered, and are displayed on the zone's console. The console can be accessed from the global zone using the `zlogin(1M)` command.

```
global # zoneadm -z mkt boot
global # zlogin -C mkt
[Connected to zone 'mkt' console]

SunOS Release 5.10 Version Generic 64-bit
Copyright 1983-2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Hostname: mkt
Loading smf(5) service descriptions: 100/100
```

At this point, the normal system identification process for a freshly installed Solaris OS instance is started. The output of this process is omitted here for brevity, and the configuration questions concerning the name service, time zone, etc., should be answered as appropriate for the site. After system identification is complete and the root password is set, the zone is ready for use.

```
SunOS Release 5.10 Version Generic 64-bit
Copyright 1983-2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Hostname: mkt

mkt console login:
```

To disconnect from the console use `~.` (tilde dot) just like in `tip(1)`. The zone can now be accessed over the network using the `telnet(1)`, `rlogin(1)` or `ssh(1)` commands, just like a standard Solaris OS system. (Note that root can only login at the console unless the */etc/default/login* file is updated).

```
mkt console login: root
Password:
Last login: Tue Mar 22 21:55:00 on console
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
# df -h
Filesystem             size   used  avail capacity  Mounted on
/                      7.9G   4.6G   3.2G    60%    /
/dev                   7.9G   4.6G   3.2G    60%    /dev
/lib                   7.9G   4.6G   3.2G    60%    /lib
/platform              7.9G   4.6G   3.2G    60%    /platform
/sbin                  7.9G   4.6G   3.2G    60%    /sbin
/u01                   7.9G   8.0M   7.8G     1%    /u01
/usr                   7.9G   4.6G   3.2G    60%    /usr
proc                     0K     0K     0K     0%    /proc
ctfs                     0K     0K     0K     0%    /system/contract
swap                    15G   272K    15G     1%    /etc/svc/volatile
mnttab                   0K     0K     0K     0%    /etc/mnttab
fd                       0K     0K     0K     0%    /dev/fd
swap                    15G     0K    15G     0%    /tmp
swap                    15G    24K    15G     1%    /var/run
```

The */lib*, */platform*, */sbin*, and */usr* file systems are read-only loopback mounts from the global zone. This reduces the required disk space for the zone considerably, and allows the sharing of text pages, leading to more efficient use of memory. These file systems appear in the zone because they are defined in the default template used to create this zone. All other file systems are private to the zone. The */u01* file system is mounted in the zone during zone boot by `zoneadmd`. It is not mounted by the zone itself. Also note that the zone is unaware that the file system is in fact residing on */dev/md/dsk/d200*.

Installing Oracle

The group `dba` and the user `oracle` are required to run the Oracle software. Since the Oracle software uses shared memory, and the maximum amount of shared memory is now a project resource control, a project is needed in which to run Oracle. The project `ora_mkt` project is created in the zone and the `project.max-shm-memory` is set to the required value (in this case 2 GB). Since the System V IPC parameters are resource controls in Solaris 10 OS, there is no need to update the */etc/system* file and reboot.

Note that the zone has its own namespace and that the user, group and project just created are therefore only visible inside the `mkt` zone.

```
mkt # mkdir -p /export/home
mkt # groupadd dba
mkt # useradd -g dba -d /export/home/oracle -m -s /bin/bash oracle
mkt # passwd oracle
mkt # projadd -c “Oracle” user.oracle
mkt # projadd -c "Oracle" -U oracle ora_mkt
mkt # projmod -sK "project.max-shm-memory=(privileged,2G,deny)" ora_mkt
mkt # cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
ora_mkt:101:Oracle:oracle::project.max-shm-memory=(privileged,2147483648,deny)
user.oracle:100:Oracle:::project.max-shm-memory=(privileged,2147483648,deny)
```

The Oracle software and database are installed in */u01*. In this example, the Oracle software is installed in the zone itself to create an Oracle installation independent from other Oracle installations. The software could be installed in the global zone and then loopback mounted in the local zones. Doing so allows sharing of the binaries by multiple zones, but also creates a coupling between Oracle installations with regards to patch levels and more. This example shows how to use zones to consolidate Oracle instances with maximum isolation from each other, so the software is not shared. The installation can now be performed. Since */usr* is mounted read-only in the zone, the default location */usr/local/bin* suggested by the Oracle Installer should be changed to a writable directory in the zone, such as */opt/local/bin*. The marketing database can be created using the procedure used earlier. Using the `smf` service for the marketing database from the *Managing Workloads* example, the database instance can be started by importing the manifest and enabling the `mktdb` service in the zone.

Creating the Second Zone

The first zone used a directory in */export/zones* in the global zone. Since this does not limit the size of the root file system of the local zone it could fill up the file system in the global zone, where */export/zones* is located. To prevent a local zone from creating this problem, the zone root file system is created on a separate file system. The second zone is for the sales database and requires the following resources.

- A 100 MB file system for the zone root file system mounted in the global zone on */export/zones/sales*. This file system is created on a Solaris Volume Manager soft partition (*/dev/md/dsk/d100*). A normal slice could also be used but would be quite wasteful given the limited number of slices available on a disk.
- To show how devices can be used in a zone, the disk slice `c1t1d0s3` is exported to the zone by the global zone administrator. A UFS file system is created on this slice inside the zone. This requires that both the block and character devices for the slice be exported to the zone. Note that this is for demonstration purposes only and is not the recommended way to use UFS file systems in a zone.

- A virtual network interface with IP address 192.168.1.15 on the `hme0` interface of the global zone is also needed.

```
global # newfs /dev/md/rdsk/d100
global # mkdir -p /export/zones/sales
global # mount /dev/md/dsk/d100 /export/zones/sales
global # chmod 700 /export/zones/sales
```

Configuring and Installing the Second Zone

The steps required to configure and install this zone are the same as for the first zone, with the exception that two devices are added to the zone configuration.

```
global # zonecfg -z sales
sales: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:sales> create
zonecfg:sales> set zonepath=/export/zones/sales
zonecfg:sales> set autoboot=true
zonecfg:sales> add net
zonecfg:sales:net> set physical=hme0
zonecfg:sales:net> set address=192.168.1.15/24
zonecfg:sales:net> end
zonecfg:sales> add device
zonecfg:sales:device> set match=/dev/rdsk/c1t1d0s3
zonecfg:sales:device> end
zonecfg:sales> add device
zonecfg:sales:device> set match=/dev/dsk/c1t1d0s3
zonecfg:sales:device> end
zonecfg:sales> verify
zonecfg:sales> commit
zonecfg:sales> exit
global # zoneadm -z sales install
Preparing to install zone <sales>.
Creating list of files to copy from the global zone.
Copying <2584> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <916> packages on the zone.
Initialized <916> packages on zone.
Zone <sales> is initialized.
The file </export/zones/sales/root/var/sadm/system/logs/install_log>
contains a log of the zone installation.
```

Booting the Zone

The first time a zone is booted after installation, the system identification process is performed. It is possible to skip the system identification questions during the first boot of the zone by creating a `sysidcfg` file in the zone prior to the first boot. The location of the `sysidcfg` file from the global zone is */export/zone/sales/root/etc/sysidcfg*. A sample `sysidcfg` file is shown below, and can be customized to fit the situation.

```
global # cat /export/zone/sales/root/etc/sysidcfg
system_locale=C
timezone=US/Pacific
network_interface=primary {
     hostname=hostname
terminal=xterm
security_policy=NONE
name_service=NIS {
     domain_name=yourdomain.com
}
root_password=sS3G0h84sqwJA
```

To suppress the question about the NFS version 4 domain, set the `NFSMAPID_DOMAIN` line in the */export/zones/sales/root/etc/nfs/default* file to the appropriate value for the site and create the */export/zones/sales/root/etc/.NFS4inst_state.domain* file.

The */dev/dsk/c1t1d0s3* and */dev/rdsk/c1t1d0s3* devices are added to the zone configuration to show how devices can be imported into a zone. Note that the only devices present in the */dev/dsk* and */dev/rdsk* directories are the devices that were explicitly added to the zone configuration.

```
global # zoneadm -z sales boot
global # zlogin sales
sales # ls -l /dev/dsk
total 0
brw-r-----   1 root     sys       32,  3 Mar 24 11:44 c1t1d0s3
sales # ls -l /dev/rdsk
total 0
crw-r-----   1 root     sys       32,  3 Mar 24 11:44 c1t1d0s3
```

A new file system is created and added to the zone's */etc/vfstab* file.

```
sales # newfs /dev/rdsk/c1t1d0s3
sales # mkdir /u01
sales # mount /dev/dsk/c1t1d0s3 /u01
sales # cat /etc/vfstab
#device         device          mount           FS      fsck    mount   mount
#to mount       to fsck         point           type    pass    at boot options
#
/proc           -               /proc           proc    -       no      -
ctfs            -       /system/contract        ctfs    -       no      -
objfs           -       /system/object  objfs   -       no      -
fd              -               /dev/fd         fd      -       no      -
swap            -               /tmp            tmpfs   -       yes     -
/dev/dsk/c1t1d0s3 /dev/rdsk/c1t1d0s3 /u01       ufs     2       yes nologging

sales # df -h
Filesystem             size   used  avail capacity  Mounted on
/                       94M    70M    14M    83%    /
/dev                    94M    70M    14M    83%    /dev
/lib                   7.9G   4.6G   3.2G    60%    /lib
/platform              7.9G   4.6G   3.2G    60%    /platform
/sbin                  7.9G   4.6G   3.2G    60%    /sbin
/usr                   7.9G   4.6G   3.2G    60%    /usr
proc                     0K     0K     0K     0%    /proc
ctfs                     0K     0K     0K     0%    /system/contract
swap                    15G   272K    15G     1%    /etc/svc/volatile
mnttab                   0K     0K     0K     0%    /etc/mnttab
fd                       0K     0K     0K     0%    /dev/fd
swap                    15G     0K    15G     0%    /tmp
swap                    15G    24K    15G     1%    /var/run
/dev/dsk/c1t1d0s3      4.9G   5.0M   4.9G     1%    /u01
```

Notice the difference beteen the */u01* file system in this zone and the */u01* file system in the `mkt` zone. In this zone the physical device is visible while in the `mkt` zone it is not visible.

Installing Oracle

The installation of the Oracle software is the same as that for the mkt zone. Since the zones have completely separate namespaces, the user, group and project for Oracle must be created in this zone also. The project `user.oracle` should have the resource control `project.max-shm-memory` added to it to allow Oracle access to the required shared memory.

```
sales # mkdir -p /export/home/oracle
sales # groupadd dba
sales # useradd -g dba -m -d /export/home/oracle -s /bin/bash oracle
sales # passwd oracle
sales # projadd -c "Oracle" -U oracle ora_sales
sales # projmod -sK "project.max-shm-memory=(privileged,2G,deny)" ora_sales
sales # cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
ora_sales:100:Oracle:oracle::project.max-shm-memory=(privileged,2147483648,deny)
```

The Oracle installation can now be performed. Since */usr* is mounted read-only from the global zone, the default location */usr/local/bin* suggested by the Oracle Installer should be changed to a writable directory such as */opt/local/bin*. The sales database can be created using the procedure on page 93. Using the `smf` service for the sales database the *Managing Workloads* example, the database instance can be started by importing the manifest and enabling the `salesdb` service in the zone.

Controlling CPU Consumption of Zones

The `zone.cpu-shares` resource control can be used to limit the CPU usage of zones with respect to other zones. This resource control is set through the `zonecfg(1M)` command. To give the `sales` zone twice the amount of CPU resources as the `mkt` zone, the number of `zone.cpu-shares` of the `sales` zone is set to twice the number of `zone.cpu-shares` of the `mkt` zone:

```
global # zonecfg -z sales
zonecfg:sales> add rctl
zonecfg:sales:rctl> set name=zone.cpu-shares
zonecfg:sales:rctl> add value (priv=privileged,limit=20,action=none)
zonecfg:sales:rctl> end
zonecfg:sales> exit
global # zonecfg -z mkt
zonecfg:mkt> add rctl
zonecfg:mkt:rctl> set name=zone.cpu-shares
zonecfg:mkt:rctl> add value (priv=privileged,limit=10,action=none)
zonecfg:mkt:rctl> end
zonecfg:mkt> exit
```

The resource control is made active at the next zone boot. To set the `zone.cpu-shares` resource control on a running zone the `prctl(1)` command can be used.

```
global # prctl -n zone.cpu-shares -r -v 20 -i zone sales
global # prctl -n zone.cpu-shares -r -v 10 -i zone mkt
```

To observe processes, the `prstat(1M)` command has been enhanced for zones with the `-Z` and `-z` options. The following `prstat -Z` output from the global zone shows processes running in the global and local zones. The bottom of the output shows a summary line for every running zone. Both zones are running eight instances of the `nspin` utility to show how CPU usage is controlled by the `zone.cpu-shares` resource control when contention arises for CPU resources. As can be seen from the output, the sales zone is given twice the amount of CPU resources, even while both zones are requesting the same amount of CPU resources from the system.

```
global # prstat -Z
PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 28848 root     1144K  680K cpu10   12    0   0:00:34 8.2% nspin/1
 28844 root     1144K  680K cpu2    13    0   0:00:33 8.0% nspin/1
 28845 root     1144K  680K run      9    0   0:00:33 8.0% nspin/1
 28846 root     1144K  680K cpu3     8    0   0:00:33 8.0% nspin/1
 28843 root     1144K  816K run     11    0   0:00:33 7.8% nspin/1
 28849 root     1144K  680K cpu0    13    0   0:00:32 7.7% nspin/1
 28847 root     1144K  680K run     12    0   0:00:32 7.6% nspin/1
 28850 root     1136K  672K cpu1    14    0   0:00:32 7.5% nspin/1
 28772 root     1144K  680K run      8    0   0:00:18 4.1% nspin/1
 28771 root     1144K  680K run      3    0   0:00:19 4.1% nspin/1
 28775 root     1136K  672K run     10    0   0:00:19 4.1% nspin/1
 28774 root     1144K  680K run      9    0   0:00:19 4.1% nspin/1
 28769 root     1144K  680K run      1    0   0:00:19 4.0% nspin/1
 28768 root     1144K  816K run     12    0   0:00:17 4.0% nspin/1
 28770 root     1144K  680K run     13    0   0:00:17 3.9% nspin/1
ZONEID    NPROC  SIZE   RSS MEMORY      TIME  CPU ZONE
     9       17   43M   30M   0.4%   0:04:30  63% sales
    10       35  105M   69M   0.8%   0:02:37  32% mkt
     0       50  219M  127M   1.5%   0:01:24 0.1% global

Total: 102 processes, 331 lwps, load averages: 10.89, 5.64, 3.09
```

To observe processes in one or more specific zones, the `prstat` command can be given a list of zones to observe with the `-z` option. The following output was taken while both zones were executing eight instances of the `nspin` command. Only eight of the sixteen `nspin` processes are shown here (those in the sales zone).

```
global # prstat -z sales -a
PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 28845 root     1144K  680K run      7    0   0:01:39 8.5% nspin/1
 28850 root     1136K  672K run     12    0   0:01:38 8.3% nspin/1
 28846 root     1144K  680K run      7    0   0:01:38 8.3% nspin/1
 28849 root     1144K  680K run     14    0   0:01:38 8.2% nspin/1
 28844 root     1144K  680K cpu0    18    0   0:01:39 8.2% nspin/1
 28843 root     1144K  816K run     11    0   0:01:38 8.1% nspin/1
 28847 root     1144K  680K cpu3    18    0   0:01:37 8.0% nspin/1
 28848 root     1144K  680K cpu10   23    0   0:01:39 7.8% nspin/1
 28401 root       11M 8584K sleep   59    0   0:00:02 0.0% svc.startd/11
 28399 root     2200K 1456K sleep   59    0   0:00:00 0.0% init/1
 28496 root     1280K 1032K sleep   59    0   0:00:00 0.0% sh/1
 28507 root     3544K 2608K sleep   59    0   0:00:00 0.0% nscd/23
 28516 root     1248K  920K sleep   59    0   0:00:00 0.0% utmpd/1
 28388 root        0K    0K sleep   60    -   0:00:00 0.0% zsched/1
 28517 root     2072K 1344K sleep   59    0   0:00:00 0.0% ttymon/1
 NPROC USERNAME  SIZE   RSS MEMORY      TIME  CPU
    16 root       39M   29M   0.3%   0:13:14  65%
     1 daemon   3528K 1312K   0.0%   0:00:00 0.0%


Total: 17 processes, 60 lwps, load averages: 15.47, 9.33, 4.85
```

Controlling CPU Consumption Inside Zones

The `zone.cpu-shares` resource control determines the CPU consumption of the zone as a whole in relation to other active zones. CPU consumption *inside* a zone is controlled by the `project.cpu-shares` resource control. Since zones have their own project database, the CPU consumption inside the zone can be controlled by the local zone administrator. To demonstrate this capability, two projects are added to the project database in the sales zone. The CPU shares of the projects are set to 40 and 10, giving the first project four times more CPU resources than the second project. Each project runs four instances of the `nspin` utility.

```
sales # projadd -K "project.cpu-shares=(privileged,40,none)" -U root abc
sales # projadd -K "project.cpu-shares=(privileged,10,none)" -U root xyz
sales # cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
ora_sales:100:Oracle:oracle::project.max-shm-memory=(privileged,2147483648,deny)
abc:101::root::project.cpu-shares=(privileged,40,none)
xyz:102::root::project.cpu-shares=(privileged,10,none)

sales # newtask -p abc
sales # id -p
uid=0(root) gid=1(other) projid=(abc)
sales # nspin -n 4 &
29004
sales # newtask -p xyz
sales # id -p
uid = 0(root) gid=1(other) projid=(xyz)
sales # nspin -n 4 &
29008

sales # prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 29009 root     1144K  680K cpu11   17    0   0:02:19  13% nspin/1
 29008 root     1144K  680K run     22    0   0:02:16  13% nspin/1
...
 28507 root     3680K 2888K sleep   59    0   0:00:00 0.0% nscd/24
 28997 root     1280K 1032K sleep   59    0   0:00:00 0.0% sh/1
PROJID    NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
   101        5 5808K 3832K   0.0%   0:09:09  52% abc
   102        5 5808K 3832K   0.0%   0:02:40  14% xyz
     1        5   13M   10M   0.1%   0:00:00 0.0% user.root
     0        8   33M   24M   0.3%   0:00:08 0.0% system
Total: 23 processes, 67 lwps, load averages: 15.89, 13.20, 11.70

global # prstat -Z
PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 29009 root     1144K  680K cpu11   28    0   0:03:35  13% nspin/1
...
 29004 root     1144K  680K run     24    0   0:01:01 3.5% nspin/1
 29006 root     1136K  672K run     27    0   0:01:01 3.4% nspin/1
ZONEID    NPROC  SIZE   RSS MEMORY      TIME  CPU ZONE
     9       21   49M   36M   0.4%   0:18:17  65% sales
    10       35  105M   70M   0.8%   1:35:49  34% mkt
     0       54  244M  138M   1.7%   0:01:25 0.0% global

Total: 110 processes, 340 lwps, load averages: 15.98, 13.96, 12.13
```

In this case, with only the `sales` and the `mkt` zones active, the `sales` zone is entitled to the following percentage of available CPU resources, as calculated by:

$$\frac{\text{zones.cpu-shares}_{sales}}{(\text{zone.cpu-shares}_{sales} + \text{zone.cpu-shares}_{mkt})} \times 100 = \frac{20}{(20 + 10)} \times 100 = 66\%$$

This 66 percent is then distributed among the projects in the zone. The project `abc` is entitled to the following percentage of available CPU resources:

$$\frac{\text{project.cpu-shares}_{abc}}{(\text{project.cpu-shares}_{abc} + \text{project.cpu-shares}_{xyz})} \times 66\% = \frac{40}{(40 + 10)} \times 66\% = 53\%$$

The `xyz` project is entitled to 13 percent of total CPU resources (as calculated by 10 / (40 + 10) * 66% = 13%). The output from the `prstat -J` command in the sales zone confirms that this is the case. Note that the global zone has been omitted from the calculations for simplicity. It does, however, use some CPU resources so the numbers calculated may differ slightly from observed behavior.

Halting Zones

There are two methods for stopping a zone. The first, and graceful, method instructs a zone to terminate running processes and run any legacy *.rc* scripts.

```
global# zlogin zonename /usr/bin/shutdown -g 10 -y -i 0
```

While faster, the second method does not warn zone users that the zone is shutting down, and does not allow the zone to gracefully terminate services. It forcefully stops all processes running in the zone and changes the zone's state to `installed`.

```
global# zoneadm -z zonename halt
```

7

Creating Solaris Containers

Combining the resource management and Solaris Zones features available in the Solaris 10 OS, system administrators can create Solaris Containers tailored for a specific use. Building on the examples in the previous chapters, assume an administrator wants to run the following workloads on a single SMP system:

- The production sales database
- The production marketing database
- A development environment for the marketing database with multiple developers working on application development
- A development environment for the sales database
- System management

The following issues prevent successful consolidation onto a single system:

- The databases are managed by two different DBA organizations that each have their own (conflicting) standards
- The database systems use different naming services
- The development systems use the same usernames, file system paths and Oracle SIDs as the production environment
- The database instances should be guarateed a certain minimum and maximum amount of CPU capacity at all times
- The production systems should get preferential treatment over the development systems
- The sales database is the most important workload
- Developers should not be able to monopolize the CPU resources on the development systems
- The marketing department is willing to pay for a maximum of two CPUs

The problem is that the sales and marketing databases cannot co-exist on a single system because of different database administration standards and the use of different naming services. This can be overcome by using a separate zone for each workload. The issue of the development environments sharing naming with production can also be overcome with zones. Each zone has its own namespace for users, file systems, network ports and naming services.

The guarantee for minimum and maximum CPU capacity can be ensured by using Dynamic Resource Pools and the Fair Share Scheduler. Resource Pools provide hard upper and lower bounds at the granularity of a CPU. By creating a pool for the sales production database, a pool for the marketing database, and a pool for all other workloads, the production databases can be given guaranteed CPU capacity.

The demand for preferential treatment of the production systems can be implemented using the Fair Share Scheduler by assigning the production zones more `zone.cpu-shares` than development zones. When contention for CPU resources arises, the production zones are given more CPU resources than the other zones.

To prevent a developer from monopolizing the CPU resources in a development zone, each developer is assigned to a project with a certain amount of `project.cpu-shares`. The Fair Share Scheduler is used inside the development zones to distribute CPU resources according to the shares.

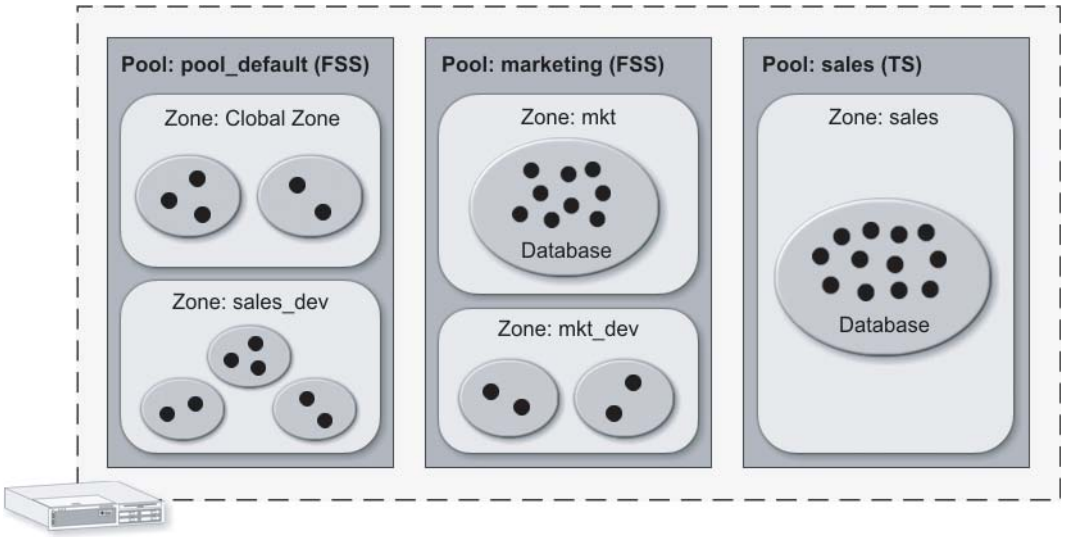

FIGURE 9-1 Solaris Zones and the resource management features of the Solaris 10 OS work together to enable applications to co-exist on systems

This leads to the following design:

- The resouce pool `sales` with processor set `large` with at least two CPUs and no upper bound on CPUs.
- Bound to this pool is a single zone `sales`, allowing exclusive access to CPUs in the `large` processor set for the sales production database only.
- Inside the zone, a single project `ora_sales`, used to limit the amount of shared memory.
- The pool uses the default Time Sharing (TS) scheduler since there is no need to arbitrate between zones in the pool or between projects in the zone.
- A `marketing` resource pool with processor set `small` with one to two CPUs.

- Bound to this pool are two zones, `mkt` for the marketing production database and `mkt_dev` for the development database.
- The pool uses the Fair Share Scheduler (FSS) to arbitrate between the two zones bound to the pool and between projects in those zones.
- Inside each zone is one project `ora_mkt` to limit shared memory for the Oracle instance.
- Each developer is assigned a unique project in the development zone with a suitable amount of CPU shares.
- The default resource pool `pool_default` with a processor set with at least one CPU.
- The `global` zone and `sales_dev` zone for sales developers are bound to the pool.
- This pool uses the FSS scheduler to arbitrate between the two zones bound to the pool, and between projects in those zones.
- Developers receive unique projects in the development zone with sufficient CPU shares.

Container Construction

Creating the Pools

The pool configuration built ealier almost matches the design, and could be used as a basis to create the required pools. However, the pools are created from scratch in order to show all relevant steps in a single location.

1. Enable the resource pools facility and save the default configuration to the */etc/pooladm.conf* file. The default configuration constists of a processor set `pset_default` with all CPUs and a single pool `pool_default`.

```
global # pooladm -e
global # pooladm -s
```

2. Create the `sales` resource pool with TS as the scheduler and the `large` processor set with at least two CPUs.

```
global # poolcfg -c 'create pset large (uint pset.min=2; uint pset.max=65536)'
global # poolcfg -c 'create pool sales (string pool.scheduler="TS")'
global # poolcfg -c 'associate pool sales (pset large)'
```

3. Create the `marketing` resource pool with FSS as the scheduler and the `small` processor set with one or two CPUs.

```
global # poolcfg -c 'create pset small (uint pset.min=1; uint pset.max=2)'
global # poolcfg -c 'create pool marketing (string pool.scheduler="FSS")'
global # poolcfg -c 'associate pool marketing (pset small)'
```

4. Set the scheduler for the default pool to the Fair Share Scheduler and instantiate the pool configuration just created:

```
global # poolcfg -c 'modify pool pool_default (string pool.scheduler="FSS")'
global # pooladm -c
global # poolcfg -dc info

system blondie
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 29072

        pool marketing
                int     pool.sys_id 5
                boolean pool.active true
                boolean pool.default false
                string  pool.scheduler FSS
                int     pool.importance 1
                string  pool.comment
                pset    small
        pool sales
                int     pool.sys_id 6
                boolean pool.active true
                boolean pool.default false
                string  pool.scheduler TS
                int     pool.importance 1
                string  pool.comment
                pset    large
        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                string  pool.scheduler FSS
                int     pool.importance 1
                string  pool.comment
                pset    pset_default
         pset large
                int     pset.sys_id 1
                boolean pset.default false
                uint    pset.min 2
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 0
                uint    pset.size 2
                string  pset.comment
```

```
cpu
                int     cpu.sys_id 3
                string  cpu.comment
                string  cpu.status on-line
cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line
pset small
         int     pset.sys_id 2
         boolean pset.default false
         uint    pset.min 1
         uint    pset.max 2
         string  pset.units population
         uint    pset.load 0
         uint    pset.size 2
         string  pset.comment
          cpu
                 int     cpu.sys_id 1
                 string  cpu.comment
                 string  cpu.status on-line
         cpu
                 int     cpu.sys_id 0
                 string  cpu.comment
                 string  cpu.status on-line

 pset pset_default
         int     pset.sys_id -1
         boolean pset.default true
         uint    pset.min 1
         uint    pset.max 65536
         string  pset.units population
         uint    pset.load 17
         uint    pset.size 2
         string  pset.comment

         cpu
                 int     cpu.sys_id 11
                 string  cpu.comment
                 string  cpu.status on-line

         cpu
                 int     cpu.sys_id 10
                 string  cpu.comment
                 string  cpu.status on-line
```

Binding Zones to Pools

Currently all zones are bound to the default pool because the `pool` property of the created zones has not been set, resulting in the zones being bound to the pool with the `pool.default` attribute set to true. Setting the zone's pool property to the name of a resource pool binds that zone and all of its processes to that pool when the zone is booted. Note that since the sales zone is bound to a resource pool with the normal TS scheduler, the `zone.cpu-shares` resource control is no longer applicable and is therefore removed from the zone configuration.

```
global # zonecfg -z sales set pool=sales
global # zonecfg -z sales remove rctl name=zone.cpu-shares
global # zonecfg -z mkt set pool=marketing
```

To bind a running zone to a pool without rebooting the zone, the `poolbind(1M)` command can be used. This dynamically rebinds the zone and its processes to a pool until the next zone boot. To have this change persist accross zone reboots, the zone's property should be set as shown above.

```
global # poolbind -q `pgrep -z sales -x init`
28399   pool_default
global # poolbind -p sales -i zoneid sales
global # poolbind -q `pgrep -z sales -x init`
28399   sales
global # poolbind -p marketing -i zoneid mkt
global # poolbind -q `pgrep -z mkt -x init`
28545   marketing
```

The `poolbind -q `pgrep -z sales -x init`` command is used to ascertain to which zone the current pool is bound by querying the binding of the `init(1M)` process of that zone. As can been seen, the sales zone was bound to the pool `pool_default` and is now bound to the `sales` pool.

Creating Development Zones

The development environments for both databases get their own zones, enabing them to use the same user names, projects and file system paths as the production environments. The development zone for the sales database, `sales_dev`, is bound to the default pool and shares the pool with all processes of the global zone. To prevent the `sales_dev` zone from monopolizing CPU resources, its `zone.cpu-shares` is set to the same value as that of the

global zone. This gives both zones equal access to CPU resources. When the Fair Share Scheduler is active in a resource pool, it only looks at processes in that pool. The amount of shares for the `sales_dev` zone is only relevant in relation to those of the global zone.

```
global # zonecfg -z sales_dev
sales_dev: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:sales_dev> create
zonecfg:sales_dev> set zonepath=/export/zones/sales_dev
zonecfg:sales_dev> set autoboot=true
zonecfg:sales_dev> set pool=pool_default
zonecfg:sales_dev> add rctl
zonecfg:sales_dev:rctl> set name=zone.cpu-shares
zonecfg:sales_dev:rctl> add value (priv=privileged,limit=1,action=none)
zonecfg:sales_dev:rctl> end
[...]
global # chmod 700 /export/zones/sales_dev
global # zoneadm -z sales_dev install
[...]
global # zoneadm -z sales_dev boot
```

The development environment of the marketing database uses the same pool as the zone for the marketing production database. The Fair Share Scheduler is used to give preferential access to the production zone. By setting the `zone.cpu-shares` of the `mkt` zone to 50, and the `zone.cpu-shares` of the `mkt_dev` zone to 10, the production database is granted five times as much CPU resources as the development database.

```
global # zonecfg -z mkt 'select rctl name=zone.cpu-shares; set
          value=(priv=privileged,limit=50,action=none);end'

global # zonecfg -z mkt_dev
mkt_dev: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:mkt_dev> create
zonecfg:mkt_dev> set zonepath=/export/zones/mkt_dev
zonecfg:mkt_dev> set autoboot=true
zonecfg:mkt_dev> set pool=marketing
zonecfg:mkt_dev> add rctl
zonecfg:mkt_dev:rctl> set name=zone.cpu-shares
zonecfg:mkt_dev:rctl> add value (priv=privileged,limit=10,action=none)
zonecfg:mkt_dev:rctl> end
[...]
global # chmod 700 /export/zones/mkt_dev
global # zoneadm -z mkt_dev install
[...]
global # zoneadm -z mkt_dev boot
```

The pool bindings for the zones can be verified using the `poolbind -q pid` command on every zone's `init(1M)` process.

```
global # poolbind -q `pgrep -z sales_dev -x init`
6718    pool_default
global # poolbind -q `pgrep -z sales -x init`
28399   sales
global # poolbind -q `pgrep -z mkt -x init`
28545   marketing
global # poolbind -q `pgrep -z mkt_dev -x init`
6579    marketing
global # poolbind -q `pgrep -z global -x init`
1       pool_default
```

Creating Development Users and Projects

Once all zones are created, it is time to create users and projects inside the development zones and set the appropriate resource controls to implement the design. The Fair Share Scheduler is used to prevent the developers from consuming the CPU resources. In both zones three users and three projects are created.

User	Project	Resource Controls	Value
oracle	ora_mkt	project.max-shm-memory	2 GB
		project.cpu-shares	100
oracle	ora_sales	project.max-shm-memory	2 GB
		project.cpu-shares	100
dev1	user.dev1	project.cpu-shares	10
dev2	user.dev2	project.cpu-shares	10

```
global # zlogin mkt_dev
mkt_dev # mkdir -p /export/home
mkt_dev # groupadd dba
mkt_dev # useradd -g dba -m -d /export/home/oracle -s /bin/bash oracle
mkt_dev # projadd -U oracle ora_mkt
mkt_dev # projmod -sK "project.max-shm-memory=(privileged,2G,deny)" ora_mkt
mkt_dev # projmod -sK "project.cpu-shares=(privileged,100,none)" ora_mkt
mkt_dev # useradd -m -d /export/home/dev1 -s /bin/bash dev1
mkt_dev # useradd -m -d /export/home/dev2 -s /bin/bash dev2
mkt_dev # projadd user.dev1
mkt_dev # projadd user.dev2
mkt_dev # projmod -sK "project.cpu-shares=(privileged,10,none)" user.dev1
mkt_dev # projmod -sK "project.cpu-shares=(privileged,10,none)" user.dev2
[Oracle installation omitted for brevity...]
```

```
global # zlogin sales_dev
sales_dev # mkdir -p /export/home
sales_dev # groupadd dba
sales_dev # useradd -g dba -m -d /export/home/oracle -s /bin/bash oracle
sales_dev # projadd -U oracle ora_sales
sales_dev # projmod -sK "project.max-shm-memory=(privileged,2G,deny)" ora_sales
sales_dev # projmod -sK "project.cpu-shares=(privileged,100,none)" ora_sales
sales_dev # useradd -m -d /export/home/dev1 -s /bin/bash dev1
sales_dev # useradd -m -d /export/home/dev2 -s /bin/bash dev2
sales_dev # projadd user.dev1
sales_dev # projadd user.dev2
sales_dev # projmod -sK "project.cpu-shares=(privileged,10,none)" user.dev1
sales_dev # projmod -sK "project.cpu-shares=(privileged,10,none)" user.dev2
[Oracle installation omitted for brevity...]
```

Verifying the Configuration

The configuration just built can be verified using the following steps:

1. Start the `prstat -Z` command in the global zone to observe the CPU utilization of the zones.
2. Start the `poolstat -r pset 5` command in the global zone to observe utlization in the resource pools.

3. Create load using the `nspin -n 4` command in the `mkt` zone as the user `oracle` in the `ora_mkt` project. Note the CPU consumption of the `mkt` zone peaks around 33% since the marketing resource pool to which the zone is bound consists of two CPUs. The other CPUs are idle.

4. Add the same load in the `mkt_dev` zone. The combined CPU usage of the `mkt` and `mkt_dev` zones is approximately 33% since they share the same resource pool. The `mkt` zone receives approximately 27% and the `mkt_dev` zone about 6% because the `mkt` zone has five times more `zone.cpu-shares` than the `mkt_dev` zone.

5. Add the same load in the `sales` zone. The `sales` zone receives 33% since it is bound to the `sales` pool, which also has two CPUs. The CPU consumption of the `mkt` and `mkt_dev` zones is not impacted by the CPU usage of the `sales` zone.

6. Add load in the `sales_dev` zone. This zone is bound to the default pool. As a result, it is able to use all of the remaining CPU capacity since it is the only zone in that pool using CPU resources.

7. Add the same load in the global zone. The global zone is also bound to the default pool, and has the same amount of `zone.cpu-shares` as the `sales_dev` zone. The CPU usage of both zones is therefore equal, and approximately 16 percent. The resulting `prstat -Z` command output looks as follows:

```
global # prstat -Z
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  1987 100      1144K  672K cpu3    20    0   0:04:17 8.4% nspin/1
[...]
  2031 root     1144K  808K run      7    0   0:00:49 4.1% nspin/1
ZONEID    NPROC  SIZE   RSS MEMORY      TIME  CPU ZONE
     2       15   42M   30M   0.4%   0:17:17  33% sales
     1       33  104M   69M   0.8%   0:23:00  27% mkt
     0       65  388M  179M   2.2%   0:03:36  17% global
     3       33  105M   71M   0.8%   0:06:27  16% sales_dev
     4       33  103M   69M   0.8%   0:03:53 5.7% mkt_dev

Total: 179 processes, 586 lwps, load averages: 19.81, 14.84, 8.04
```

8. Add load in the `sales_dev` zone in the `user.dev1` and `user.dev2` projects. The total CPU usage of the `sales_dev` zone remains the same. However, in the zone the CPU should now be divided across the three projects according to the `project.cpu-shares` in the zone. Notice that a zone bound to a resource pool is only aware of the CPUs in the associated processor set. As a result, the `sales_dev` zone only knows about two CPUs, and the usage shown in the output of the `prstat` command is therefore based on two CPUs. That is why *inside* the zone the three projects seem to use 50 percent. (The other 50 percent is used by the global zone that is also bound to the same pool.) The `user.dev1` and `user.dev2` projects receive 10/120ths each of that 50 percent since they each have 10 `project.cpu-shares` and `ora_sales` has 100 `project.cpu-shares`.

```
sales_dev # prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  2016 oracle   1144K  672K run     35    0   0:24:48  10% nspin/1
  2088 dev1     1136K  704K run      1    0   0:00:03 1.4% nspin/1
[...]
  2113 dev2     1144K  848K run     27    0   0:00:02 1.2% nspin/1
PROJID    NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
   100        5 7456K 4864K   0.1%   1:39:06  40% ora_sales
   101        5 7464K 5072K   0.1%   0:00:12 5.3% user.dev1
   102        5 7464K 5072K   0.1%   0:00:08 5.0% user.dev2
     1        5   11M 9336K   0.1%   0:00:00 0.2% user.root
     0       26   93M   63M   0.8%   0:00:14 0.0% system
     3        1 2904K 2064K   0.0%   0:00:00 0.0% default
Total: 47 processes, 132 lwps, load averages: 14.43, 10.28, 8.84
```

This example illustrates some of the ways that Solaris Containers technologies can be used to facilitate consolidation. It should be noted that not all features must be used at the same time. Depending on the circumstances some Solaris Container technologies, such as resource management, resource pools and Solais Zones, can be mixed and matched to meet the specific needs for a consolidation project. In some cases, just using the Fair Share Scheduler may be sufficient to meet requirements, while in other cases Solaris Zones can be the key technology to a succesful consolidation.

8

Integrating Solaris Containers into the Environment

Solaris Containers technology provides extensive flexibility, and selecting an appropriate set of these features for a given environment can be challenging. This chapter provides guidelines and suggestions for designing system configurations using these powerful tools. More detailed documents are available that provide the commands needed to implement these guidelines, and can be found in the references listed at the end of this document.

When designing system configurations, Solaris Containers technology can help with:

- *Storage configuration* — Solaris Containers support the use of multiple storage types, including direct attached storage (DAS), network attached storage (NAS) and storage area networks (SAN), as well as multiple file system types, and flexible layouts.
- *Flexible network configurations* — Solaris Containers can be used in conjunction with Internet Protocol Multi-Pathing (IPMP), trunking, quality of service (QoS), virtual LANS (VLANs), and network address translation (NAT).
- *Resource management controls* — Solaris Containers utilize processor sets, pools, projects, and other resource management facilities in the Solaris OS to gain control over computing resources and affect better utilization.

Solaris Containers provide a wide variety of configuration options. While default parameters ensure the creation of highly secure zones, global zone administrators can change these defaults and add more functionality to the environment. Many of these choices are discussed below, along with potential pitfalls. Additional topics can be found in the *Zones and Containers FAQ* located at *http://www.opensolaris.org/os/community/zones/faq/* on the OpenSolaris Web site.

Storage Configuration

Several factors related to storage and data systems should be considered when deploying Solaris Containers technology.

File System Structure

Two models can be used for the operating system file layout for a Solaris non-global zone.

- *Whole root*
  A whole root zone includes a complete copy of the operating system on disk. A process running in a whole root zone can only see the files in this copy, as well as files created by users in this zone. This model offers flexibility—the root user can install new versions of the system and user libraries. However, the whole root model also requires more administration, such as keeping track of different library versions and patching each one appropriately.
- *Sparse root*
  A sparse root zone does not include its own private copy of the operating system binaries and libraries. Indeed, it does not have copies of the */usr*, */sbin*, */platform*, or */lib* directories. Instead, programs in the non-global zone use the files that reside in the global zone via a loopback file system (LOFS) mount point to each aforementioned directory. It is important to note that the default model for a zone is the sparse root model, as it saves physical memory and disk space, and simplifies administration. When a sparse root zone is installed, it is aware of the same packages the global zone knows about at the time of zone installation, including non-Solaris packages. The only exceptions are packages installed in the global zone with the package parameter `SUNW_PKG_THISZONE` set to `TRUE`.

Whole root zones use more physical memory and storage resources than sparse root zones. In shared library environments like the Solaris OS, all users share in-memory copies of system libraries to reduce memory use. Because sparse root zones share the */lib* directory, all users in all sparse root zones share the same in-memory copy of these libraries, along with the global zone. Users in whole root zones access separate copies of these libraries, increasing system memory use.

Recall that the global zone is the place where the operating system kernel runs, and from where the system is controlled and configured, and where the other *non-global zones* are created. In addition to the fact that the global zone and non-global zones have differences in directory structure and layout, non-global zones do not contain the following directories:

- */boot,* files used for hardware booting that are not needed in non-global zones
- */devices*, device files that are not allowed in non-global zones
- */vol*, device entries for removable hardware

File Systems versus Raw Devices

Access to a raw device, such as */dev/rdsk/c1t0d0s0*, can be assigned to a Container through the use of the `add device` subcommand of the `zonecfg(1m)` command. Raw device access should be used with caution as it may enable Container users with sufficient privileges to bypass the security boundary built into each Container. For example, providing direct access to a raw disk partition enables the root user of a non-global zone to create a file system and write garbage to its superblock, which will cause the system to panic. Whenever possible, avoid giving Containers access to devices.

Selecting Direct Attached Storage, Network Attached Storage, and Storage Area Networks

A Container can access direct attached storage, storage area networks, and network attached storage file systems when properly configured. Consider the following when choosing an access method:

- *Direct attached storage*
  Direct attached storage (DAS) is defined by storage devices that reside in, or are directly connected to, a computer system. While direct attached storage provides the simplest access method, it limits flexibility when moving a Container or its workload to a different system.
- *Network attached storage*
  Network attached storage (NAS) is characterized by the ability to provide data access over the network. Currently, the root directory of a Container cannot be stored on network attached storage (NAS). However, NAS systems can be used to centralize zone and application storage, helping to simplify storage management and disaster recovery methods.
- *Storage area networks*
  Storage area networks (SANs) are networks of storage devices that provide data access to servers. Storage area networks can be used to centralize zone and application storage, helping to simplify storage management and disaster recovery methods.

File System Types

Storage can be assigned to zones via several methods. This section briefly describes some of these methods, and compares their use. Each method can be achieved manually from the global zone, or automatically through proper configuration with the `zonecfg(1M)` command.

Loopback File System (LOFS)

Use of the loopback file system takes an arbitrary directory in the global zone's file tree and makes it available within a zone. This can be specified in the zone's configuration.

```
global# newfs  /dev/rdsk/c1t0d0s6
global# mount /dev/dsk/c1t0d0s6 /export/opt/local
global# zonecfg  -z  zone1
add fs
set dir=/opt/local
set special=/export/opt/local
set type=lofs
end
exit
global# zoneadm -z zone1 boot
```

This can also be accomplished manually from the global zone. This can prove useful if the zone is already running.

```
global# mount -F lofs /export/opt/local zonepath/root/opt/local
```

File systems mounted in this manner can be mounted simultaneously in different zones, providing a shared file system that is accessible to both zones at the same time.

LOFS mounts are particularly useful for making CD-ROMs available to a zone in a safe manner. If it is desired to have the mount occur automatically each time the zone boots, use the `zonecfg` command to make the file system mount point available to the zone.

```
global# mkdir zonepath/root/cdrom
global# mount -F lofs /cdrom zonepath/root/cdrom
global# zonecfg
add fs
set dir=/cdrom
set special=/cdrom
set type=lofs
end
exit
```

Solaris ZFS File System

Introduced in Solaris 10 6/06, Solaris ZFS (ZFS) is a new file system that includes three different ZFS objects which can be used within a Solaris Container: a file system, dataset, and ZFS device. A ZFS file system is analogous to a UFS file system—files can be stored in a hierarchical arrangement. A ZFS dataset is a subset of a ZFS pool, and can be a file system, snapshot, or volume.[3] A ZFS device is handled just like any other device, and is explained below in the direct device section.

A ZFS file system can be made available to a Container using the following `zonecfg` commands.

```
global# zfs create zonepool/zone1-usr-local
global# zonecfg -z zone1
add fs
set dir=/usr/local
set special=/zonepool/zone1-usr-local
set type=zfs
end
exit
global# zoneadm -z zone1 boot
```

A ZFS file system can also be made available to a running Container. For example, an existing ZFS file system called `zonepool/zone1` can be made accessible from with a command using the following command.

```
global# zfs set mountpoint=zonepath/root/usr/local zonepool/zone1
```

In either case, the global zone administrator manages the properties of the file system, including its maximum size, or quota.

A ZFS dataset can be delegated to a non-global zone, and is managed by the zone administrator. Doing so allows the zone administrator to create and otherwise manage file systems in that dataset. There is one exception: the quota for the dataset can only be modified by the global zone administrator. Using a dataset is the recommended method for allowing a zone to manage its own file systems.

Adding a dataset to a zone is accomplished from within the `zonecfg` command.

3. A ZFS volume cannot be made available to a Container.

```
global# zonecfg -z zone1
add dataset
set name=zonepool/zone1
end
exit
global# zoneadm -z zone1 boot
```

At this point, the zone administrator can create file systems for the zone. For example:

```
zone1# zfs create zonepool/zone1/fs-one
zone1# zfs list
NAME                    USED   AVAIL   REFER  MOUNTPOINT
zonepool               75.1M   99.9M   11.5K  /zonepool
zonepool/zone1         18.5K   99.9M   9.50K  /zonepool/zone1
zonepool/zone1/fs-one    9K   99.9M     9K  /zonepool/zone1/fs-one
```

The global zone administrator can constrain the amount of space the dataset can use, as follows.

```
global# zfs set quota=10m zonepool/zone1
zone1# zfs list
NAME                    USED   AVAIL   REFER  MOUNTPOINT
zonepool               75.1M   99.9M   11.5K  /zonepool
zonepool/zone1         18.5K   9.98M   9.50K  /zonepool/zone1
zonepool/zone1/fs-one    9K   9.98M     9K  /zonepool/zone1/fs-one
```

However, the non-global zone administrator can manage the settings of any file systems that are children of the dataset, including quotas, provided the sum of those quotas does not exceed the quota for the dataset.

```
zone1# zfs set quota=1m zonepool/zone1/fs-one
zone1# zfs list
NAME                    USED   AVAIL   REFER  MOUNTPOINT
zonepool               75.1M   99.9M   11.5K  /zonepool
zonepool/zone1         18.5K   9.98M   9.50K  /zonepool/zone1
zonepool/zone1/fs-one    9K   1015K     9K  /zonepool/zone1/fs-one
```

UNIX File System

When using the UNIX file system (UFS), a block device is mounted. The file system on the block device is mounted on a directory in the Container. Note that while the local zone does not have access to devices, the configuration must include both the block and raw devices. A sample configuration follows.

```
global# newfs /dev/dsk/c1t0d0s6
global# zonecfg -z zone1
     add fs
     set dir=/opt/local
     set special=/dev/dsk/c1t0d0s6
     set raw=/dev/rdsk/c1t0d0s6
     set type=ufs
     add options [ro,nodevices]
     end
     exit
global# zoneadm -z zone1 boot
```

Directories and files available to the non-global zone can be managed from the global zone with this method. If the zone is already running, the global administrator can accomplish this manually with the following command. Note the commands shown above must be used in order for the mount to occur the next time the zone boots.

```
global# mount  /dev/dsk/c1t0d0s6   zonepath/root/opt/local
```

Direct Device

The direct device method gives zone administrators direct control over a file system's devices, and enables direct management of the file system. However, zone administrators gain greater control over the system components which can affect other zones. For example, just as the root user in a non-zoned system can use device access to panic a UNIX system, assigning direct device access to a zone may give the zone administrator the ability to panic the system, including all zones.

```
global# zonecfg -z zone1
        add device
             set match=/dev/rdsk/c1t0d0s6
             end
        add device
             set match=/dev/dsk/c1t0d0s6
             end
global# zoneadm -z zone1 boot
zone1# newfs /dev/rdsk/c1t0d0s6
zone1# mount /dev/dsk/c1t0d0s6 /opt/local
```

Alternatively, the file system can be mounted from the global zone with the following command. By specifying the mount command in this way, the mount point can still be managed from within the zone.

```
global# mount zonepath/root/dev/dsk/c1t0d0s6 zonepath/root/opt/local
```

Network File System

Non-global zones can mount Network File System (NFS) shares into their directory structure, just like non-zoned systems. For manual mounts:

```
zone1# mount -F nfs nfs-server:/export/zone1 /opt/myfiles
```

The Solaris OS automatically mounts the NFS file system at boot time, if the following line is added to the non-global zone's */etc/vfstab* file:

```
nfs-server:/export/zone1  -  /opt/myfiles  nfs  -  yes  -
```

Each Container can mount the same remote NFS share into its own directory structure. However, a Container cannot mount an NFS share from its own global zone. Instead, a global zone which shares a file system with other computers via NFS can also create an LOFS mount into the Container. Furthermore, global zone users cannot see into a non-global zone's NFS-mounted file system. As of this writing, it is not possible to LOFS-mount a directory in the global zone which is an NFS mount point.

A global zone `root` user can mount and unmount NFS shares into a Container by using the `zlogin` command. For example:

```
global# zlogin twilight mount -F nfs nfs-server:/directory/mnt
```

General File System Considerations

The methods discussed thus far have several attributes that must be considered. Straightforward deployments rarely require consideration of these topics. Indeed, the UFS and LOFS methods are simple and safe to use in most cases. However, the following factors should be considered in more demanding situations.

Ability to Mount Read-Write versus Read-Only

Some situations mandate the use of read-only mounts, such as sparse root zones and any directories specified with the `inherit-pkg-dir` attribute. Other situations which may benefit from read-only mounts in zones include NFS mounts (including static files, such as binaries and libraries), and static repositories, such as operating system patches. Other read-only mounts can be configured from the global zone, or from within the non-global zone with NFS and direct device mounts.

Shareable File Systems

Several situations benefit from the use of shared file systems. However, care must be taken when sharing a file system between the global zone and non-global zones. A global zone process should not trust the data or programs that can be accessed by non-global zones. A non-global zone administrator, or an intruder, could replace a common program with a malicious program. When sharing a system, the following guidelines may prove helpful:

- *Data files modified by a process in one zone should not be modified by processes in another zone*

  For example, Web servers can be defaced by intruders who take advantage of weaknesses in Web server software to gain system access and then modify Web page content. This type of attack relies on a read/write mounted file system containing the Web pages to support legitimate users (Figure 8-1).

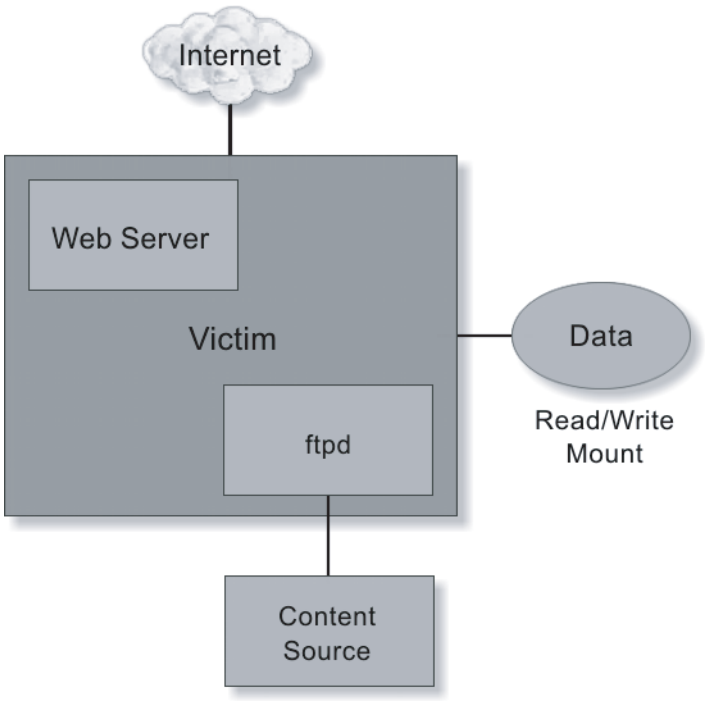

FIGURE 8-1 A typical Web server that is prone to attack

This type of attack can be prevented by ensuring the environment in which the Web server software is running does not have write access to Web page data. This can be achieved by creating a Container for the Web server software, and another Container for users. Users reside on a separate network that is not accessible from the Internet (Figure 8-2).

With this model, Web page content can be created by a system in a demilitarized zone and transferred to a process in Zone 1 of a Web server. This process writes the data into a file system mounted read-only in Zone 2. Web server software in Zone 2 is able to read the data. However, the content is protected from alteration—even if an intruder breaks into the Web server zone and gains root access. Security can be improved by restricting most types of access in Zone 1.

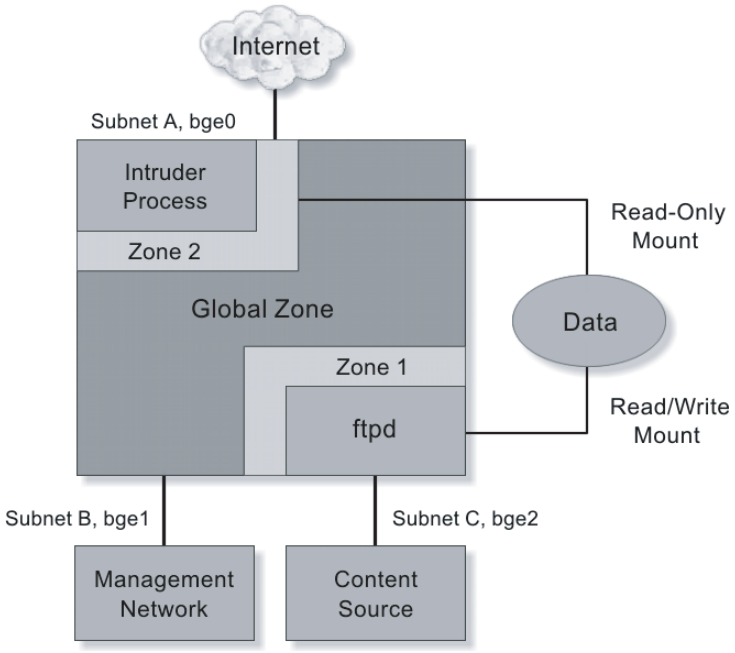

FIGURE 8-2 The use of Solaris Containers technology can help prevent attacks

- *Implementing inter-zone inter-process communication with a shared file system*

  By default, processes in different zones cannot communicate except via an IP network connection. A primitive inter-process communication (IPC) mechanism can be created via a shared file system. By using LOFS, global-zone administrators can provide shared read-write access to a file system from two different zones.

  Using shared file systems creates new possibilities as well as new challenges. With appropriate file access permissions, processes in different zones can use one or more files in the shared file system to synchronize activities (such as locking database tables) or send data to each other. Since the two zones do not normally share a user name space, methods to achieve shared file access are similar to methods used for NFS, such a common user directory service like the lightweight directory access protocol (LDAP) or access control lists (ACLs).

  Use of shared file systems also creates new possibilities for intruders. A person who gains access to one zone may be able to use the shared file system to gain access to, or otherwise affect, the other zone. In addition, synchronization and data transfer can usually be achieved via the network. Since network transfers between zones typically involve simple in-memory copy activities, there is rarely a benefit to using a shared file system. However, shared file systems can be helpful for applications previously written to use a shared file space, and which cannot be modified to use the network. Note that a shareable, read-write file system can only be mounted into two containers concurrently using LOFS.

- *Ability to mount file systems in the global zone when the zone is not running*

  In general, this is useful if the global zone must create or modify files that normally reside in a zone's directory structure. While this is possible using most methods, the file system typically must be unmounted before booting the zone.

- *Manage the file system from within the zone*

  Some situations require file system creation and management from within the non-global zone. This can be accomplished using direct device assignment into the non-global zone. Currently it is not possible to assign a Veritas Volume Manager (VxVM) volume into a non-global zone.

- *Use volume management*

  A file system that uses UFS and the Solaris Volume Manager (SVM) may be mounted into a zone. In fact, the root directory of a zone may be a SVM soft partition, which offers several benefits. First, this technique enables hundreds or thousands of zones to each have a file system. This is typically impossible on non-zoned systems due to the limit of eight slices per non-SAN disk. In addition, a separate file system per zone can help simplify file backup and restoration, and prevent a process in one zone from filling up a shared file system. When this is a concern, each zone should be installed in its own file system.

- *Use file systems such as NFS, UFS, QFS, or VxFS whenever possible, instead of directly assigning devices*

  Use of these file systems improves security and increases service or zone portability. Direct device assignment increases complexity when attempting to replicate a zone on a different system due to hardware configuration differences.

- *From the global zone, unmount a file system is mounted into a non-global zone*

  This is possible using the UFS method described above. If a file in the file system is in use, the unmount command fails with an appropriate error message. In this case, use the `umount -f` command to forcibly unmount the file system.

Backup and Restore

Many methods and technologies replicate data at a certain point in time. This section discusses two such methods—one that uses a traditional tape backup software package, and one that uses Solaris OS features to make a disk copy of the data which can be transferred to long term storage, such as tape or optical media.

There are basic issues common to both methods. Since zones share most of the Solaris OS file space with the global zone by default, there is no need to backup these shared areas at the zone level. The only file systems that should be backed up are those needing restoration when rebuilding a system, including application data files. Because zone configuration information belongs to the global zone, organizations can consider only backing up application data from the zone. Zone specific operating system files, such as those in the zone's */etc* directory, should be backed up directly from the global zone.

Tape Backup

Symantec Veritas NetBackup is a popular software package which enables enterprise-wide coordination of tape backup. NetBackup version 6.0, version 5.0 (with MP 4), and version 5.1 (with MP 2) provide support for Solaris Zones. At the time of this writing, only the NetBackup client software is supported by Symantec within a non-global zone. The Master Server and Media Server software is supported in a global zone or in a non-zoned system. Existing NetBackup architectures do not need modification when applied to zoned systems. Simply treat each non-global zone as if it were a standalone server running the NetBackup client software. Figure 8-3 depicts a simple diagram of a common NetBackup architecture.

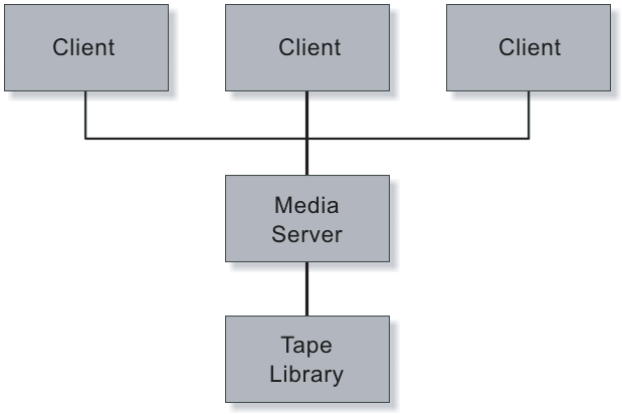

FIGURE 8-3 A typical NetBackup architecture

Figure 8-4 illustrates an architecture with NetBackup client software in non-global zones, each of which sends its data stream to a media server via a local area network (LAN). Note that in one case the LAN connection must have sufficient bandwidth for two backup streams.

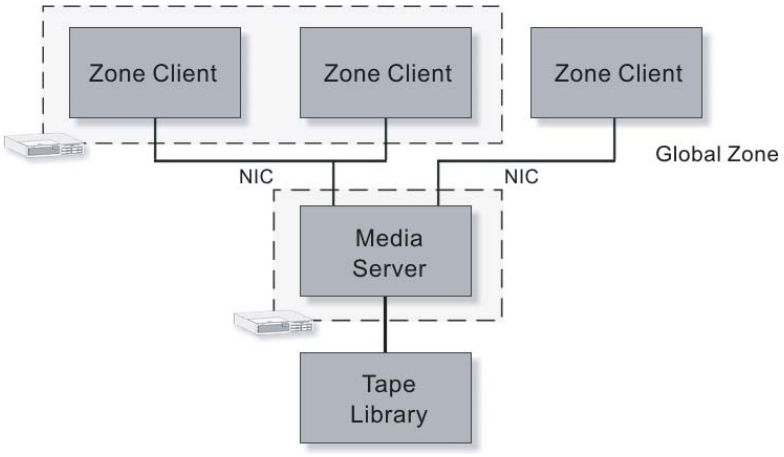

FIGURE 8-4 A backup architecture that uses Solaris Zones technology and a separate media server

To further consolidate servers, or take advantage of the higher network bandwidth and lower latency between zones residing on the same system, co-locate a media server on the same system (Figure 8-5). Note that the connection between the server and the tape library must have sufficient bandwidth for all three backup streams if concurrent backups are desired.

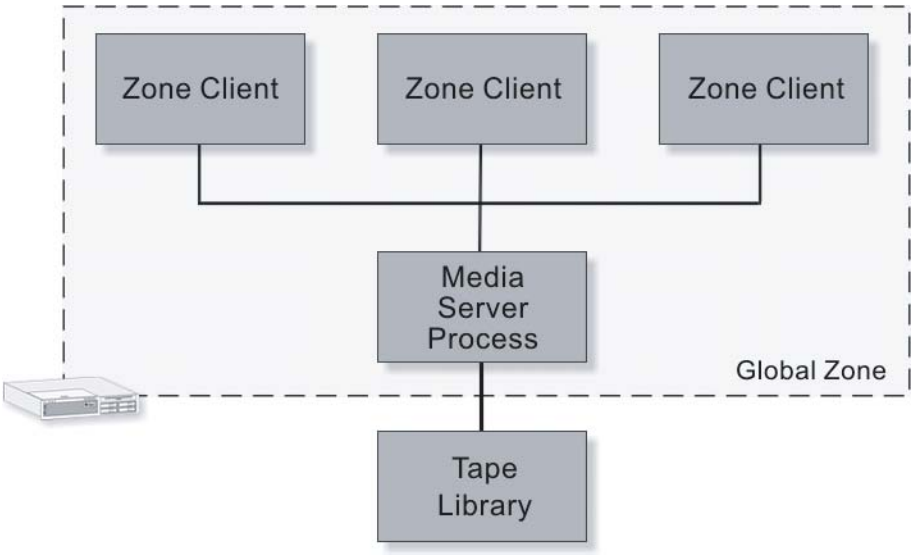

FIGURE 8-5 A backup architecture that places the media server in the global zone

In all cases, knowledge of the data transmission rates along the backup stream can prove helpful. Table 8-1 lists well-known data rates, along with inter-zone network transmission rates measured on some systems.

TABLE 8-1 Sample data rates and inter-zone network transmission rates

Transfer Type	Theoretical Maximum Data Rate		Practical Data Rates
	(MB/Second)	(GB/Hour)	(MB/Second)
Fast Ethernet	12.5	45	~ 9
Gigabit Ethernet	125	450	~ 90
Ultra Wide SCSI	40	144	~ 35
Ultra160 SCSI	160	576	~ 150
Fibre Channel (1 Gbit)	125	450	~ 100
Fibre Channel (2 Gbit)	250	900	~ 200
Inter-Zone (1.7 GHz Pentium M)			50 to 1,200
Inter-Zone (2.0 GHz Athlon)			100 to 1,400
Inter-Zone (2x 300 MHz Sun Enterprise™ E250 Server)			10 to 90

Using this model requires coordination between the global and non-global zones. The media server must backup a zone only after the zone and its applications have quiesced the data to be backed up.

Disk Snapshot

In order to shorten the window of quiesced data, the global zone can use the `fssnap(1M)` command to take a snapshot of the file system in the zone. Using the model in Figure 8-4, the NetBackup client software can reside in the global zone of the application zones, and access the snapshot directly for transfer to the master server. If the model depicted in Figure 8-5 is used, the client software can run in either the global or non-global zone. However, the initial release of the Solaris 10 OS does not permit a non-global zone to use the `fssnap` command to create the snapshot. As a result, there is limited value in placing the client software in the zone, unless the workload can be quiesced for the duration of the backup. Alternatively, synchronize the `fssnap` taking place in the global zone with a short time period when the application is quiesced.

Network Configuration

All network management of a system, including its zones, is performed from the global zone. A zone's network configuration is usually specified before the zone is installed, and may be changed once it is booted. The global zone administrator can even change a zone's network configuration while the zone is running using the `ifconfig` command.

```
global# ifconfig hme0 addif 192.168.1.3/24 zone mercury
global# ifconfig hme0:3 up
```

While a zone can view much of its network configuration information, it cannot modify the configuration. For example, a zone can use the `ifconfig(1M)` command to view its IP address(es):

```
mercury# ifconfig -a
lo0:1:
flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL>
        mtu 8232 index 1 inet 127.0.0.1 netmask ff000000
e1000g0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu
1500
 index 2 inet 192.168.0.71 netmask ffffff00 broadcast
 192.168.0.255
```

To prevent a zone from changing basic configuration parameters, such as network interface status, a zone cannot use the `ifconfig` command to set parameters.

```
mercury# ifconfig e1000g0:1 down
ifconfig: setifflags: SIOCSLIFFLAGS: e1000g0:1: permission denied
```

Note – Limiting the ability of a zone to modify the network configuration is implemented in the Solaris OS kernel through the absence of the `SYS_NET_CONFIG` privilege in a non-global zone.

Dynamic Host Configuration Protocol

The Solaris 10 OS 3/05 does not permit a zone to obtain its IP address from a server using the Dynamic Host Configuration Protocol (DHCP). Furthermore, a zone cannot act as a DHCP server. The global zone is free from both of these restrictions.

Changing the IP Address for a Zone

A global zone administrator can change the IP address of a zone's logical interface. To change a zone's IP address, use the `zonecfg` command to change the IP address in the zone configuration. Next, modify the naming service to reflect this change.These steps should be done while the zone is down to eliminate the possibility of applications and network devices becoming confused.

```
global# zonecfg -z mercury
zonecfg:mercury> select net physical=bge0
zonecfg:mercury:net> set address=192.168.2.2
zonecfg:mercury:net> end
zonecfg:mercury> exit
```

The procedure to update the hostname-to-IP address mapping depends on the naming service being used by the zone. If the naming service uses local files, the */etc/inet/hosts* and */etc/inet/ipnodes* files should be updated manually. If a naming service such as LDAP is used, follow the steps described by the software.

- If a zone has multiple IP addresses, each address can be changed using the method described above.
- IP addresses can be added to a running zone with the `ifconfig(1M)` command and the `zone` parameter.

- The IP address can be changed with the `ifconfig` command while the zone is running. The `zonecfg` command must be used as shown above if the new IP address are to be used the next time the zone boots.

Routing

The following examples discuss routing and zones. These examples assume the route for zone1 is 192.168.1.1/24, and the route for zone2 is 192.168.2.2/24.

- *No routes configured*

  If routes are not configured, zones on the same subnet can communicate with each other. However, zones on different subnets, including the global zone, cannot communicate with one another.

- *Enabling zone communication*

  Consider two zones on the same system that are located on different subnets. In order for these zones to communicate with each other, the system must be able to identify a communication route. The route can be either a host route, such as 192.168.2.2/32, a net route, such as 192.168.2.0/24, or a default route. Creating a default route that enables zone communication requires creating a default route for the subnet. Note the command identified below can be used before or after the zone is booted.

```
global# grep zone1 /etc/hosts
192.168.2.2   zone1
global# ping zone1
no answer from zone1
global# route add default 192.168.2.1
add net default: gateway 192.168.2.1
global# ping zone1
zone1 is alive
```

- *Blocking zone communication*

  Communication between zones can be blocked using the `-reject` or `-blackhole` options to the `route` command. For example, to prevent `zone1` from sending packets to `zone2`, create `-reject` or `-blackhole` routes between each zone pair. The `-reject` option indicates that packets sent to the specified address should receive a negative acknowledgement. The `-blackhole` option indicates no response should be sent.

```
global# route add 192.168.1.2 192.168.1.3 -interface -blackhole
global# route add 192.168.1.3 192.168.1.2 -interface -blackhole
```

- *Passing traffic through a network device*

  Passing all traffic between two zones through a network device, such as a router, is not supported at this time since inter-zone traffic never reaches a network interface card.

Firewalls and Filters

The Solaris OS includes IP filters that filter network traffic from the network into a zone, or from the zone out to the network. IP filters cannot be used to filter traffic passing between zones on the same system, as inter-zone traffic remains inside the system and never reaches firewalls and filters.

Internet Protocol Multi-Pathing and Sun Trunking

Internet Protocol Multi-Pathing (IPMP) and Sun Trunking can be used to improve network bandwidth and availability. These techniques differ in several important characteristics, including failover time, bandwidth characteristics, and requirements.

Sun Trunking enables network interface ports to be grouped together to improve availability and performance. Once the trunking is complete, a zone assigned to a network port in a trunk automatically uses another port of the trunk if its port or link fails. Port failover happens quickly and is transparent to the zone and its processes.

A network address in a zone can be assigned to an IPMP group to enable it to failover. All interfaces that comprise the group must reside in the same broadcast domain. When an interface fails, its IP address(es) move to the failover interface. Once the failover completes, the zone communicates using an interface for which it was not originally configured. While this does not cause any functional problems, it may confuse users who expect the zone to only use a single interface.

If VLANs are also being used with IPMP, all of the interfaces in a group must also be part of the same VLAN. These configuration tasks must be performed by the global zone and network administrators.

Subnet Masks

Because a zone's network interfaces are configured by the global zone, netmask information must be stored in the global zone. If default subnet masks are used, the zone's subnet mask is configured correctly by default. If non-default subnet masks are used, be sure to store the mask information in the global zone's */etc/netmasks* file. Subnet masks may also be specified on the `zonecfg` command line using `/` notation, such as 192.168.3.46/24.

Printing

Little additional information is required to properly configure printing from within a zone. Non-global zone administrators configure printers within the zone as if the zone were a separate system, enabling network printer configuration without assistance. Configuring a direct attached printer requires assistance. The global zone administrator must use the `add device` subcommand of the `zonecfg(1M)` command to add the printer's device entry to the zone.

Security Risks

Before assigning direct device access to a Container, consider the following factors:

- *How trustworthy and experienced are the people who will have privileged access to the Container?*

  Keep in mind people with privileged access to a Container have the same ability to cause problems with devices as privileged users of the global zone.

- *What type of users will be able to login to the Container?*

  Any user who can run programs has the potential to cause performance problems for the Container and, possibly, other Containers. The resource management features of the Solaris OS can be used to mitigate this problem.

- *What are the availability goals for the Container and other services provided by the system?*

  In a system without Containers, it is possible for a malicious user to impact the availability of a Unix system. This is still true for a user in a Container. However, global zone users with privileged access have the potential to impact the availability of multiple Containers. As a result, applications with different availability goals should reside on different systems.

Resource Management

Several system resources and kernel parameters can be configured to create optimal system configurations.

Resource Capping

The physical memory used by a group of processes can be constrained through the resource capping features of the Solaris OS. The resource capping daemon occasionally calculates the amount of physical memory used by these processes. If the value exceeds a specified limit, the kernel pages out some of the pages owned by the processes. These actions occur independently of the kernel and other resource management controls.

Resource capping can be combined with zones to constrain the amount of physical memory used by processes in zones, or by entire zones. To configure physical memory constraints:

- Create a project for each set of processes of the zone to manage
- Specify a maximum physical memory amount for the project
- Enable resource capping

1. Create a project in LDAP, NIS, or the zone's */etc/project* file. Use the following entry format to cap physical memory in the */etc/project* file.

```
projectname:projIDnum:comment:usernames:groupnames:rcap.max-
rss=bytes
```

2. If Web server software is installed and configured to run as the user *username*, the entry might look like the following:

```
user.username:101:A web server zone:username::rcap.max-
rss=1073741824
```

3. By default, the */etc/nsswitch.conf* file specifies that the */etc/project* file is the repository for project information. As a result, that entry does not need modification.

4. Turn on resource capping using the `svcadm` command. Enabling resource capping also sets the */etc/rcap.conf* file with default values.

```
# svcadm enable rcap
```

It is important to profile the memory used by an application prior to choosing a memory cap. Most applications need a certain amount of memory at all times, called the *resident set size* (RSS). If the resource capping daemon, `rcapd`, forces the application to run with less memory than its working set size, paging activity results, reducing overall system performance. Once a program's memory usage behavior is understood, choose a memory cap for the application that is larger than its working set size but small enough to allow other applications to run well.

To find the working set size for a process, create a project for the application, choose a very large value for `rcap.max-rss`, and measure the project's memory usage with the `rcapstat` command. The following example displays the usage for a project named `user.jvictor`.

```
# rcapstat 5
id project      nproc    vm   rss   cap    at avgat    pg avgpg
100 user.jvictor 0    0K    0K 1024M    0K    0K    0K    0K
100 user.jvictor 1 1288K  908K 1024M    0K    0K    0K    0K
100 user.jvictor 1 1288K  908K 1024M    0K    0K    0K    0K
100 user.jvictor 1 1288K  908K 1024M    0K    0K    0K    0K
100 user.jvictor 1 1288K  908K 1024M    0K    0K    0K    0K
100 user.jvictor 2 2568K 1804K 1024M    0K    0K    0K    0K
100 user.jvictor 2 2568K 1804K 1024M    0K    0K    0K    0K
100 user.jvictor 2 2568K 1828K 1024M    0K    0K    0K    0K
100 user.jvictor 2 2568K 1828K 1024M    0K    0K    0K    0K
100 user.jvictor 3 3736K 2720K 1024M    0K    0K    0K    0K
100 user.jvictor 3 3736K 2720K 1024M    0K    0K    0K    0K
100 user.jvictor 3 3736K 2732K 1024M    0K    0K    0K    0K
100 user.jvictor 2 2568K 1828K 1024M    0K    0K    0K    0K
100 user.jvictor 2 2568K 1828K 1024M    0K    0K    0K    0K
100 user.jvictor 1 1288K  908K 1024M    0K    0K    0K    0K
100 user.jvictor 1 1288K  908K 1024M    0K    0K    0K    0K
```

This small example shows that the project `user.jvictor`, the default project for the user `jvictor`, should be able to function well with a 3 MB memory cap. When memory caps are enforced, the `at` column shows an integer that represents the amount of memory the `rcapd` daemon marked for page out during the sample. The `pg` column shows the actual amount of memory paged out during the sample. Note that a process may also request new pages of memory during the time period. If the `rcapd` daemon is always forcing a process to page out memory, the memory cap is probably set too low.

Resource Capping Guidelines

Several resource capping guidelines may prove useful, including:

- To change the memory cap for a project, change the `rcap.max-rss` value for the project and then use the `svcadm restart rcap` command to instruct the `rcapd` daemon to obtain the new value.

- When choosing a memory cap for a project, consider the total amount of memory in the system, the needs of all processes in the project, and the needs of all applications which will run on the system, in all zones. A project's processes are not allowed to use more than the memory allotted, even if free memory is available in the system. On the other hand, if all physical memory is allotted and another application is added to the system, at least one application will suffer from insufficient memory.
- It is very important to profile the memory usage of applications that use a lot of shared memory, such as databases. The resource capping daemon cannot distinguish between shared and non-shared pages. This leads to a larger total RSS for all database processes in a project than the actual amount of memory used by those processes.
- Set a memory cap enforcement threshold greater than zero to ensure paging does not occur when RAM is available.
- The use of resource capping can impact performance. If enough physical memory is available for the combination of applications, and if the `rcapd` daemon rarely needs to take action, the impact on performance should be negligible. Applications whose memory usage is reduced by the `rcapd` daemon suffer more, as each process is suspended while its pages are being marked for page-out.
- Do not kill the `rcapd` daemon. If the `rcapd` daemon is marking pages for page-out when it is killed, the process remains suspended until a user with sufficient privileges resumes operation of that process.

Resource Capping and Solaris Containers Technology

Solaris resource capping is not specifically zone-aware. Running the `rcapd` daemon in the global zone only enforces memory caps on global zone processes. To cap memory usage in non-global zones, enable the `rcap` service in each zone in which memory should be capped. Keep in mind the `rcapd` daemon only acts on processes in the zone in which it is running. Resource capping management may be simplified by using LDAP or NIS as the project database. In that case, each zone only needs to modify the */etc/nsswitch.conf* file and enable the `rcap` service. In this scenario, a new network naming service should be used for users to reduce the confusion caused by different users in different zones that happen to share a user ID.

When choosing memory caps for applications in different zones, consider the amount of physical memory that will be shared by multiple zones. Some memory is shared among sparse root zones because they use shared libraries. For example, programs in different sparse root zones share the physical memory space used by the `libc` library. There is one exception: per-zone services. Per-zone SMF services consume memory, and must be considered when determining the amount of physical memory for a system, as well as memory caps.

Resource Management Using Kernel Parameters

The Solaris OS uses many kernel data structures that describe entities like processes and devices. Some of these data types are limited in quantity, although most quantities can be changed by users with sufficient privileges. Because the exhaustion of limited resources could negatively impact applications running on the system, the Solaris OS includes resource controls for many of these data types. Fortunately, most Solaris OS kernel data structures are dynamic, reducing the need to actively tune these parameters.

The following sections identify the kernel parameters which can be used to alter the behavior of Solaris Containers. All Solaris OS tunable parameters are described in the *Solaris Tunable Parameters Reference Manual* available at *http://docs.sun.com*.

Processes

The consolidation of multiple systems into multiple Containers on a single system does not significantly reduce the number of total processes, and increases the number of processes one system must contain and manage. Sometimes systems need assistance understanding what limits to place on the usage of processes. With multiple Containers and workloads on a single system, the need for controls increases. The following controls are available to limit the ability of a Container or a process to create more processes or process threads.

- `pidmax`

  The kernel parameter `pidmax` sets the largest value for a process ID (PID). This value is also the maximum number of simultaneous processes. The default is 30,000 processes, and the maximum value is 999,999. The `pidmax` parameter value needs to be adjusted from the default value if more than 30,000 processes are running on all Containers in the system.

- `max_nprocs`

  The `max_nprocs` kernel parameter also controls the maximum number of simultaneous processes in the system. However, it is also used to determine the size of other system data structures, such as the size of the directory name lookup cache and the number of disk quota structures. For systems with 2 GB of RAM or more, the default `max_nprocs` value is 32,778. If the system is expected to handle more than 30,000 simultaneous processes, set this value to the same value as `pidmax`.

- `maxuprc`

  The `maxuprc` parameter specifies the maximum number of processes available to a single user. If there is concern that a user or application might consume all the PIDs in the system, and prevent other users and Containers from completing their tasks, change this from its default value (typically approximately 30,000) to a smaller value.

- `zone.max-lwps`

  The `zone.max-lwps` parameter is a zone attribute that caps the number of process threads that can exist simultaneously in a given Container. To allow a large number of Containers to co-exist, it may be necessary to increase both the `pidmax` and `max_nprocs`

parameters. To constrain the number of threads and processes that any one Container is running, use the `maxuprc` and `zone.max-lwps` parameters. The DTrace utility can also be used to limit the number of processes. Appendix B includes an example DTrace script to accomplish this task.

Virtual Memory Parameters

The Solaris 10 OS release 3/05 does not include virtual memory parameters that impact Containers. If virtual memory tuning is required, treat the workloads in multiple Containers as if the workloads co-existed in a system without Containers.

File System Parameters

Because a system that employs Containers is likely to have more workloads running simultaneously than typical systems that do not use Container technology, file system parameters may need to be tuned. Information on tuning file systems parameters for large, diverse workloads can be found in the *Solaris Tunable Parameters Reference Manual* located at *http://docs.sun.com*

Pseudo Terminal Parameters

Pseudo terminal parameters (`ptys`) are allocated dynamically. As a result, it is not necessary to tune `pty` related variables to handle Containers.

STREAMS Parameters

There are no `STREAMS` parameters that need tuning to accommodate the use of Containers.

System V IPC

Message queue, semaphore, and shared memory parameters are now project attributes. As a result, a Container can be assigned to a project, and different message queue, semaphore, and shared memory parameter values can be used in each Container. For more information, see *Chapter 6: Resource Controls* of the *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones: Resource Management* located at *http://docs.sun.com/app/docs/doc/819-2450.*

Scheduling and Other Parameters

The Fair Share Scheduler enables users to achieve most scheduling goals, and is described elsewhere in this document. In addition, the global zone `root` user can use the Real Time scheduler for a zone's processes. Other scheduling solutions are beyond the scope of this document.

IP Quality of Service

Network quality of service features can be applied to non-global zones, and must be managed from the global zone.

Resource Usage Billing with Extended Accounting

The extended accounting features in the Solaris OS provide an extensible facility for recording information about system activity on a per-process or per-task basis. Now, the extended accounting system provides additional information regarding zones. The `zonename` field is now included in records when requested. If accounting is enabled from the global zone, accounting records can be collected for all zones. Information collected is tagged with the zone name, providing greater security of accounting records if consolidation information is desired.

Oracle Software

Several constraints exist when running Oracle software in a zone. While using Solaris Intimate Shared Memory (ISM) in a zone works fine with Oracle software, the use of Dynamic Shared Memory (DISM) does not work, as a zone does not have sufficient privileges. In addition, the resource capping enforcement daemon, `rcapd`, does not understand that multiple processes can share memory, including shared memory features and common text pages. This complicates the estimation of memory needed by an application. If administrators calculate the appropriate amount of memory needed, the `rcapd` daemon functions correctly. Furthermore, Oracle Real Application Clusters (Oracle RAC) does not work in a zone. Finally, either the Fair Share Scheduler or Oracle Database Resource Manager can be used with the Oracle Database 10g software in a zone, but not both.

Provisioning and Installation

The following sections provide an overview of provisioning and installation considerations.

Sparse versus Whole Root Models

To create a whole root zone, use the `create` subcommand of the `zonecfg` command and delete the default `inherit-pkg-dir` settings. Note that `zonecfg -b` creates an empty root, not a whole root zone. The `-b` flag is useful when designing very specific customizations to zone configurations.

Package Management and Solaris Containers Technology

Packages are the software delivery mechanism used in the Solaris OS. In an effort to minimize the management burden, packages installed in the global zone are automatically installed in each Container. However, Containers can be configured to include or exclude specific packages. In addition, package creators can specify that a package be installed on all zones in the system.

- Solaris OS packages

  A system running the Solaris 10 OS should include all of the packages used by applications running on the system. In general, all Solaris OS packages installed in the global zone and applicable to zones are available to all zones and must remain at the same patch level.

- Other packages

  When deciding which non-Solaris OS packages to install, consider the following.

  - If the system only includes sparse root zones, and provides one type of service or only a few types of services, install all packages in all zones. This model simplifies administration, as there is no need to keep track of the mapping between packages and zones.
  - If the system provides many different types of services, such as Web development and database testing environments, greater control over the existence of applications within zones can provide benefits. In this scenario, install packages directly into the zones which need them. Be sure to use the `pkgadd(1M)` command in the non-global zones and track the package installations performed.

More information on package installation can be found in *Chapter 23: About Packages and Patches on a Solaris System with Zones Installed (Overview)* of the *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones* located at *http://docs.sun.com/app/docs/doc/819-2450.*

Patch Management and Solaris Containers Technology

Decisions regarding patch installation follow directly from the decisions made when installing packages. If a package is installed directly into a non-global zone, it must be patched using the same model.

Flash Archives

Flash archives can be used to provision a large number of potentially usable zones. Simply enable the ones to be used. A flash archive can be made of a zoned system using the following guidelines:

- All zones must be stopped when the flash archive is made.
- Unless the source and target systems use identical hardware configurations, device assignments must be changed after the flash archive is installed. This usually requires changing the network port assignment. All other device-specific information, such as disks, processor sets and more, must be analyzed carefully, and perhaps changed with the `zonecfg(1M)` command once the server is provisioned.
- Soft partitions created with the Solaris Volume Manager cannot be flash archived yet as the Solaris OS installer is not aware of these features.

Note – Flash archives are not zone-aware.

Security

The following sections provide an overview of security considerations. Additional information can be found in *Practical Security Using Solaris Containers in the Solaris 10 OS* available at *http://www.sun.com/bigadmin/features/articles/container_security.html*

Process Rights Management

The zones security model requires several Solaris privileges not be made available to any process or user in a non-global zone. Table 8-2 lists the privileges that are not available.

TABLE 8-2 Privileges not available to zones in the Solaris 10 OS

dtrace_kernel	proc_priocntl	sys_linkdir
dtrace_proc	proc_zone	sys_net_config
dtrace_user	sys_config	sys_res_config
net_rawaccess	sys_devices	sys_time
proc_clock_highres	sys_ipc_config	sys_user_compat
proc_lock_memory		

The absence of these privileges means that a process in a non-global zone cannot:

- Use the Dynamic Tracing Facility (DTrace)
- Use high resolution real-time timers
- Lock memory pages into physical memory
- Trace or send signals to processes in other zones
- Access the network layer directly
- Configure network devices or routing
- Perform system configuration tasks
- Install, modify, or remove device drivers
- Increase the size of a System V IPC message queue buffer
- Link and unlink directories
- Configure resource pools
- Take CPUs online and offline
- Call a third-party loadable module that calls the `suser()` kernel function to check for allowed access
- Change the system clock
- Elevate the zone's priority above its current level
- Elevate the zone's real-time scheduling class, although a user with sufficient privileges in the global zone can place a real-time process in a non-global zone

Auditing and Access Control

The Basic Security Model (BSM) and Role-Based Access Controls (RBAC) should be considered when using Solaris Containers technology.

- Basic Security Model

  While `syslog` responds to application requests, the Basic Security Module (BSM) is a kernel-based mechanism that provides kernel auditing and device allocation. Available since the release of the Solaris OS 2.3, these features enable the Solaris OS to meet C2-level criteria. It is important to recognize that enabling C2 auditing impacts performance, with estimates ranging from five to ten percent overhead per system call. In addition, audit trails can use large amounts of disk space. Basic BSM commands include `bsmconv()` and `bsmunconv()`, and must be used in the global zone. More information

can be found in the `audit(1M)` man page, as well as *Part VII, Solaris Auditing* of the *System Administration Guide: Security Services* manual located at *http://docs.sun.com/apps/docs/doc/816-4557*.

- Role-Based Access Controls

  Role-based access controls (RBAC) enable system administrators to enforce the security principles of least privilege, ensuring no user is given more privilege than is necessary to perform a job. With RBAC, administrators can create user accounts, or roles, for specific individuals, enabling a variety of security policies. Role-based access controls (RBAC) can be used in a zone just as if the zone were a separate system. Keep in mind that some of the privileges necessary for roles are not available in a zone.

- Intruder traps

  Traps used to catch system intruders, also called *honeypots*, can be implemented by creating a non-global zone with a publicly available network interface. In the unlikely event an intruder breaks into a non-global zone, the intruder is unable to detect intrusion detection tools running in the global zone.

Namespace Isolation and Naming Services

Each zone can choose its own name service. If a zone chooses to use local files as the repository for user information, then that zone only considers that information. As a result, there may be some confusion regarding file ownership. The username of a file owner may be different when the owner's name is viewed from the zone or the global zone, which maintains its own user repository.

Generally, if the computing environment includes a network-based naming service supported by the Solaris OS, it is easier to manage zone users if information is stored in the naming service. In addition, network-based naming services make it easier to handle multiple zones that share a set of users.

9

Managing the Environment

No environment is complete without management tools. The Solaris OS provides or supports a variety of tools aimed at easing the construction and maintenance of deployment environments that utilize Solaris Containers technology on Sun systems.

Sun Management Center Software

Sun™ Management Center software offers a single point of management for all Sun systems and the Solaris OS. It provides a platform upon which enterprises can base administrative and management operations, helping make the systems and the services they provide highly available. A powerful and flexible tool for managing networks, Sun Management Center software enables system administrators to easily perform remote system configuration, monitor performance, and detect and isolate hardware and software faults all through an intuitive graphical or command line interface.

Solaris Container Manager Software

An add-on to the Sun Management Center software, the Solaris Container Manager software helps maximize resource utilization and increase data center productivity by providing a central browser-based console for creating and managing Solaris Containers. The software enables administrators to easily partition a single instance of the Solaris OS into multiple containers that can each run an application, fostering improved resource utilization. With the Solaris Container Manager software,administrators can name a Container, specify its workload and accessible compute resources, and indicate how many resources are allocated to individual applications.

The Solaris Container Manager software provides several features that can help provide visibility into resource utilization, and ease the management process.

- *Centralized management*, making it easy to see how Solaris Containers are configured, as well as what resources are currently available.
- *Container replication*, enabling the same Container definition to be deployed and tracked across multiple systems—useful when applications scale horizontally across systems or when workloads on different systems require similar settings.
- *Container and process monitoring*, enabling administrators to zoom into a Container and view detailed information at any time.
- *Automatic change jobs capabilities*, enabling Container resource allocation to be scheduled on a daily, weekly, or monthly basis.
- *Usage graphs*, helping adminsitrators understand how applications and systems are behaving.
- *Alarm management*, providing the ability to set alarms for each Container so that resource contention problems can be handled proactively.

Consolidation Tool for Sun Fire Servers

The Consolidation Tool for Sun Fire Servers helps speed the deployment of Solaris Containers technology. With a wizard-based GUI, this tool simplifies and automates the installation of consolidated applications, helping administrators create fully virtualized and consolidated environments using Solaris Containers technology. In particular, the tool eases the definition and creation of Solaris Containers by deploying processor sets in a way that optimizes performance, taking advantage of Solaris Containers technology—including Solaris Zones, resource pools, processor sets, and more—and making intelligent choices between full root and sparse root systems.

Predictive Self-Healing Technology

The reliability of the system, or virtual environment, is becoming critical. Reliability has many aspects—resiliency against hardware faults, understanding what needs to be running on a system for applications to work, and making sure these system services are available to applications. In addition, ensuring the system is not running so many services that it is rendered vulnerable to intrusion or faults is a key concern. Introduced in the Solaris 10 OS, Predictive Self-Healing technology enables Sun systems to accurately predict component failures and mitigate many serious problems before they actually occur. It is designed to maximize the availability of the system and application services by automatically diagnosing,

isolating, and recovering from faults. This helps to not only reduce hardware failures but also to reduce the impact of application failures, leading to increased system and application availability.

The ability to be proactive requires two major technologies: one that enables the system to look for faults, and preferably catch them before they occur or cause data loss, and another that knows what kernel and application services are running on the system, and their dependencies, and knows how to restart them if and when those actions are needed. Toward this end, Predictive Self-Healing technology consists of two components: the Solaris Fault Manager and Solaris Service Manager software.

The Solaris Fault Manager receives data relating to hardware and software errors and automatically diagnoses the underlying problem. Once diagnosed, the Solaris Fault Manager automatically responds by offlining faulty components. The Solaris Service Manager ensures application service level availability is maintained at the operating system level. From the perspective of the operating system, applications become services that are managed and monitored as first-class citizens, enabling automatic service recovery even in the face of a catastrophic failure. Services included in the Solaris OS are pre-defined within the service management framework, including full dependency information, and third-party and user-developed applications can be added easily to the framework without source code modification.

Solaris Containers and Predictive Self-Healing Technology

The combination of virtualization and reliability is especially important when consolidating applications. This section describes how Solaris Containers and Solaris Predictive Self Healing features work together to address these needs, and provides some ideas on how to get started and put these new technologies to work. Emphasis is placed on illustrating how this functionality can be used to create isolated environments customized for specific applications.

Solaris Containers and Predictive Self-Healing technologies work together by creating separate execution environments, each with their own namespace and assigned resources. Each environment can have its own self-healing personalities—personalities that can be changed, copied, and reloaded as needed. In addition, these technologies enable administrators to determine the current state of the environment, making it easier to use the Solaris OS for consolidation efforts.

Solaris Service Manager Software

On a system, the Solaris Service Manager software is represented by the Service Management Facility `smf(5)`. The Service Management Facility is an infrastructure that provides several functions:

1. Definition of services for the Solaris OS, which can be the state of a device, a running application, or a set of other services. Each service is referred to by a unique identifier.
2. A formal relationship between services, with explicit dependencies.
3. Automatic starting and restarting of services.
4. A repository for storing service state and configuration properties, eliminating the need for dozens of configuration files scattered throughout the system.

At a high level, the system is managed by a master *restarter* named `svc.startd`. This daemon enforces dependencies, starts and stops services, and basically keeps an eye on how the machine is running. All related configuration information is stored in a *repository* on the system, and is managed by the `svc.configd` daemon. One or more *delegated restarters* are given a subset of services to manage, and are written specifically to deal with this subset. For example, the `inetd` daemon manages most networking services as a delegated restarter.

Services

A *service* is the fundamental unit of the Service Management Facility. Each service can have one or more *instances*, specific configurations of a service. For example, an Apache daemon configured to serve *www.sun.com* on port 80 is an instance of the Apache service. The Apache software may have several instances, each with a different configuration. The service holds basic configuration properties that are inherited by each of its instances. However, each instance can override configuration properties as needed. Special services, called *milestones*, correspond to a specific system state, such as *basic networking* or *local file systems available*. Milestones are essentially a list of other services, and are considered to be online when each of their component parts is online.

Each service is identified with a Fault Management Resource Identifier (FMRI), an unique identifier representing a service or instance. For example, the telnet service is represented by `svc:/network/telnet:default`, where `svc:/network/telnet` describes the service, and `default` describes a specific instance. FMRIs can be a bit of a handful to type. As a result, most SMF commands accept shortened versions of a service's FMRI, given that it only has one instance. For example, most utilities accept `network/telnet` as the FMRI for telnet, since it comes installed with only one instance.

Note that `telnet` is preceded with the word `network`. The Service Management Facility contains several categories for services to provide naming organization and uniqueness. This categorization enables administrators to quickly determine where a services lives, and to what it is related. Standard categories include:

- `application`
- `device`
- `milestone`
- `network`
- `platform`
- `site`
- `system`

States

Each service on a machine is always in one of seven discrete states that are observable by the Service Management Facility CLI tools. These states include:

- `degraded`, while the service is running, something is wrong or its capacities are limited in some way
- `disabled`, the service has been disabled and is not running
- `legacy_run`, a legacy *rc***.d* script has been started by the system and is running
- `maintenance`, the instance encountered an error and needs to be repaired by an administrator
- `offline`, the service is enabled but not running usually because a service it depends on is not online
- `online`, the service is both enabled and running successfully
- `uninitialized`, the `svc.startd` daemon has not yet read the service configuration

Manifests

One of the powerful features of the Service Management Facility is that it knows the relationships between different services on the system and how they are related. *Manifests* are the mechanism that enable the system to learn about these relationships. An SMF manifest is an XML file describing a service. All the manifests in the system are stored in */var/svc/manifest* under categorical subdirectories. If custom services are not intended to be converted to the SMF model, these files do not need to be edited. However, these files can provide a helpful reference.

During the boot process, the `svc.configd` SMF daemon looks in the manifest directory. If new manifests exist, the daemon imports them into the repository. This can also be done manually by administrators. The entire system is run using information in the repository, not the manifests—manifests are simply a delivery mechanism for service descriptions. An active system is administered using the SMF command line tools.

Compatibility

These technological advancements came with an important design goal—ensure all of the layered software installed and functioned just as it did in previous Solaris OS versions. Many Solaris OS users rely on scripts and services they have carefully honed over time. While there are advantages to converting these services to take advantage of the benefits of SMF, it is not required. Custom scripts located in */etc/rc*.d* continue to execute on run-level transitions. However, some Solaris OS kernel scripts are already converted and no longer need to be used.

The service states include the `legacy_run` state, which is used to identify services started through the old */etc/rc*.d* mechanism. The SMF observational tools use this state to identify legacy services. When a service is labeled with the `legacy_run` state, it means the script was located in a */etc/rc*.d* directory that was run upon successful transition to a run level.

Many standard Solaris OS services are already converted to SMF. As a result, there are fewer scripts in the */etc/init.d* and */etc/rc*.d* directories than in previous operating system releases.

Starting and Restarting Services

When the system understands how to start and stop services, as well as dependencies between services, a dependency tree can be built. This dependency tree is not new—in many ways, the */etc/rc*.d* scripts are another expression of this tree. However, the dependency tree and the state of all services is now maintained in the kernel. For the first time, the kernel knows the direct relationship between the service and process namespaces.

The implications are far reaching. For example, when a process exits for any reason, including the accidental or intentional killing of the process, it is terminated by the kernel. Since the kernel now knows whether or not the process represents a service instance that needs to continue, it can immediately intervene and restart the service instance as another process, if needed. Additionally, the kernel understands the relationship of this process to other services on the system, including how they may or may not be impacted by the change. As a result, systems are less likely to run unintentionally in a degraded state when services are signed up for SMF. Systems can now essentially self-heal, or inform administrators through SMF interfaces in the event they cannot do so.

In addition, it is possible to turn enable (turn on) and disable (turn off) services, and understand the repercussions these actions will have on the other services running on the system. For example, the system can be told to enable a specific service, and in the same action enable and start all the services upon which it depends. Similarly, if a service is running and its configuration file is changed, the system can refresh the service and ensure it runs with the new values. If a service is disabled, generally it is for a good reason. For example, administrators may disable the `telnet` service in order to keep individuals from logging into the system remotely using the `telnet` command. The dependency tree enables administrators to identify the other services affected and mitigate the impact.

Service Profiles

SMF also includes the ability to save a service profile. Recall that SMF maintains a list of all services and their instances on the system, including whether or not those services are enabled or disabled. In essence, this list, or *service profile*, defines the personality of the system. SMF gives administrators the ability to capture the current service profile, load a new profile, or revert to an old profile. Whenever a system with a particular set of behaviors (or personality) needs to be created, it can be done so by replicating this list.

Working with SMF

The best way to understand how SMF works is to walk through some examples. This section provides an overview of the main SMF commands, or *administrative interfaces*, and shows some of them in action.

Administrative Interfaces

A lot of time and effort was put into making the administration of a system running the Solaris OS with SMF as painless as possible. Users no longer need to run the `grep` command and search for processes in output listings, or wonder if those processes are running, or hunt for configuration files. Now, SMF service administration is performed through a central interface, enabling administrators to observe the state of services and their dependencies and properties, and make changes to services.

TABLE 9-1 The `smf(5)` command line interface tools.

Command	Description
`svcs(1)`	Enables administrators to observe the state of all service instances on the system, and provides detailed views of service dependencies, processes, and more
`svcadm(1M)`	Provides service administration, including the ability to enable, disable, and restart services
`inetadm(1M)`	Enables administrators to observe and configure services that are controlled by the `inetd` daemon
`svccfg(1M)`	Enables administrators to manipulate the contents of a repository, usually properties in a service
`svcprop(1)`	Enables property values to be observed in a read-only manner; outputs are formalized for easy use in shell scripts

The Basics

To understand the nature of SMF, start with the `svcs(1)` command. A versatile command, `svcs(1)` is likely to be used in day-to-day administrative work. The basic output is a list of all services on the system that should be running, including the service state, the time the instance started, and the full FMRI.

```
# svcs
STATE          STIME    FMRI
legacy_run     22:36:00 lrc:/etc/rc2_d/S20sysetup
...
legacy_run     22:36:02 lrc:/etc/rc3_d/S90samba
online         22:35:55 svc:/system/svc/restarter:default
...
online         22:36:01 svc:/milestone/multi-user:default
online         22:36:02 svc:/milestone/multi-user-server:d efault
offline        22:35:56 svc:/application/print/ipp-listene
r:default
offline        22:35:59 svc:/application/print/rfc1179:def ault
#
```

The `svcs -a` command produces a similar but longer list—it now details all the services that could be running, including those that are not enabled. Output from this command include lines like the following:

```
disabled       22:35:56 svc:/network/nfs/cbd:default
```

An In-Depth Look

Let's take a look at the `ssh` service. The implications of the state of the `ssh` service help illustrate SMF concepts—either the `ssh` service exists and it is possible to log into the system, or the service is not present.

1. Check the status of the `ssh` service. Because the ssh service is unique, it is not necessary to use the full FMRI.

```
# svcs ssh
STATE          STIME    FMRI
online         22:36:00 svc:/network/ssh:default
```

2. Obtain the full listing for the **ssh** service using the **-l** option of the **svcs** command. The output shows more information on the current state of the **ssh** service, as well as the services upon which it depends, the type of dependency, and the status of those services.

```
# svcs -l ssh
fmri         svc:/network/ssh:default
name         SSH server
enabled      true
state        online
next_state   none
state_time   Sun Feb 03 22:44:10 2006
logfile      /var/svc/log/network-ssh:default.log
restarter    svc:/system/svc/restarter:default
contract_id  153
dependency   require_all/none svc:/system/filesyst em/local
(online)
dependency   optional_all/none svc:/system/filesys tem/autofs
(online)
dependency   require_all/none svc:/network/loopbac k (online)
dependency   require_all/none svc:/network/physica l (online)
dependency   require_all/none svc:/system/cryptosv c (online)
dependency   require_all/none svc:/system/utmp (onl ine)
dependency   require_all/restart file://localhost/e tc/ssh/
sshd_config (online)
```

3. Alternatively, obtains the status of the **ssh** service and lists all the processes representing the service instance using the **ssh** command. Note the number 874 is the process ID (PID) of the **sshd** daemon.

```
# ssh -p ssh
STATE          STIME    FMRI
online         22:36:00 svc:/network/ssh:default
               22:36:00      874 sshd
```

4. Verify that a user can connect to the system from a different machine with the `ssh` command. If the login is successful, everything is working fine.

```
[ other_system #] ssh zone1
Password:
Last login: Sun Feb 3 22:37:10 2006
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
#
```

5. Disable the `ssh` service using the `svcadm(1M)` command.

```
# svcadm disable ssh
# svcs -p ssh
STATE          STIME    FMRI
disabled       22:42:33 svc:/network/ssh:default
```

6. Attempt to log into to system from another machine. If the `ssh` service is disabled the action should fail.

```
[ other_system #] ssh zone1
ssh: connect to host zone1 port 22: Connection refused
[ other_system #]
```

7. Enable the `ssh` service using the `svcadm(1M)` command, and verify the service is running. Note the offline state indicates the ssh service is enabled but is waiting for another service to go online.

```
# svcadm enable ssh
# svcs -p
STATE          STIME    FMRI
offline        22:43:40 svc:/network/ssh:default
```

8. Identify the list of services upon which the `ssh` service depends using the `svcs` command. The sample output indicates the `ssh` service depends on the cryptographic services (with a `require_all` dependency). This service is disabled here to illustrate `svcs` functionality.

```
# svcs -d ssh
STATE          STIME    FMRI
disabled      22:43:27 svc:/system/cryptosvc:default
online        22:35:57 svc:/network/physical:default
online        22:35:57 svc:/network/loopback:default
online        22:35:58 svc:/system/filesystem/local:default
online        22:35:59 svc:/system/utmp:default
online        22:36:00 svc:/system/filesystem/autofs:default
```

9. Enable the `ssh` service as well as all the services upon which it depends using the `-r` option of the `svcadm` command. Note the `-v` option enables verbose mode.

```
# svcadm -v enable -r ssh
svc:/network/ssh:default enabled.
svc:/system/filesystem/local enabled.
svc:/milestone/single-user enabled.
svc:/system/identity:node enabled.
svc:/network/loopback enabled.
svc:/system/filesystem/minimal enabled.
svc:/system/filesystem/usr enabled.
svc:/system/filesystem/root enabled.
svc:/system/device/local enabled.
svc:/milestone/devices enabled.
svc:/system/manifest-import enabled.
svc:/network/physical enabled.
svc:/system/cryptosvc enabled.
svc:/system/filesystem/minimal:default enabled.
svc:/system/utmp enabled.
svc:/milestone/sysconfig enabled.
# svcs -p ssh
STATE          STIME    FMRI
online         22:44:10 svc:/network/ssh:default
               22:44:10     1148 sshd
```

10. Log into the system from another machine to verify the **ssh** service is running.

```
[ other_system #] ssh zone1
Password:
Last login: Sun Feb 3 22:42:07 2006 from 10.9.2.100
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
#
```

Note that using the **-D** flag identifies which services are directly dependent on a service.

Using Service Profiles

At its simplest, a service profile is a list of enabled services. Because services influence the way a system reacts to events, a service profile describes the system personality. Among other things, the **svccfg(1M)** command enables administrators to export this list so it can be saved, or load a previously saved list so the system can adopt that personality. Exported lists are formatted in XML.

```
# svccfg extract
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM '/usr/share/lib/xml/dtd/
service_bundle.dtd.1'>
<service_bundle type='profile' name='extract'>
  <service name='system/console-login' type='servi ce'
version='0'>
    <instance name='default' enabled='true'/>
  </service>
...
</service_bundle>
```

Ideally, a service profile should be saved in a file in the */var/svc/profiles* directory, where the system houses several profiles. This listing with pre-configured profiles enables administrators to get quickly to a particular system state. For example, the **generic_limited_net.xml** profile configures the system with a basic network state, turns off all services that transmit passwords in clear text, and more.

```
# svccfg extract > /var/svc/profile/myprofile.xml
# ls /var/svc/profile
generic.xml                 ns_ldap.xml
generic_limited_net.xml     ns_nis.xml
generic_open.xml            ns_nisplus.xml
inetd_generic.xml           ns_none.xml
inetd_services.xml          platform.xml
inetd_upgrade.xml           platform_i86pc.xml
myprofile.xml               platform_none.xml
name_service.xml            prophist.SUNWcsr
ns_dns.xml                  ns_files.xml
```

Use the **svccfg** command to load a service profile. Use the **-v** option to list all the services this action enables, disables, or updates. Using this tool, administrators can bring systems to a known state. This can be helpful when attempting to move a system to a secure state, or enabling a server to diagnose another system.

```
# svccfg apply /var/svc/profile/myprofile.xml
```

Restarters

Predictive Self-Healing does as its name implies—it enables systems to be regenerated automatically after a fault occurs. A portion of this functionality is handled by the Fault Manager, with the remainder handled by the Service Manager. Consider the following self-healing example.

1. Identify a process that can be made to fail, such as the **sendmail** process.

```
# svcs -p sendmail
STATE          STIME    FMRI
online         22:36:00 svc:/network/smtp:sendmail
               22:36:01      913 sendmail
               22:36:01      914 sendmail
```

2. Inject a fault by killing the process. In this example, the `sendmail` process abruptly terminated, perhaps as a result of running on a failing hardware component, such as a DIMM.

```
# pkill -9 sendmail
```

3. Check to see how the service is functioning. Note the process IDs (PIDs) have changed, and the service is running once again.

```
# svcs -p sendmail
STATE          STIME    FMRI
online         22:48:04 svc:/network/smtp:sendmail
               22:48:04     1188 sendmail
               22:48:04     1189 sendmail
```

While short, this example illustrates the power of this tool—the fault was identified and corrected very quickly. Keep in mind the fault did not lie within the operating system or a zone or require a system or zone reboot. It was located on a small boundary, the process that could be controlled by the restarter for the service.

Working Together

How do Solaris Containers and Predictive Self-Healing technologies work together?

- Every zone has its own Service Manager. As a result, every zone can have its own service profile. Consider a zone that is running a service that needs the `telnetd` daemon, and another zone needs to turn the `telnetd` daemon and other service off for security reasons. This is easily accomplished by both global and local administrators.
- System administrators can create a development Container for determining which services are needed for an application to run. A service profile can then be created for the Container. When moving to a production system, administrators simply need to install the Container—both the zone and resource management components—and apply the service profile. These steps can be reproduced many times, and assure the system is in a known state. When in doubt, administrators can reapply the service profile and revert to this known state, or a debug state with different services enabled or disabled.
- The Service Manager aims to contain faults at the service level. Every zone has his own Service Manager, and so has its own restarters. However, administrators can always choose to reboot the zone, or go to a different init level, if needed.
- Zones also have their own repositories. As a result, administrators can add own unique manifests to the environment. Every time the zone reboots, it checks for new manifests and loads them into the repository.

- If it is important to give services within a particular Container a specific resource guarantee, or limit, services can also be assigned to a project. Consequently, when the Container boots, the service automatically starts in the right project and receives the correct resources, such as CPU fair shares.

10

Troubleshooting

If a zone is not behaving as expected, investigate the cause of the behavior. While most popular tools are available for use in a zone, there are some important differences.

Methods to Access a Troubled Zone

The first step in troubleshooting a zone involves accessing the system and the troubled zone. Several troubleshooting tools are available, including graphical user interfaces (GUIs) like the Sun Management Center software and the Solaris Container Manager software, as well as command line methods for zone access. Note the command line methods require the zone to run `telnet`, a user login, and a console login.

Telnet and Shells

Users can `telnet` into a zone from another zone. This includes the ability to `telnet` into a zone from the global zone, or from another system. Under normal conditions, users can also use the `ssh`, `rsh`, `rlogin`, and `ftp` commands to access a zone as if it were a separate system. However, some situations prevent these methods from working properly.

User Login with the `zlogin` Command

Users can login to a zone from the global zone on the same system if appropriate privileges are granted.

```
global# zlogin mercury
[ Connected to zone 'mercury' pts/5]
Last login: Thu Sep  8 06:04:42 on pts/5
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
#
```

Logging in as a specific user is also supported. The user must be defined in the zone, or a directory service that defines the user must be available to the zone.

```
global# zlogin -l username mercury
[ Connected to zone 'mercury' pts/5]
Last login: Thu Sep  8 06:15:34 on pts/5
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
```

It is possible to run a command in a running zone from the global zone. Privileges must be granted to the user to allow this capability. For example, running the `ls -l` command in the zone `mercury` as the user *username* behaves as if the user was logged into the system. The files in the user's home directory are listed. For user logins to work, the zone must be running and capable of accepting user logins. If user logins fail to work, the login service is experiencing a problem. Other methods, described below, can be useful in this situation.

```
global# zlogin -l username mercury /bin/ls -l
total 16
drwxr-xr-x   2 username 1001      70 Sep  8 06:29 Documents
drwxr-xr-x   2 username 1001      70 Sep  8 06:35 Sources
#
```

Zone Console Login

Unix systems have long supported consoles via direct attached serial lines or virtual consoles accessed via the network. Each zone has a virtual console. If a zone is running and traditional login methods fail, it may be possible to access the zone via its virtual console using the `zlogin -C` command.

```
global# zlogin -C mercury
[ Connected to zone 'mercury' console]
<at this point you must press 'return' to get a login prompt>

mercury console login:
```

Only one virtual console exists per zone. Users can exit the `zlogin` session without logging out of the console, leaving the currently running program or shell available for later use via another `zlogin -C` session. Exiting a `zlogin` session is similar to exiting a `tip` session. Simply type `~.` and press the `Enter` key on a new line.

Safe-Mode Login

Users that cannot login to the console of a running zone can try using a safe-mode login. This creates a shell which exists in the zone, but does not go through the login process. Safe-mode login is achieved with the `zlogin -S` command.

```
global# zlogin -S mercury
[ Connected to zone 'mercury' pts/5]
#
```

Boot Single User

If there is a problem in the boot sequence for a zone, boot the zone in single user mode using the `zoneadm` command. The zone will have `running` status. Use the `zlogin -C` command to log into the zone console. When a zone is running in single user mode, users can use the `zlogin -l` command to login as any user.

```
global# zoneadm -z mercury boot -s
global#
```

Network Troubleshooting

Common troubleshooting can be used with zones. However, a non-global zone cannot use `snoop` to view network traffic, and `snoop` cannot be used to view inter-zone network traffic, even from the global zone.

11

Putting It All Together—Consolidating Enterprise Applications with Solaris Containers

Improving the manageability and efficiency of enterprise infrastructure services poses a significant challenge to many organizations. The problem is compounded by the proliferation of individual servers used to run key enterprise applications including directory, portal, identity, mail, and calendar services. A recent Gartner report[1] indicates that the power, space, and heat requirements of rackmounted servers are limiting growth and cost efficiency among 80 percent of enterprise data centers. Consolidating enterprise infrastructure services that run on multiple servers onto the Sun Fire T2000 platform using the Sun Java™ Enterprise System (Java ES) software and Solaris Containers technology can simplify management, improve performance, and increase the efficiency of delivering enterprise infrastructure services.

About the Sun Fire T2000 Server

The Sun Fire T2000 server is a high-density compute server platform based on the UltraSPARC T1 processor. The major benefits of this platform are high aggregate throughput performance with very efficient power, cooling, and space consumption. At the center of this new platform is the UltraSPARC T1 processor, which combines chip multiprocessing and hardware multithreading to provide a thread-rich environment for improved scalability for many applications. This new processor hardware architecture provides the following features in a single-chip package:

- Eight cores or individual execution pipelines
- Four threads per core providing a total of 32 active thread contexts
- Each core has separate level 1 Instruction and data caches shared by the four threads
- All eight cores share a unified level 2 cache on chip
- Memory is unified to provide low latency to all cores
- The processor is fully SPARC V7, V8, and V9 binary compatible

1. *Gartner 12-04-2005 Report*, http://www.gartner.com

In addition to the UltraSPARC T1 processor, the Sun Fire T2000 platform supports up to 32 GB of DDR2 SDRAM memory, four 1000 Base-T on-board network interfaces, and up to four 73 GB SAS disk drives. In a rack-optimized 2RU enclosure that typically draws 325 watts of power, the Sun Fire T2000 provides a high-performance, low-power alternative to many x64-based systems.

About Sun Java Enterprise System Software

Many enterprise computing environments utilize a variety of products from a number of vendors to deliver infrastructure services such as directory, email, calendar, identity, and portal services. This can result in a host of potential problems, including high acquisition costs, unnecessary deployment delays, compatibility and interoperability issues, unpredictable schedules, expensive licensing agreements, complicated version control and release schedules, and more.

Sun has taken a radical new approach to software infrastructure to help reduce costs throughout the IT project life cycle from acquisition, through deployment, and on to operation and maintenance. The Sun Java Enterprise System (Java ES) offers a single, comprehensive software system containing all of the critical enterprise infrastructure components every business needs to build applications and services. Java ES is an open, integrated, enterprise infrastructure software suite that offers customers unique advantages, including:

- Seamless integration of many important infrastructure software components, resulting in substantial time savings from evaluation, integration, and configuration issues, and enabling better focus on business problems.
- Open standards-based software components.
- A common installer, which makes it easy to install and configure the software components.
- An attractive price point for all software components.

Deploying Java ES 2005-Q4 on a Sun Fire T2000 Server Using Solaris Zones

When used together, the UltraSPARC T1 processor, Solaris Containers, and Sun Java Enterprise System technologies can greatly improve the management, performance, and efficiency of an enterprise infrastructure. This section describes a particular methodology for using Java ES and Solaris Containers to consolidate key enterprise applications onto a single Sun Fire T2000 server.

Several Java ES servers are essentially Web applications that need an underlying Web container. Deploying these applications on a single Web container is not a scalable solution. Testing clearly shows that such deployment cannot make full use of the vast compute and memory resources offered by the Sun Fire T2000 platform. Solaris Zones offer an easy way to deploy these applications on different Web containers, each Web container, in turn, hosted on a different zone. In other words, each of the Java ES applications that needs an underlying Web container is deployed on a different instance of a Web container. When deployed in different Web containers, Java ES applications do not contend with each other for heap space. This deployment allows good utilization of the compute and memory resources offered by the Sun Fire T2000 server. Testing shows that such a deployment can support nearly three times as many users compared to the number of users supported on a deployment with a single Web container.

The consolidation examples discussed below uses six unique zones, with each zone configured to host a Java ES component. In this example, Java ES components are assigned to the following zones:

- Zone 1: Directory server
- Zone 2: Access Manager
- Zone 3: Portal server
- Zone 4: Messaging server, including the message transfer agent (MTA)
- Zone 5: Calendar server
- Zone 6: Communications Express

Figure 11-1 shows the architecture of the Java ES deployment using Solaris Zones technology. Arrows indicate the flow of requests among the Java ES components. Dotted line arrows indicate LDAP requests.

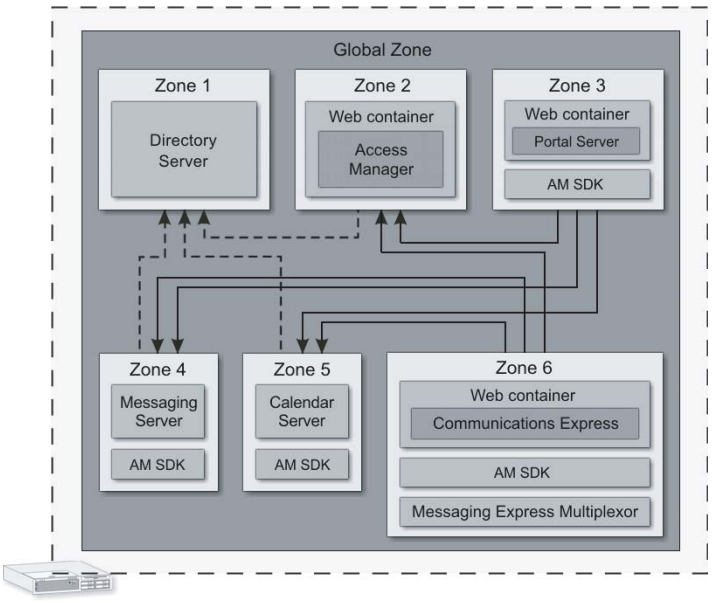

FIGURE 11-1 Java ES deployment on Solaris Zones

While Solaris Zones provide a virtual environment to shield each of the Java ES components, by default all zones have access to the full set of logical CPUs enabled in the system. To ensure that no single zone utilizes excessive CPU cycles, the Solaris OS provides the capability to create and bind resource pools to each zone. Testing showed the default configuration to be adequate for this Java ES deployment.

Configuring Solaris Zones

The following steps illustrate how to configure the zones used in this deployment. For more detailed information on configuring zones, see the *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

1. Create a zone for each Java ES component using the `zonecfg` command. This example creates a zone named `jes_zone1`.

```
global # zonecfg -z jes_zone1
jes_zone1: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:jes_zone1> create
zonecfg:jes_zone1> set zonepath=/export/home/zones/jes_zone1
zonecfg:jes_zone1> set autoboot=true
zonecfg:jes_zone1> add fs
zonecfg:jes_zone1:fs> set dir=/export/home/public
zonecfg:jes_zone1:fs> set special=/export/home/public
zonecfg:jes_zone1:fs> set type=lofs
zonecfg:jes_zone1:fs> end
zonecfg:jes_zone1> add fs
zonecfg:jes_zone1:fs> set dir=/data
zonecfg:jes_zone1:fs> set special=/data
zonecfg:jes_zone1:fs> set type=lofs
zonecfg:jes_zone1:fs> end
zonecfg:jes_zone1> add net
zonecfg:jes_zone1:net> set address=10.6.221.115
zonecfg:jes_zone1:net> set physical=ipge0
zonecfg:jes_zone1:net> end
zonecfg:jes_zone1> add attr
zonecfg:jes_zone1:attr> set name=comment
zonecfg:jes_zone1:attr> set type=string
zonecfg:jes_zone1:attr> set value="JES Zone 1"
zonecfg:jes_zone1:attr> end
zonecfg:jes_zone1> remove inherit-pkg-dir dir=/lib
zonecfg:jes_zone1> remove inherit-pkg-dir dir=/platform
zonecfg:jes_zone1> remove inherit-pkg-dir dir=/sbin
zonecfg:jes_zone1> remove inherit-pkg-dir dir=/usr
zonecfg:jes_zone1> verify
zonecfg:jes_zone1> commit
zonecfg:jes_zone1> exit
```

The `zonepath` parameter specifies where the local zone root file system is created. Although sparse zones are used by default, it is necessary to use whole root zones to install and configure Java ES 2005-Q4 on zones. A whole root zone takes up much more space than a sparse zone, but it provides a great deal of flexibility. Inside whole root zones files and packages can be removed easily, which is not always possible within sparse zones.

The `remove` commands used in the preceding example enable the configuration of whole root zones. Note that the physical devices specified for each net resource configured for the new local zone were already configured in the global zone with a separate IP address. When the zone is booted, the `ifconfig` utility reports the new interface as `ipge0:1`. In the global zone, both interfaces `ipge0` and `ipge0:1` are reported, however, only `ipge0:1` is visible in the local zone. The */data* file system can be used for storage purposes such as Java ES configuration data and user calendar and mail data.

2. Install and boot the zone by using the `zoneadm` utility as follows:

```
global # zoneadm -z jes_zone1 install
Preparing to install zone <jes_zone1>.
Creating list of files to copy from the global zone.
...
global # zoneadm -z jes_zone1 ready
global # zoneadm -z jes_zone1 boot
```

3. Log in to the new zone using the console option and configure the name and password information for this zone. Use the DNS service when configuring the zones for Java ES instead of NIS. In the following example, the host name is set to `jes-zone1`.

```
global # zlogin -C jes_zone1

SunOS Release 5.10 Version Generic_118822-22 64-bit
Copyright 1983-2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Hostname: jes-zone1

jes-zone1 console login:
```

4. Repeat Steps 1 through 3 to create the zones `jes_zone2`, `jes_zone3`, `jes_zone4`, `jes_zone5`, and `jes_zone6`. (Note the parameters for the net resource must be changed for each zone.). The Sun Fire T2000 server is configured with four onboard controllers named `ipge0` through `ipge3`. This deployment uses only one network interface. However, based on the deployment requirement, all network interfaces can be used. For instance,

the interface `ipge0` could be used for portal server traffic, and interface `ipge1` for messaging server traffic. Doing this fosters better network throughput and improves overall performance. (One gigabit network interface was adequate for this deployment.)

5. Verify that the zones are ready for Java ES deployment by using the `zoneadm` utility as shown in the following example.

```
global # zoneadm list -cv
 ID NAME           STATUS        PATH
   0 global        running       /
   1 jes_zone3     running       /export/home/zones/jes_zone3
   2 jes_zone2     running       /export/home/zones/jes_zone2
   3 jes_zone6     running       /export/home/zones/jes_zone6
   4 jes_zone1     running       /export/home/zones/jes_zone1
   5 jes_zone5     running       /export/home/zones/jes_zone5
   6 jes_zone4     running       /export/home/zones/jes_zone4
```

6. Log in to any of the newly created zones and verify that each has a network interface and that the mount points */export/home/public* and */data* are visible. This deployment stores all the configuration and user data including user mail files in the */data* directory.

```
global # zlogin jes_zone1
[ Connected to zone 'jes_zone1' pts/2]
Last login: Sun Nov  6 09:23:20 on pts/3
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
#  zonename
jes_zone1
# ifconfig -a
lo0:1: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
ipge0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.6.221.115 netmask ffffff00 broadcast 10.6.221.255
# df -kl
Filesystem            kbytes    used     avail    capacity  Mounted on
/                   42342570  20667793  21251352     50%    /
/dev                42342570  20667793  21251352     50%    /dev
/export/home/public 42342570  20667793  21251352     50%    /export/home/public
/data               206448473 98558248  105825741    49%    /data
proc                0         0         0            0%    /proc
ctfs                0         0         0            0%    /system/contract
swap                18027624  272       18027352     1%    /etc/svc/volatile
mnttab              0         0         0            0%    /etc/mnttab
fd                  0         0         0            0%    /dev/fd
swap                18027424  72        18027352     1%    /tmp
swap                18027376  24        18027352     1%    /var/run
```

Monitoring and Managing Zones

The Solaris OS contains a number of utilities that are zone aware, enabling the efficient monitoring and management of CPU resources. Use the `prstat` and `mpstat` facilities to report the CPU utilization for each zone. For example:

```
global # prstat -Z
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 29096 root     3525M 2293M cpu24   49    0   2:01:09  27% webservd/958
 29007 root      279M  229M cpu26   53    0   2:28:44  23% ns-slapd/54
 29251 root     3507M 1559M run     49    0   0:44:06 7.4% webservd/500
 29046 root     3177M 1937M cpu7    54    0   0:39:18 5.3% webservd/598
 29101 bs202758  192M  100M cpu7    59    0   0:12:16 2.5% imapd/9
 29098 bs202758  184M   92M run     40    0   0:11:55 2.5% imapd/6
 29099 bs202758  187M   95M run     40    0   0:11:50 2.5% imapd/6
 29102 bs202758  187M   95M cpu22   40    0   0:11:47 2.2% imapd/6
 29095 bs202758  186M   90M cpu10   40    0   0:11:50 2.1% imapd/6
 29100 bs202758  186M   96M sleep   55    0   0:11:40 1.9% imapd/6
 29132 bs202758  172M  141M cpu3    52    0   0:08:45 1.6% cshttpd/34
 29133 bs202758  176M  145M cpu4    59    0   0:08:38 1.2% cshttpd/38
 29130 bs202758  176M  145M cpu1    59    0   0:08:50 1.1% cshttpd/41
 29131 bs202758  180M  149M sleep   59    0   0:08:55 0.9% cshttpd/40
 29126 bs202758  172M  142M run     41    0   0:08:46 0.9% cshttpd/38
 29134 bs202758  172M  141M sleep   59    0   0:08:18 0.7% cshttpd/36
 29111 bs202758   59M   29M sleep   59    0   0:02:05 0.4% csadmind/9
 29116 bs202758  233M  140M cpu5    59    0   0:01:57 0.3% mshttpd/8
 29114 bs202758  234M  139M cpu25   60    0   0:01:51 0.3% mshttpd/11
 29107 bs202758  212M  115M sleep   59    0   0:01:43 0.2% mshttpd/8
 29115 bs202758  242M  143M sleep   59    0   0:01:54 0.2% mshttpd/8
ZONEID    NPROC  SIZE   RSS MEMORY      TIME  CPU ZONE
     2       38 4312M 2466M   7.6%   2:04:11  27% jes_zone3
     1       34 1019M  372M   1.1%   2:30:25  23% jes_zone1
     6       92 5673M 2928M   9.0%   1:47:09  15% jes_zone4
     3       41 4591M 1817M   5.6%   0:48:46 7.5% jes_zone6
     5       41 1889M 1074M   3.3%   0:56:03 6.7% jes_zone5
     4       34 3939M 2105M   6.5%   0:41:12 5.4% jes_zone2
     0       48  206M   58M   0.1%   0:01:02 0.0% global
Total: 328 processes, 3988 lwps, load averages: 44.00, 38.04, 25.79
```

Overview of Deploying Sun Java Enterprise System 2005-Q4 on Solaris Zones

The instructions in this section are not intended to act as a replacement for the individual component documentation, or the Java ES documentation, but merely to serve as a guide. For more detailed instructions or information on different deployment scenarios refer to the Sun Java Enterprise System Documentation.

This deployment uses Java Enterprise System 2005-Q4. Performance testing was done on the following Java ES components:

- Sun Java System Messaging Server 6 2005-Q4
- Sun Java System Calendar Server 6 2005-Q4
- Sun Java System Portal Server 6 2005-Q4
- Sun Java System Communications Express 6 2005-Q4
- Sun Java System Directory Preparation Tool
- Communication Services Delegated Administrator 6 2005-Q4
- Sun Java System Web Server 6.1 SP5 2005-Q4
- Sun Java Access Manager 7 2005-Q4
- Sun Java System Directory Server 5 2005-Q4
- Sun Java System Administration Server 5 2005-Q4

The following sections serve as a guide for the various stages involved in Java ES deployment. The example installation described below uses the following example data that must be changed for each individual installation:

- Domain name: map.beta.com
- Zone host name(s): `jes-zone1`, `jes-zone2`, .. through `jes-zone6`
- Password: password used for all passwords except the `amldapuser` password

Deploying Directory Server in Zone-1

1. Log in to `zone-1` as `root` and start the Java ES installer as follows:

```
jes-zone1# installer
```

2. Select the following products to install: Sun Java System Directory Preparation Tool, and Sun Java System Directory Server 5 2005-Q4.

3. Select the common server settings as shown in the following table during the installation.

Option	[Default Value]	Enter
Installation type	Custom installation	accept default
Common server settings		
Host name	`jes-zone1`	accept default
DNS domain name	`map.beta.com`	accept default
Host IP address	Machine IP address	Verify correct address
Admin user ID	`admin`	accept default
Admin passwd	--	password
System user	`root`	`root`
System group	`other`	`root`

4. Select the directory server settings shown in the following table during the installation.

Option	[Default Value]	Enter
Admin user ID	`admin`	accept default
Password	--	password
Directory manager DN	[`cn=Directory Manager`]	accept default
Directory manager password	--	password
Directory server root	[*/var/opt/mps/serverroot*]	accept default
Server identifier	[`jes-zone1`]	accept default
Server port	[`389`]	accept default
Suffix	[`dc=map,dc=beta,dc=com`]	accept default
Administration domain	[`map.beta.com`]	accept default
System user	[`root`]	`root`
System group	[`root`]	`root`
Directory to store config data	[Store Config Data on this server]	accept default
Directory to store user and group info	[Store user data and group data on this server]	accept default
Populate suffix	[Populate with Sample Data]	accept default.

5. Start the directory server as follows and ensure that it starts successfully.

```
jes-zone1# cd /var/opt/mps/serverroot/slapd-jes-zone1
jes-zone1# ./start-slapd
```

Deploying Access Manager in Zone-2

1. Log in to `zone-2` as `root` and start the Java ES installer as follows.

```
jes-zone2# installer
```

2. Select the following products to install:
   - Sun Java System Web Server 6.1 SP5 2005-Q4
   - Sun Java Access Manager 7 2005-Q4 and all supporting software
   - Communication Services Delegated Administrator 2005-Q4
   - Deselect Sun Java System Directory Server 5 2005-Q4 (Use the remote directory server installed on `jes-zone1`.)

3. Select common server settings. Refer to the steps described in “Deploying Directory Server in Zone-1” on page 158.

4. Select the Web server settings shown in the following table during the installation.

Option	[Default Value]	Enter
Admin user ID	`admin`	accept default
Admin passwd	--	password
Web server host	`jes-zone2.map.beta.com`	accept default
Admin port	`8888`	accept default
Admin runtime user ID	`root`	accept default
Web server runtime user ID	`webservd`	`root`
Web server runtime group ID	`webservd`	`root`
HTTP port	`80`	accept default
Document root directory	[*/opt/SUNWwbsvr/docs*]	accept default
Web server start on boot	[`yes`]	`no`

Note – Changing the Web server's runtime `userID` and `groupID` was necessary in order to run Access Manager using this same instance of the Web server.

5. Select the Access Manger settings shown in the following table during the installation.

Option	[Default Value]	Enter
Admin user ID	`amadmin`	accept default
Password	--	password
LDAP user ID	`amldapuser`	accept default
LDAP password	--	password1
Password encryption key	`87dfjkauefjkdafdadf`...	password
Install type	`Legacy Mode` (version 6.x style)	accept default
Access Manager web container options	--	Choose Sun Java System Web Server
Hostname	[`jes-zone2.map.beta.com`]	`jes-zone2.map.beta.com`
Web server port	[`80`]	accept default
Web server instance directory	[`/opt/SUNWwbsvr/https-jes-zone2.map.beta.com`]	`/opt/SUNWwbsvr/https-jes-zone2.map.beta.com`
Web server document directory	[`/opt/SUNWwbsvr/docs`]	`/opt/SUNWwbsvr/docs`
Is server instance port secure	[`No`]	accept default
Web container host	[`jes-zone2.map.beta.com`]	accept default
Services deployment URI	[`amserver`]	accept default
Common domain deployment URI	[`amcommon`]	accept default
Cookie domain	[`.beta.com`]	`.map.beta.com`
Administration console	`Deploy new console`	accept default
Console deployment URI	[`amconsole`]	accept default
Password deployment URI	[`ampassword`]	accept default
Console host name	[`jes-zone2.map.beta.com`]	`jes-zone2.map.beta.com`
Console port	[`80`]	accept default
Directory server host		`jes-zone1.map.beta.com`
Port		`389`
Access Manager directory root suffix	[`dc=map,dc=beta,dc=com`]	`dc=map,dc=beta,dc=com`
Directory manager	[`cn=Directory Manager`]	accept default
Password	--	password
Directory server provisioned with user data?	[`No`]	`No`

6. Start the Web server as follows and verify that it starts successfully.

```
jes-zone2# cd /opt/SUNWwbsvr/https-jes-zone2.map.beta.com
jes-zone2# ./start
```

Note – Verify that the Access Manager Web modules, including `amserver` and `amconsole`, are loaded during Web server startup.

Deploying Portal Server in Zone-3

1. Log in to `zone-3` as `root` and start the Java ES installer as follows:

```
jes-zone3# installer
```

2. Select the following products to install:
   - Sun Java System Web Server 6.1 SP5 2005-Q4
   - Sun Java System Portal Server 6 2005-Q4
   - Sun Java Access Manager 7 2005-Q4 (Select only the Access Manager SDK, deselect the other Access Manager components.)
   - Deselect Sun Java System Directory Server 5 2005-Q4 (Use the remote directory server installed on `jes-zone1`.)
3. Select common server settings. Refer to the steps described in “Deploying Directory Server in Zone-1” on page 158.
4. Select the Web server settings as shown in the following table during the installation.

Option	[Default Value]	Enter
Admin user ID	admin	accept default
Admin passwd	--	password
Web server host	jes-zone3.map.beta.com	accept default
Admin port	8888	accept default
Admin runtime user ID	root	accept default
Web server runtime user ID	webservd	root
Web server runtime group ID	webservd	root
HTTP port	80	accept default
Document root directory	[/opt/SUNWwbsvr/docs]	accept default
Web server start on boot	[yes]	no

5. Select the Access Manger settings as shown in the following table during the installation.

Option	[Default Value]	Enter
Admin user ID	`amadmin`	accept default
Password	--	password
LDAP user ID	`amldapuser`	accept default
LDAP password	--	password1
Password encryption key	`87dfjkauefjkdafdadf...`	password
Install type	`Legacy Mode` (version 6.x style)	accept default
Access Manager: Directory Server Settings		
Directory server host		`jes-zone1.map.beta.com`
Port		`389`
Access Manager directory root suffix	[`dc=map,dc=beta,dc=com`]	`dc=map,dc=beta,dc=com`
Directory manager	[`cn=Directory Manager`]	accept default
Password	--	password
Directory server provisioned with user data?	[`Yes`]	`Yes`
Access Manager: Web Container for Running Access Manager Services		
Host		`jes-zone2.map.beta.com`
Services deployment URI	[`amserver`]	accept default
Cookie domain	[`.beta.com`]	`.map.beta.com`
Services port	[`80`]	accept default
Server protocol	[`HTTP`]	accept default

6. Select the Portal Server: Web Container settings shown in the following table during the installation.

Option	[Default Value]	Enter
Web container	Sun Java System Web Server	Sun Java System Web Server
Installation directory	[`/opt/SUNWwbsvr`]	accept default
Server instance	[`jes-zone3.map.beta.com`]	accept default
Server instance port	[`80`]	accept default
Server document root	[`/opt/SUNWwbsvr/docs`]	accept default
Secure server instance port	[`No`]	accept default

Option	[Default Value]	Enter
Load balancer controlling multiple portal servers	[No]	accept default
Load balancer protocol		accept default
Load balancer host		accept default
Load balancer port		accept default
Deployment URI	[/portal]	accept default
Install sample portal	--	Install the sample portal

7. Start the Web server and verify that it starts successfully.

```
jes-zone3# cd /opt/SUNWwbsvr/https-jes-zone3.map.beta.com
jes-zone3# ./start
```

Note – Check that the PS Web modules, including portal, are loaded during the Web server startup.

Preparing Directory Server for Messaging and Calendar Server Installations

1. Log in to `zone1`, change to the directory where the `comm_dssetup` script is stored, and run the script as follows:

```
jes-zone1# cd /opt/SUNWcomds/sbin
jes-zone1# /var/opt/mps/serverroot/bin/slapd/admin/bin/perl comm_dssetup.pl
```

The Perl script prompts for a series of options. The following table shows how to respond to the prompts.

Option	[Default Value]	Enter
Directory server root	[/var/opt/mps/serverroot]	accept default
Directory server instance	slapd-jes-zone1	accept default
Directory manager DN	[cn=Directory Manager]	accept default
Directory manager password	--	password
Use directory server for users/groups	[Yes]	accept default.
Users/groups base suffix	[dc=map,dc=beta,dc=com]	accept default.
Schema type?	[2]	accept default
Update the schema files?	[yes]	accept default
Configure new indexes?	[yes]	accept default.
Reindex new indexes?	[yes]	accept default.

2. Confirm the choices and `comm_dssetup` proceeds. Continue with the next step after `comm_dssetup` completes.

Configuring Delegated Admin and Communications CLI for Creating Users

1. Log in to `zone-2` as root, change to the directory where the configuration script was installed, and execute the configuration script as shown below.

```
jes-zone2# cd /opt/SUNWcomm/sbin
jes-zone2# ./config-commda
```

The script prompts for a series of options. The following table shows how to respond to the prompts.

Option	[Default Value]	Enter
Directory to store user mgt data files	[/var/opt/SUNWcomm]	accept default
Install delegated admin utility, console and server		Select all
AM hostname	[jes-zone2.map.beta.com]	accept default
AM port	[8080]	80

Option	[Default Value]	Enter
Default domain	[map.beta.com]	map.beta.com
Default SSL port	[443]	accept default
Web container	[Web Server]	Web Server
Web server root directory	[/opt/SUNWwbsvr]	accept default
Web server instance identifier	[jes-zone2.map.beta.com]	accept default
Web virtual server identifier	[https-jes-zone2.map.beta.com]	accept default
Web server HTTP port	[80]	80
Default domain separator	[@]	accept default
Access Manager base directory	[/opt/SUNWam]	accept default
Web server root directory	[/opt/SUNWwbsvr]	accept default
Web server instance identifier	[jes-zone2.map.beta.com]	accept default
Web virtual server identifier	[https-jes-zone2.map.beta.com]	accept default
Web server HTTP port	[80]	80
URL of directory server	[ldap://jes-zone2.map.beta.com:389/]	ldap://jes-zone1.map.beta.com:389/
Bind as	[cn=Directory Manager]	accept default.
Password	--	password
AM top level admin	[amadmin]	accept default
AM admin passwd	--	password
Access Manager internal LDAP auth username	amldapuser	accept default
AM internal LDAP auth passwd for amldapuser	--	password1
Organization DN	[o=map.beta.com,dc=map,dc=beta,dc=com]	o=map.beta.com,dc=map,dc=beta,dc=com
Top level admin for default organization	[admin]	accept default
Password	--	password
Load sample service packages	--	Yes (Checked)
Load sample organizations	--	Yes (Checked)
Preferred mailhost for sample	[jes-zone2.map.beta.com]	jes-zone4.map.beta.com

2. Restart the Web server as follows.

```
jes-zone2# cd /opt/SUNWwbsvr/https-jes-zone2.map.beta.com
jes-zone2# ./stop
jes-zone2# ./start
```

3. Modify the domains and create sample users as shown in the following example.

```
jes-zone2# cd /opt/SUNWcomm/bin
jes-zone2# /opt/SUNWcomm/bin/commadmin domain modify -D admin -w password -X jes-
zone2.map.beta.com -n map.beta.com -p 80 -d map.beta.com -S mail,cal -H jes-
zone4.map.beta.com


jes-zone2# /opt/SUNWcomm/bin/commadmin user create -D admin -F John -l jdoe -L Doe -n
map.beta.com -p 80 -w password -W password -X jes-zone2.map.beta.com -S mail,cal -E
jdoe@map.beta.com -H jes-zone4.map.beta.com -k legacy


jes-zone2# /opt/SUNWcomm/bin/commadmin user create -D admin -F Calendar -l calmaster -
L Master -n map.beta.com -p 80 -w password -W password -X jes-zone2.map.beta.com -S
mail,cal -E calmaster@map.beta.com -H jes-zone4.map.beta.com -k legacy
```

Deploying the Messaging Server in Zone-4

1. Log in to `zone-4` as root.
2. Stop the `sendmail` daemon as follows:

```
jes-zone4# /etc/init.d/sendmail stop
```

3. Start the Java ES installer as follows:

```
jes-zone4# installer
```

4. Select the following products to install:
   - Sun Java System Messaging Server 6 2005-Q4
   - Sun Java Access Manager 7 2005-Q4 (Select only the Access Manager SDK, deselect the other Access Manager components.)
   - Deselect Sun Java System Directory Server 5 2005-Q4 (Use the remote directory server installed on `jes-zone1`.)

5. Select Common Server Settings. Refer to the steps described in “Deploying Directory Server in Zone-1” on page 158.

6. Select the administration server and Access Manager settings.

Option	[Default Value]	Enter
Server Root	[/var/opt/mps/serverroot]	accept default
Admin Port	390	accept default
Admin Domain	map.beta.com	accept default
System User	[root]	root
System Group	[root]	root
Administration Server: Configuration Directory Server Settings		
Admin User ID	admin	accept default
Admin Password	--	password
Directory Server Host	[jes-zone4.map.beta.com]	jes-zone1.map.beta.com
Directory Server Port	389	accept default
Access Manager: Administration Settings		
Admin User ID	amadmin	accept default
Password	--	password
LDAP User ID	amldapuser	accept default
LDAP Password	--	password1
Password Encryption Key	87dfjkauefjkdafdadf...	password
Install type	Legacy Mode (version 6.x style)	accept default
Access Manager: Directory Server Settings		
Directory Server Host		jes-zone1.map.beta.com
Port		389
Access Manager directory root suffix	[dc=map,dc=beta,dc=com]	dc=map,dc=beta,dc=com
Directory Manager DN	[cn=Directory Manager]	accept default
Password	--	password
Directory server provisioned with user data?	[Yes]	Yes
Access Manager: Web Container for Running Access Manager Services		
Host		jes-zone2.map.beta.com
Services deployment URI	[amserver]	accept default

Option	[Default Value]	Enter
Cookie domain	[.beta.com]	.map.beta.com
Services port	[80]	accept default
Server protocol	[HTTP]	accept default

7. Configure the messaging server as follows:

```
jes-zone4# cd /opt/SUNWmsgsr/sbin
jes-zone4# ./configure
```

8. Specify the fully qualified host name of the messaging server, FQHN: [jes-zone4.map.beta.com].
9. Define the directory to store configuration and data files [/var/opt/SUNWmsgsr].
10. Install MTA, MS store, and Messenger Express. There is no need to install multiplexor.
11. Provide the name of the mail server user and group: UNIX username [mailsrv], UNIX group [mail].

    The installation script prompts for a series of options. The following table shows how to respond to the configuration options.

Option	[Default Value]	Enter
URL of directory server	[ldap://jes-zone1.map.beta.com:389]	accept default.
Bind as	[cn=Directory Manager]	accept default.
Password	--	password
User/group server LDAP	[ldap://jes-zone1.map.beta.com:389]	accept default.
Bind as	[cn=Directory Manager]	accept default.
Password	--	password
Postmaster email address	--	foo@jes-zone4.map.beta.com
Password for messaging server accounts	--	password
Default email domain	[map.beta.com]	map.beta.com
Organization DN	[o=map.beta.com,dc=map,dc=beta,dc=com]	o=map.beta.com, dc=map,dc=beta,dc=com

12. Start the messaging server as follows:

```
jes-zone4# /opt/SUNWmsgsr/sbin/start-msg
```

Deploying Calendar Server in Zone-5

1. Log in to `zone-5` as `root` and start the Java ES installer as follows:

```
jes-zone5# installer
```

2. Start the Java ES installer and select the following products to install:
   - Sun Java System Calendar Server 6 2005-Q4
   - Sun Java Access Manager 7 2005-Q4 (Select only the Access Manager SDK, deselect the other Access Manager components.)
   - Deselect Sun Java System Directory Server 5 2005-Q4 (Use the remote directory server installed on `jes-zone1`).

3. Select the common server settings. Refer to the steps described in "Deploying Directory Server in Zone-1" on page 158.

4. Select the administration server and Access Manager settings shown below.

Option	[Default Value]	Enter
Access Manager: Administration Settings		
Admin user ID	`amadmin`	accept default
Password	--	password
LDAP user ID	`amldapuser`	accept default
LDAP password	--	password1
Password encryption key	`87dfjkauefjkdafdadf...`	password
Install type	`Legacy Mode` (version 6.x style)	accept default
Access Manager: Directory Server Settings		
Directory server host		`jes-zone1.map.beta.com`
Port		`389`
Access Manager directory root suffix	[`dc=map,dc=beta,dc=com`]	`dc=map,dc=beta,dc=com`
Directory manager DN	[`cn=Directory Manager`]	accept default
Password	--	password
Directory server provisioned with user data?	[`Yes`]	`Yes`
Access Manager: Web Container for Running Access Manager Services		
Host		`jes-zone2.map.beta.com`
Services deployment URI	[`amserver`]	accept default

Option	[Default Value]	Enter
Cookie domain	[.beta.com]	.map.beta.com
Services port	[80]	accept default
Server protocol	[HTTP]	accept default

5. Configure the calendar server as follows:

```
jes-zone5# cd /opt/SUNWics5/cal/sbin
jes-zone5# ./csconfigurator.sh
```

   - Enter the LDAP server host name jes-zone1.map.beta.com.
   - Enter the LDAP server port as 389.
   - Enter the Directory Manager DN as cn=Directory Manager and the password as password.
   - The Base DN should be o=map.beta.com,dc=map,dc=beta,dc=com. This may require editing.
   - Enter the Calendar Server Administrator User ID as calmaster, password as password.
   - Enable email alarms.
   - Set the administrator email address to root@jes-zone4.map.beta.com.
   - Set the SMTP Host Name to jes-zone4.map.beta.com.
   - Choose the runtime configuration. Choose the default values of Service Port, Runtime User ID, and Runtime Group ID. Change Max Sessions to 10000, Max Threads to 64, and Max Processes to 4.
   - Choose the default values for the rest of the settings.

6. Start the calendar server as follows:

```
jes-zone5# /opt/SUNWics5/cal/sbin/start-cal
```

Installing Communications Express and Messenger Express in Zone-6

1. Log in to zone-6 as root and start the Java ES installer as follows:

```
jes-zone6# installer
```

2. Select the following products to install:
   - Sun Java System Web Server 6.1 SP5 2005-Q4
   - Sun Java System Communications Express 6 2005-Q4

- Sun Java System Messaging Server 6 2005-Q4
- Sun Java Access Manager 7 2005-Q4 (Select only the Access Manager SDK, deselect the other Access Manager components.)
- Deselect Sun Java System Directory Server 5 2005-Q4 (Use the remote directory server installed on `jes-zone1`.)

3. Select the common server settings. Refer to the steps described in "Deploying Directory Server in Zone-1" on page 158.

4. Select the administration server settings shown in the following table.

Option	[Default Value]	Enter
Server Root	[/var/opt/mps/serverroot]	accept default
Admin Port	390	accept default
Admin Domain	map.beta.com	accept default
System User	[root]	root
System Group	[root]	root
Administration Server: Configuration Directory Server Settings		
Admin User ID	admin	accept default
Admin Password	--	password
Directory Server Host	[jes-zone6.map.beta.com]	jes-zone1.map.beta.com
Directory Server Port	389	accept default

5. Select the Web server settings shown in the following table.

Option	[Default Value]	Enter:
Admin user ID	admin	accept default
Admin passwd	--	password
Web server host	jes-zone6.map.beta.com	accept default
Admin port	8888	accept default
Admin runtime user ID	root	accept default
Web server runtime user ID	webservd	root
Web server runtime group ID	webservd	root
HTTP Port	80	accept default
Document root directory	[/opt/SUNWwbsvr/docs]	accept default
Web server start on boot	[yes]	no

6. Select the administration server and Access Manager settings as shown below.

Option	[Default Value]	Enter
Access Manager: Administration Settings		
Admin user ID	`amadmin`	accept default
Password	--	password
LDAP user ID	`amldapuser`	accept default
LDAP password	--	password1
Password encryption key	`87dfjkauefjkdafdadf...`	password
Install type	`Legacy Mode` (version 6.x style)	accept default
Access Manager: Directory Server Settings		
Directory server host		`jes-zone1.map.beta.com`
Port		`389`
Access Manager directory root suffix	[`dc=map,dc=beta,dc=com`]	`dc=map,dc=beta,dc=com`
Directory manager DN	[`cn=Directory Manager`]	accept default
Password	--	password
Directory server provisioned with user data?	[`Yes`]	`Yes`
Access Manager: Web Container for Running Access Manager Services		
Host		`jes-zone2.map.beta.com`
Services deployment URI	[`amserver`]	accept default
Cookie domain	[`.beta.com`]	`.map.beta.com`
Services port	[`80`]	accept default
Server protocol	[`HTTP`]	accept default

Deploying Messenger Express in Zone-6

Communications Express requires that Messenger Express multiplexor (MEM) component be configured on the same zone. Perform the following tasks to configure and deploy MEM on `zone-6`.

1. Stop the sendmail daemon as follows.

```
jes-zone6# /etc/init.d/sendmail stop
```

2. Configure MEM as follows:

```
jes-zone6# cd /opt/SUNWmsgsr/sbin
jes-zone6# ./configure
```

3. Specify the fully qualified host name of the messaging server, FQHN: [jes-zone6.map.beta.com]
   - Define the directory to store configuration and data files [/var/opt/SUNWmsgsr].
   - Select only Messenger Express. There is no need to install MTA, MS store, and multiplexor.
   - Provide the name of the mail server user and group: UNIX username [mailsrv], UNIX group [mail].

   The installation script prompts for a series of options. The following table shows how to respond to the configuration options:

Option	[Default Value]	Enter
URL of directory server	[ldap://jes-zone1.map.beta.com:389]	accept default
Bind as	[cn=Directory Manager]	accept default
Password	--	password
User/group server LDAP	[ldap://jes-zone1.map.beta.com:389]	accept default
Bind as	[cn=Directory Manager]	accept default
Password	--	password
Postmaster email address	--	foo@jes-zone4.map.beta.com
Password for messaging server accounts	--	password
Default email domain	[map.beta.com]	map.beta.com
Organization DN	[o=map.beta.com,dc=map,dc=beta,dc=com]	o=map.beta.com, dc=map,dc=beta,dc=com

4. Configure the MEM in proxy mode and start the MEM. Note that the MEM will be running on port 2080, so UWC can use port 80.

```
jes-zone6# cd /opt/SUNWmsgsr/sbin
jes-zone6# ./configutil -o local.service.http.proxy -v 1
jes-zone6# ./configutil -o local.service.http.proxy.admin -v admin
jes-zone6# ./configutil -o local.service.http.proxy.admin.jes-zone4 -v admin
jes-zone6# ./configutil -o local.service.http.proxy.adminpass -v password
jes-zone6# ./configutil -o local.service.http.proxy.adminpass.jes-zone4 -v password
jes-zone6# ./configutil -o local.service.http.proxy.port.jes-zone4.map.beta.com -v 80
jes-zone6# ./configutil -o service.http.allowadminproxy -v no
jes-zone6# ./configutil -o service.http.ipsecurity -v no
jes-zone6# ./configutil -o service.http.port -v 2080
jes-zone6# /opt/SUNWmsgsr/sbin/start-msg
```

5. Configure the back-end messaging server to allow proxy logins. To do so, log in to zone-4, where the back-end messaging server is deployed. Execute the following commands, and restart the back-end messaging server.

```
jes-zone4# cd /opt/SUNWmsgsr/sbin
jes-zone4# ./configutil -o service.http.allowadminproxy -v 1
jes-zone4# /opt/SUNWmsgsr/sbin/stop-msg
jes-zone4# /opt/SUNWmsgsr/sbin/start-msg
```

Deploying Communications Express in Zone-6

After the Messenger Express multiplexor (MEM) is configured, follow the steps in this section to configure and deploy UWC on zone-6.

1. Start the Web server container as follows.

```
jes-zone6# cd /opt/SUNWwbsvr/https-jes-zone6.map.beta.com
jes-zone6# ./start
```

2. Configure the Access Manager SDK as outlined in the following steps.

- Change to the directory that contains the amconfig input file template, `amsamplesilent`.. Copy the input template file to a new file.

```
jes-zone6# cd /opt/SUNWam/bin
jes-zone6# cp amsamplesilent amsamplesilent.uwc
```

- Edit the `amsamplesilent.uwc` file to set the Access Manager SDK configuration parameters as shown below.

```
DEPLOY_LEVEL=4
SERVER_NAME=jes-zone2
SERVER_HOST=jes-zone2.map.beta.com
SERVER_PORT=80
ADMIN_PORT=8888
DS_HOST=jes-zone1.map.beta.com
DS_DIRMGRPASSWD=password
ROOT_SUFFIX="dc=map,dc=beta,dc=com"
ADMINPASSWD=password
AMLDAPUSERPASSWD=password1
COOKIE_DOMAIN=.map.beta.com
AM_ENC_PWD="password"
NEW_OWNER=root
NEW_GROUP=root
PAM_SERVICE_NAME=other
WEB_CONTAINER=WS6
DIRECTORY_MODE=4
AM_REALM=disabled
WS61_INSTANCE=https-jes-zone6.map.beta.com
WS61_HOST=jes-zone6
```

- Run the `amconfig` command using the edited file.

```
jes-zone6# ./amconfig -s amsamplesilent.uwc
```

- Restart the Web server.

```
jes-zone6# cd /opt/SUNWwbsvr/https-jes-zone6.map.beta.com
jes-zone6# ./stop
jes-zone6# ./start
```

3. Configure Communications Express.

```
jes-zone6# cd /opt/SUNWuwc/sbin
jes-zone6# ./config-uwc
```

The installation script prompts for a series of options. The following table shows how to respond to the configuration options.

Option	[Default Value]	Enter
Directory to store config and data files	[/var/opt/SUNWuwc]	accept default
Install mail and calendar components		select both
Hostname	[jes-zone6]	accept default
DNS domain	[map.beta.com]	accept default
Web container	[Web Server]	accept default
Web server root directory	[/opt/SUNWwbsvr]	accept default
Web server instance identifier	[jes-zone6.map.beta.com]	accept default
Virtual server identifier	[https-jes-zone6.map.beta.com]	accept default
HTTP port	[80]	accept default
Web container user ID	[webservd]	root
Web container group IP	[webservd]	root
URI path	[/uwc]	accept default
Hosted domain support	[No]	accept default
URL of directory server	[ldap://jes-zone6.map.beta.com:389/]	[ldap://jes-zone1.map.beta.com:389/]
Bind DN	[cn=Directory Manager]	accept default.
Password	--	password
DC tree suffix	[dc=map,dc=betat,dc=com]	accept default.
Default domain	[map.beta.com]	accept default

Option	[Default Value]	Enter
IS login URL	[http://jes-zone6.map.beta.com:80/amserver/UI/Login]	[http://jes-zone2.map.beta.com:80/amserver/UI/Login]
IS administrator DN	--	uid=amadmin,ou=people,dc=map,dc=beta,dc=com
IS administrator password	--	password
Messenger Express port	[80]	2080
Calendar server hostname	[jes-zone6.map.beta.com]	jes-zone5.map.beta.com
Calendar server port	[9004]	80
Calendar admin user ID	[calmaster]	accept default
Calendar administrator user password	--	password
URL of PAB directory derver	[ldap://jes-zone1.map.beta.com:389]	accept default
Bind as	[cn=Directory Manager]	accept default
Password	--	password

4. Restart the Web server container as follows.

```
jes-zone6# cd /opt/SUNWwbsvr/https-jes-zone6.map.beta.com
jes-zone6# ./stop
jes-zone6# ./start
```

Configuring Single Sign-On for Communications Services Products

Setting up single sign-on for Communications Express, Access Manager, the messaging server, and the calendar server, takes the following steps.

1. Specify the Communications Express settings on `zone-6` by verifying the following settings in the */var/opt/SUNWuwc/WEB-INF/config/uwcauth.properties* file.
   - Set `uwcauth.identity.enabled` to `true`
   - Set `uwcauth.identity.login.url` to `http://jes-zone2.map.beta.com:80/amserver/UI/Login`
   - Set `uwcauth.identity.cookiename` to `iPlanetDirectoryPro`
   - Set `uwcauth.identity.binddn` to `uid=amadmin,ou=people,dc=map,dc=beta,dc=com`

- Set uwcauth.identity.bindcred to password
- Set uwcauth.http.port to 80
- Set uwcauth.https.port to 443

Note – Restart the Web server if any of the preceding settings are changed.

2. Specify the messaging server settings on zone-4 and zone-6.

   To enable Communications Express users to access Messenger Express using the Access Manager session, run the configutil commands as shown in the following example on zone-4 (where the back-end messaging server is deployed) and zone-6 (where MEM is deployed), and restart the servers.

```
# cd /opt/SUNWmsgsr/sbin
# ./configutil -o local.webmail.sso.amnamingurl -v http://jes-zone2.map.beta.com:80/
amserver/namingservice
# ./configutil -o local.webmail.sso.uwcenabled -v 1
# ./configutil -o local.webmail.sso.uwclogouturl -v http://jes-zone6.map.beta.com:80/uwc/
base/UWCMain\?op=logout
# ./configutil -o local.webmail.sso.uwcport -v 80
# ./configutil -o local.webmail.sso.uwccontexturi -v "uwc"
# ./configutil -o local.webmail.sso.amcookiename -v iPlanetDirectoryPro
# ./configutil -o local.webmail.sso.uwchome -v http://jes-zone6.map.beta.com/uwc
# ./configutil -o local.webmail.sso.enable -v 0
```

3. Specify the calendar server settings on zone-5. To set up single sign-on for the calendar server, follow these steps:
   - Stop the calendar server on zone-5.
   - Open the */opt/SUNWics5/cal/config/ics.conf* file in an editor, such as vi.
   - Set service.http.allowadminproxy to yes.
   - Set local.calendar.sso.amnamingurl to http://jes-zone2.map.beta.com:80/amserver/namingservice.
   - Set local.calendar.sso.singlesignoff to yes.
   - Set local.calendar.sso.amcookiename to iPlanetDirectoryPro.
   - Set local.calendar.sso.logname to am_sso.log.
   - Set service.calendarsearch.ldap to no.
   - Set service.http.ipsecurity to no.
   - Start the calendar server.

Tuning Sun Java Enterprise System Software for Improved Performance

The default out-of-box Java ES configuration does not effectively utilize the compute and memory resources of the Sun Fire T2000 platform. The following sections focus on some minimal tunings that enable performance on the Sun Fire T2000 platform to be improved. Note that this is by no means an exhaustive list. For the most optimal settings, refer to the individual product reference and performance guides.

Tuning the Directory Server

1. Increase the directory server database cache size.

   Each directory server uses a database cache that holds pages from the database containing indexes and entries. The database cache size (`nsslapd-dbcachesize`) is specified in bytes, and the cache space is allocated at server startup.

2. Increase the directory server entry cache size.

   The entry cache holds recently accessed entries, formatted for delivery to client applications. The entry cache size for a suffix (`nsslapd-cachememsize`) is specified in bytes, and the entry cache is allocated as needed.

   The following example illustrates the preceding recommendations. The values that are edited are highlighted. Be sure to shut down the directory server before making these changes.

   In the `dse.ldif` file located in the `config` directory of the directory server:

```
dn: cn=config,cn=ldbm database,cn=plugins,cn=config
nsslapd-dbcachesize: 54580838


n: cn=userRoot,cn=ldbm database,cn=plugins,cn=config
nsslapd-cachememsize: 126292787
```

3. Isolate databases and logs, including the transaction log and access log, on different disks.

Tuning a Web Container

Tune the Sun™ Open Net Environment (Sun™ ONE) Web Server (Web container) in all the zones, including `zone-2` (where Access Manager is deployed), `zone-3` (where the portal server is deployed), and `zone-6` (where Communications Express is deployed). Edit the *magnus.conf* and *server.xml* files in the Web server `config` directory using the following settings.

1. Increase the Sun ONE Web Server `ListenQ` size.

   This parameter and the related Solaris `tcp_conn_req_max_q` and `tcp_conn_req_max_q0` settings should match the throughput of the Sun One Web Server HTTP server. These queues act as a buffer to manage the irregular rate of connections coming from web users.

2. Improve the server thread concurrency.

   Increasing the number of active HTTP threads that handle the incoming HTTP requests can increase the concurrency and thereby improve the performance of the web server. The `'RQThrottle'` setting in the *magnus.conf* file specifies the maximum number of request processing threads in the Web server.

3. Increase the number of acceptor threads and connection queue size.

   Acceptor threads are threads that wait for connections. These threads accept connections and put them in a connection queue where they are then picked up by request processing threads.

4. Tune the Java VM.

   Increase the Java VM heap size from the default 256 MB to make best use of the memory resources available on the Sun Fire T2000 platform. Also, apply the GC (garbage collection) tunings.

The following examples summarize these tuning changes. The values that are added or changed are highlighted.

In the *magnus.conf* file:

```
RqThrottle 512
RqThrottleMin 128
ThreadIncrement 64
ConnQueueSize 8192
ListenQ 8192
```

In the *server.xml* file:

```
<SERVER qosactive=”false”>
<LS id="ls1" port="80" servername=”jes-zone2.sfbay.sun.com”
defaultvs="https-jes-zone2.sfbay.sun.com" ip="any" security="off"
blocking="no" acceptorthreads="4"
</SERVER>
 <JAVA javahome="....>
      < JVMOPTIONS>-Xms3136M -Xmx3136M</JVMOPTIONS>
      <JVMOPTIONS>-server</JVMOPTIONS>
      <JVMOPTIONS>-XX:+DisableExplicitGC</JVMOPTIONS>
      <JVMOPTIONS>-XX:+UseMPSS</JVMOPTIONS>
      <JVMOPTIONS>-XX:+UseParallelOldGC</JVMOPTIONS>
      <JVMOPTIONS>-XX:+UseParallelGC</JVMOPTIONS>
      <JVMOPTIONS>-XX:ParallelGCThreads=8</JVMOPTIONS>
      <JVMOPTIONS>-XX:+PrintGCTimeStamps</JVMOPTIONS>
      <JVMOPTIONS>-XX:+PrintGCDetails</JVMOPTIONS>
</JAVA>
```

Note – Restart the Web server after applying the preceding recommendations.

Tuning Access Manager

1. Apply all the tuning recommendations in “Tuning a Web Container” on page 181.
2. Apply patch 120954-02 as follows to fix bugs and improve the performance of the Access Manager server.

```
jes-zone2# patchadd -G 120954-02
```

Note – After adding the patch, do not forget to follow the post-patch instructions, which primarily involve running the `amconfig` command.

3. Increase the sizes of the notification thread pool and the task queue length.

The parameter `threadpool.size` specifies the size of the notification thread pool (total number of threads), and the `threadpool.threshold` parameter specifies the maximum task queue length. If the task queue reaches the maximum length, further incoming requests are rejected along with a `ThreadPoolException` until the queue has vacancy. These errors are logged in the `amSession` file in the Access Manager debug (*/var/opt/SUNWam/debug*) directory.

4. Consider increasing the number of allowed sessions and the size of the SDK cache.

   Monitor the stats directory (*/var/opt/SUNWam/stats*) to find information on the `Max sessions in session table` and SDK cache hits. Consider increasing the values of both parameters.

5. Increase the size of the LDAP connection pool.

The following examples illustrate these tuning changes. The values that are added or changed are highlighted.

Make the following changed in the *AMConfig.properties* file located in the */etc/opt/SUNWam/config* directory.

```
com.iplanet.am.notification.threadpool.size=32
com.iplanet.am.notification.threadpool.threshold=50000
com.iplanet.am.sdk.cache.maxSize=100000
com.iplanet.am.session.maxSessions=25000
```

Specify the following values in the *serverconfig.xml* file located in the */etc/opt/SUNWam/config* directory.

```
<ServerGroup name="default" minConnPool="130" maxConnPool="130">
```

In the Access Manager console, perform the following tasks:

1. Log in to the Access Manager console as `amadmin`.
2. Select the `Service Configuration` tab.
3. Click `Core` under `Authentication Modules`.
4. Edit the Default LDAP Connection Pool Size to be `130:130`.
5. Click `Save`.

Note – Restart the Web server after applying the above recommendations.

Tuning the Portal Server

1. Apply all the tuning recommendations in “Tuning a Web Container” on page 181.
2. Apply the Access Manager patch 120954-02 as follows:

```
jes-zone3# patchadd -G 120954-02
```

3. Tune the AM SDK. Edit the *AMConfig.properties* and *serverconfig.xml* files as described in “Tuning Access Manager”. The files can be found in the */etc/opt/SUNWam/config* directory.
4. Tune the caller pool and template scan interval. The default caller pool settings force the portal server to create a new thread for every channel rendered rather than using a thread pool. Also consider adjusting the template scan interval time. Edit the following properties in the file */etc/opt/SUNWps/desktop/desktopconfig.properties* file.

```
callerPoolMinSize=128
callerPoolMaxSize=256
callerPoolPartitionSize=32
templateScanInterval=3600
```

5. Tune the portal channels and containers by removing the channels and containers that are not needed for better performance.

   Perform the following tasks on the Access Manager console:

   - Log in to the Access Manager console as `amadmin`.
   - Select the `Identity Management` tab.
   - Select `View->Services`.
   - Click `Portal Desktop` (under `Portal Server Config`).
   - Click `Manage Channels and Containers`.
   - Click `JSPTabContainer` (the master container) to view all the containers that are visible on the portal desktop. Remove the unused containers. This deployment used only `MyFrontPagetabPanelContainer`. If containers are removed, click `Save` under `Channel Management`.
   - Click `MyFrontPageTabPanelContainer` to view all the channels visible on the portal desktop. Remove any unneeded channels, such as `SampleXML`. This deployment used five channels including `UserInfo`, `App`, `BookMark`, `BookMark2`, and `SampleJSP`. If any channels are removed, click `Save` under `Channel Management`.

Note – Restart the Web server after applying any of the preceding recommendations.

Tuning the Messaging Server

1. Increase the default number of the IMAP, HTTP, and POP processes as shown in the following example:

```
jes-zone4# cd /opt/SUNWmsgsr/sbin
jes-zone4# ./configutil -o service.pop.numprocesses -v 8
jes-zone4# ./configutil -o service.imap.numprocesses -v 8
jes-zone4# ./configutil -o service.http.numprocesses -v 8
```

2. Increase the number of dispatcher processes and the size of the job queue as shown in the following examples. The values that are to be edited are highlighted.

   Specify the following values in the *job_controller.cnf* file located in the *config* directory of the messaging server.

```
[ POOL=DEFAULT]
job_limit=10
```

   Specify the following values in the *dispatcher.cnf* file located in the *config* directory of the messaging server.

```
MIN_PROCS=8
MAX_PROCS=32
```

Note – Restart the messaging server once these changes are complete.

3. Use RAID technology for the Message Store.

   If the message store requires multiple disks, use redundant array of independent disks (RAID) technology to simplify the management of multiple disks. With RAID technology, data can be spread across a series of disks, while the disks appear as one logical disk volume to simpligy disk management. This deployment used a Sun StorageTek™ 3510 FC Array with twelve 36 GB 15K rpm disk drives, and the Solaris Volume Manager software to create the RAID-0 volume.

4. Apply Access Manager patch 120954-02 as follows.

```
jes-zone4# patchadd -G 120954-02
```

Tuning the Calendar Server

1. Tune the number of calendar server processes and increase the maximum calendar sessions. Restart the calendar server after the changes are complete.

   Specify the following values in the *ics.conf* file located in the *config* directory of the calendar server.

```
service.http.maxsessions = "10000"
service.http.numprocesses = "6"
service.http.maxthreads = "64"
```

2. Apply Access Manager patch 120954-02.

```
jes-zone5# patchadd -G 120954-02
```

Tuning Communications Express

1. Apply all the tuning recommendations in "Tuning a Web Container" on page 181.
2. Tune the AM SDK. Edit the *AMConfig.properties* and *serverconfig.xml* files as described in "Tuning Access Manager" on page 182. The files can be found in the */etc/opt/SUNWam/config* directory.
3. Restart the Web server after the changes are complete.
4. Apply Access Manager patch 120954-02 as follows.

```
jes-zone6# patchadd -G 120954-02
```

Tuning the Solaris Operating System

1. Increase the file descriptor limits. Increase the values of `rlim_fd_max` and `rlim_fd_cur` in the */etc/system* file to increase the number of file descriptors for all Java ES components.

```
set rlim_fd_max=65536
set rlim_fd_cur=65536
```

Note – Reboot the system after editing the */etc/system* file.

2. Increase the settings for Solaris OS TCP/IP listen queues. The `tcp_conn_req_max_q` queue determines the number of completed connections waiting to return from an `accept()` call, and the `tcp_conn_req_max_q0` queue determines the maximum number of connections with the handshake incomplete.

```
/usr/sbin/ndd -set  /dev/tcp  tcp_conn_req_max_q  8192
/usr/sbin/ndd -set  /dev/tcp  tcp_conn_req_max_q0  8192
```

Note – To automatically have these `ndd` commands executed after each system reboot, place them in a file, such as */etc/rc2.d/network-tuning*.

Sun Java Enterprise System Performance Test Case

The performance of the Java Enterprise System with the suggested tuning parameters was tested on a test Sun Fire T2000 platform. The JESMark benchmark was used as the workload. JESMark is a Sun internal benchmark designed to test the performance of the Java Enterprise System as a whole. The Java Enterprise System offers virtually endless deployment scenarios. It is impractical to test all the facets of the Java ES software suite with a single benchmark. JESMark attempts to stress commonly used features and integration scenarios which cover the majority of customer deployments.

The core components of the Java ES tested include:

- Sun Java System Directory Server
- Sun Java System Access Manager
- Sun Java System Portal Server
- Sun Java System Messaging Server
- Sun Java System Calendar Server
- Sun Java Communications Express

Overview of the JESMark Benchmark

The JESMark benchmark models an employee portal of a large corporation. The portal serves as a launching pad for all employee services, such as email and calendar services. Once launched, the e-mail and calendar services run as separate applications in a browser window. Accordingly, the JESMark benchmark comprises different sub-benchmarks that test mail, calendar, and portal services. The following sections briefly describe the various sub-benchmarks.

Portal Workload

The portal is the central point of access. Employees log in to the portal to find information and use links in the portal to access various other services. The authentication is then propagated to the other services invoked or launched by the portal. The portal sub-benchmark attempts to test the page-rendering performance and resource handling of the portal server. The sub-benchmark also tests how well several standard portal channels are integrated with other services.

The main Java ES components that are stressed include the portal server, Access Manager, and the directory server. The interactions between the portal server and messaging and calendar servers are also tested.

Calendar Workload

Employees have a calendar channel on their portal, which allows them to see new calendar events. However, employees still use either a separate Web browser window (Communications Express client) or a separate calendar client (Outlook Express) to manage calendars.

This sub-benchmark emulates both Outlook Express clients and Communications Express clients that access the calendar server. The transactions include common operations such as creating, reading and deleting calendar events, and accessing the month/day/week view of all calendar events.

This sub-benchmark primarily stresses the functionalities of the calendar server, Communications Express, and their interactions with back-end servers, Access Manager, and the directory server.

Email Workload

Employees commonly have an email channel on their portal, which allows them to see new messages and their headers. However, employees still use their favorite mail client (using protocols such as POP or IMAP) or a separate Web browser window (HTTP) to manage email.

This sub- benchmark emulates all email clients, including POP3, IMAP, and Webmail clients. The transactions include common operations such as reading, deleting, saving, and sending email. It primarily stresses the functionalities of the messaging server, Communications Express, and their interactions with back-end servers, Access Manager, and the directory server.

Logical Architecture

Figure 11-2 shows the various client drivers that comprise the JESMark and Java Enterprise System components that they stress. It also shows the interactions among the Java ES components. Arrows indicate the flow of requests.

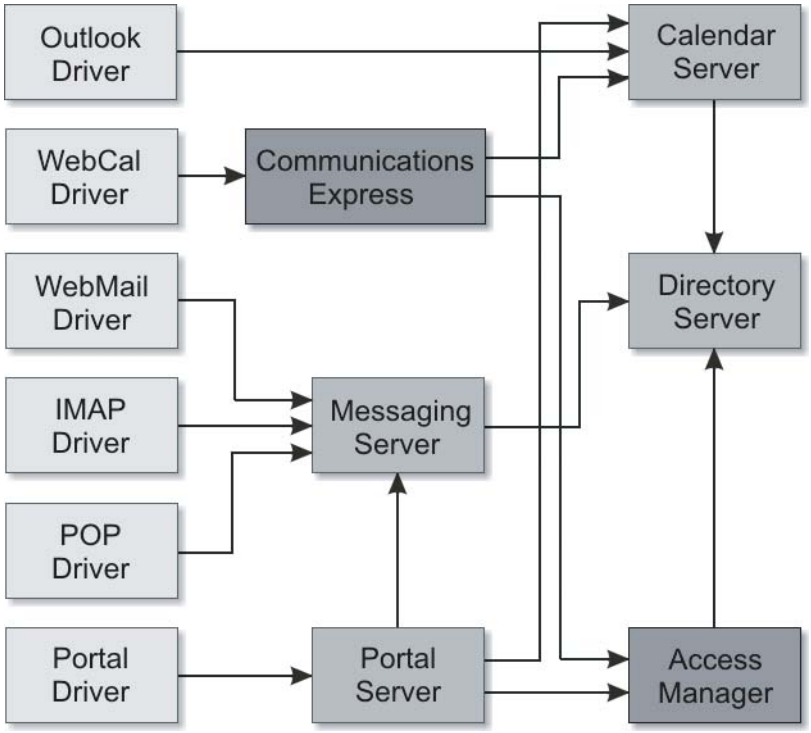

FIGURE 11-2 Client drivers comprising JESMark and Java ES

Testing Scenario

The Sun Fire T2000 server used in the performance tests featured an eight-core UltraSPARC T1 processor, 32 GB of main memory, four 1000 Base-T onboard network interfaces, one Sun StorageTek 3510 FC array with twelve 36 GB 15K rpm disk drives, and one 73 GB SAS disk drive. This system was installed with the Solaris 10 OS (1/06) .

Java ES deployment scenarios were tested with and without using Solaris Zones software. In the first test scenario, all components of the Java Enterprise System 2005-Q4 were deployed in the default global zone. In the second test scenario, the system was configured with six local Solaris Zones using Solaris Containers technology. Each of the six zones hosted a component of the Java Enterprise System 2005-Q4. The Java ES components are assigned to zones as follows:

- Zone1: Directory server
- Zone2: Access Manager (Identity Server)
- Zone3: Portal server
- Zone4: Messaging server (including MTA)
- Zone5: Calendar server
- Zone6: Communications Express and Messenger Express Multiplexor

The test setup described in this chapter used eight Sun Fire 280R servers, configured with two UltraSPARC-III+ processors and 8 GB of memory, as the client systems for running all client drivers.

Performance Results

The following table shows the results of the JESMark tests evaluating the performance of Java Enterprise System on a Sun Fire T2000 server.

System	Users	JESMark op/s	CPU Utilization	Free Memory
Sun Fire T2000	1000	71	20.00%	25 GB
Sun Fire T2000 /Solaris Zones	2500	208	85.00%	20 GB

In the first test configuration, the Java ES deployment could not make use of the vast compute and memory resources offered by the Sun Fire T2000 platform. The system could only sustain the load of 1000 users despite the fact that 80 percent of CPU resources were available. This performance limitation was a result of scalability issues with the Web container. There was severe contention from the portal server, Access Manager, and Communications Express applications, all of which competed for the heap and memory resources of a single Web container instance.

In the second test configuration, there was no such contention because each of the Java ES components was deployed in different Web container instances, each of which was hosted on a different Solaris Zone. This deployment makes good use of the memory and compute

resources offered by the Sun Fire T2000 platform. As shown in the preceding table, the Java ES deployment that uses Solaris Container technology is capable of supporting nearly three times the number of users supported by the first configuration, which did not use Solaris Containers.

12

About the Authors

Harry Foxwell is a Senior Technical Specialist for the Government division of Sun Microsystems in the Washington, D.C. area, where he is responsible for solutions consulting and customer education on the Solaris OS, Linux operating system, and grid technologies. Prior to joining Sun in 1995, Harry worked as a UNIX and Internet specialist for Digital Equipment Corporation, and has worked with UNIX systems since 1979. He also maintains one of Sun's internal web sites devoted to Linux technical information, and has been influential in developing and promoting Sun's Linux operating system and x86 hardware strategy. Harry received his doctorate in Information Technology from George Mason University in May, 2003, and has since taught graduate level courses at George Mason University in operating systems and electronic commerce.

Menno Lageman is consultant in Sun's Professional Services organization in the Netherlands, where he assists Sun customers in implementing and optimizing their systems. His specialities include the Solaris OS, high end servers, and resource mangement. Prior to joining Sun in 1999, Menno held several technical positions in mainframe and UNIX environments.

Isaac Rozenfeld is an IT Architect in Sun's Advanced Datacenter Practice group, focusing on the adoption of the Solaris 10 OS. Based in the New York area, Isaac participates in advisory, architecture and delivery of technical solutions at several financial services organizations. Since joining Sun in 1998, Isaac has focused on platform and resource management in the Solaris OS. Isaac received a Bachelor of Science degree in computer science from Queens College, City University of New York. Isaac can be reached at *isaac@sun.com*. He also maintains a blog at `http://blogs.sun.com/unixman`.

Sreekanth Setty is a member of the Performance, Availability, and Architecture Engineering (PAE) group at Sun Microsystems. During his 10 years at Sun, Sreekanth has focused on performance analysis, tuning, prototyping, benchmarking, and sizing in various tiers of enterprise infrastructure, with an emphasis on web (SSL) servers, middleware servers, database servers, and messaging servers. Sreekanth holds M.S in Computer Science from the University of Texas at Austin.

Jeff Victor has worked in the computer industry since 1987, and has held software design and development, network and telecommunications management, and pre-sales technical roles. Since joining Sun in 1997, he has held various Systems Engineering roles, focusing on data center architectures and virtualization. He also maintains the *Solaris Zones and Containers FAQ*. Jeff holds a B.S. in Computer Science from Rensselaer Polytechnic Institute in Troy, New York.

13

Glossary

ACL	Access control list.
API	Application programming interface.
ABI	Application binary interface.
batch job	A workload typically thought of as a single transaction with a single purpose. Batch jobs are often assumed to proceed without the need for human intervention, performing some form of data transformation.
DAS	Direct attached storage.
Direct attached storage	Storage that resides in, or is directly connected to, a computer system.
DNS	Domain Name Service.
DSD	Dynamic System Domain.
dynamic reconfiguration	A set of enhancements to the Solaris OS that provides the capability of dynamically attaching/detaching system boards in a live system without halting the operating system or any user programs.
dynamic system domain	A single instance of the Solaris OS running on a collection of system resources, including processors, memory, networking i/o and storage.
extended accounting	A feature of the Solaris OS that provides an extensible facility for recording information about system activity on a per-process or per-task basis.
Fair Share Scheduler	Software that gives administrators the ability to specify that certain processes be given more resources than others. Should an analysis of workload data indicate that CPU resources are being monopolized by a particular workload, or are being under-utilized, administrators can modify the allocation policy for CPU resources to effect better utilization.
FSS	Fair Share Scheduler.
GUI	Graphical User Interface.

heap size The amount of memory allocated for the process data segment.

high availability An approach to achieving high information resource availability that leverages recovery between nodes.

horizontal scaling The ability to use multiple servers within a single tier.

HPC High Performance Computing.

IPMP Internet Protocol Multi-Pathing.

IPQoS IP Quality of Service. IPQoS enables bandwidth to be controlled throughout the network.

job See batch job.

LAN Local area network.

LDAP Lightweight Directory Access Protocol.

LOFS Loopback file system.

NAS Network attached storage.

NAT Network address translation.

Network attached storage Storage that provides data access over a network.

NFS Network File System.

NIS Network Information Server.

project A workload that can indicate which user or group is allowed to participate in it.

physical memory control The ability to manage the physical memory usage of an application, including the number of physical memory pages to be reserved for use by a set of processes.

QoS Quality of Service.

Quality of Service The overall quality delivered by a networked service.

resource controls A mechanism for placing bounds on resource usage to prevent workloads from over-consuming resources.

resource management The act of managing the access to, and use of, system resources by a set of services or service components.

resource pools An association of resources, such as a set of processors or pool of memory, which are reserved for exclusive use by a workload.

role based access control (RBAC) A feature of the Solaris OS that enables system administrators to enforce the security principle of least privilege, ensuring no user is given more privilege than is necessary to perform a job. With RBAC, administrators can create user accounts, or roles, for specific individuals, enabling a variety of security policies.

SAN Storage area network.

server consolidation The migration of applications from multiple servers onto a single system.

server sprawl The proliferation of individual hardware servers and accompanying management and resource allocation problems.

server virtualization An abstraction of one or more servers that provides a consistent and/or aggregated view of their external interfaces and externally visible functions.

service A set of actions that satisfy a request.

service level agreement A guarantee of the service level a computing environment will provide to a user.

service points Points of access into a Service Point Architecture.

SLA Service Level Agreement.

SMP Symmetric Multiprocessing.

Solaris Container A view of the environment within which a Solaris service or collection of Solaris service components execute and are managed.

Solaris Zone A virtualized environment within an instance of the Solaris OS.

statistical tools Software built into the Solaris OS that provides a flexible way to record resource consumption on a task or process basis.

Storage area network A network of storage devices that provide data access to servers.

Sun Cluster Software that enables systems to be grouped together to deliver high availability to mission-critical environments.

Sun Management Center Software that provides a powerful, easy-to-use platform for administrative and management operations. Providing a single point of management for all Sun systems, Sun Management Center enables system administrators to perform remote system configuration, monitor performance, and isolate hardware and software faults.

Sun Grid Engine Sophisticated software that aggregates available computing resources and delivers compute power as a network service. Users can treat a collection of distributed computing systems as a single, large computational resource and balance the workload across the systems.

Sun Trunking Technology that enables network interface ports to be grouped together to improve availability and performance.

Swap sets A mechanism for managing swap space usage on a per-application basis, without changing the application itself.

task A specific job within a project, or a collection of processes performing a single job.

tier A functionally separated hardware and software component that performs a specific function.

TS Time Sharing scheduler.

vertical scaling The ability to scale services within a system.

virtualization The act of adding a layer of abstraction to an entity, or collection of entities, which maintains the interfaces and external properties of the entity, yet enables the original entity to change.

VLAN Virtual LAN, or virtual local area network.

workload An aggregation of all processes belonging to an application, or a group of applications, that are related from a business perspective.

14

References

Articles, Books, Papers, and Guides

	Title	URL
Administration and Management	System Administration Guide: Basic Administration	http://docs.sun.com/app/docs/doc/817-1985
	System Administration Guide: Network Services	http://docs.sun.com/app/docs/doc/816-4555
	System Administration Guide: Solaris Containers-Resource Management and Solaris Zones	http://docs.sun.com/app/docs/doc/819-2450
	System Administration Guide: Security Services	http://docs.sun.com/app/docs/doc/816-4557
	Service Management Facility (SMF) in the Solaris 10 OS	http://sun.com/blueprints/0206/819-5150.pdf
Consolidation	Consolidating Applications with Solaris Containers	http://sun.com/datacenter/consolidation/solaris10_whitepaper.pdf
	Consolidation through Virtualization with Sun's x64 Servers	http://sun.com/amd/briefs/consolidation-sol-bf.pdf
	J2EE Containers	http://docs.sun.com/source/819-0215/containers.html
Grid Computing	Enterprise Grid Computing, ACM Queue, July/August 2005	http://acmqueue.com
Resource Management	Solaris 10 Resource Manager Developer's Guide	http://docs.sun.com/app/docs/doc/817-1975
Security	BigAdmin Feature Article: Practical Security Using Solaris	http://sun/com/bigadmin/features/articles/container_security.html

	Title	URL
	Jails: Confining the Omnipotent Root	http://docs.freebsd.org/44doc/papers/jail/jail.html
	Practical Security Using Solaris Containers in the Solaris 10 OS	http://sun.com/bigadmin/features/articles/container_security.html
	Secure Virtual Machine Architecture Reference Manual	http://enterprise.amd.com/downloadables/Pacifica_Spec.pdf
	Toward Systemically Secure IT Architectures	http://sun.com/software/security/docs/systemic-security-wp-1.pdf
Solaris Containers	Best Practices for Running Oracle Databases in Solaris Containers	http://sun.com/bigadmin/features/articles/db_in_containers.html
	Solaris Containers: Server Virtualization and Manageability	http://sun.com/software/solaris/whitepapers.xml
	Solaris Containers—What They Are and How to Use Them	http://sun.com/blueprints/0505/819-2679.pdf
	Zones and Containers FAQ	http://opensolaris.org/os/community/zones/faq
Sun Java Enterprise System	Sun Java Enterprise System software and documentation	http://docs.sun.com/app/docs/coll/1286.1
Workload Management	Creating Self-Balancing Solutions with Solaris Containers	http://sun.com/blueprints/0605/819-2888.pdf
	Web Search for a Planet: The Google Cluster Architecture	IEEE Computer Society, IEEE Micro, March-April, 2003. :Luiz Andre Barroso, Jeffery Dean, Urs Holzle
Virtualization	IEEE Computer, May 2005, Resource Virtualization Renaissance special issue	http://computer.org/computer/homepage/0505/GEI/
	ACM/Usenix Conference on Virtual Execution Environments, June 2005	http://veeconference.org
	Operating System Virtualization, Ideas International, Inc., July 13, 2005	http://ideasinternational.com/TTM_Sept_05.pdf

Web Sites of Interest

	Description	URL
Blogs	Sun blogs	http://blogs.sun.com/roller/page/Gregp/20050224 http://blogs.sun.com/dp http://blogs.sun.com/sch http://blogs.sun.com/lianep http://blogs.sun.com/tobin http://blogs.sun.com/darren
Consolidation	Consolidation Tool for Sun Fire Servers	http://cooltools.sourcenet.net/consolidation
	Sun and Oracle Consolidation	http://sun.com/third-party/global/oracle/consolidation/solaris10.html
	Sun Mainframe Rehosting	http://sun.com/datacenter/mainframe
Management	Sun Management Center	http://sun.com/software/products/sunmanagementcenter
	Solaris Container Manager	sun.com/software/products/container_mgr
OpenSolaris	OpenSolaris Project	http://opensolaris.org
Processors	Throughput Computing	http://sun.com/processors/throughput
Servers	Sun Servers	http://sun.com/servers
	Sun Fire E6900 Server	http://sun.com/servers/midrange/sunfire_e6900
Software	Sun Java System Application Server	http://sun.com/software/products/appsrvr
Solaris Zones	OpenSolaris Community: Zones	http://opensolaris.org/community/zones
Workload Management		
	BigAdmin System Administration Portal: DTrace	http://sun.com/bigadmin/content/dtrace
	BigAdmin System Administration Portal: Predictive Self-Healing	http://sun.com/bigadmin/content/selfheal
	BigAdmin System Administration Portal: Solaris Zones	http://sun.com/bigadmin/content/zones
	Dynamic Reconfiguration and Dynamic System Domains	http://sun.com/servers/highend/dr_sunfire

15

Index

A
Access Manager
  deployment example 160
acctadm 27
Application isolation
  definition 7
  overview 8
  relationship to Solaris Containers 8
  relationship to Solaris Zones 8
Applications
  isolating 65
B
Backup 112
  disk snapshot 115
  tape 113
Backup and restore 112
C
Calendar server
  deployment example 170
Commands
  acctadm 27
  id 26
  ipcs 26
  libpool 54
  newtask 26, 27, 43
  pgrep 26
  pkill 26
  pooladm 55
  poolbind 26, 55, 94
  poolcfg 55, 58
  poold 55
  poolstat 55
  prctl 26, 30, 33
  priocntl 26
  projadd 26
  projdel 26
  projects 26
  projmod 26
  prstat 26, 43
  ps 26
  psrset 49
  rctladm 33
  useradd 26
  wracct 27
  zlogin 69, 70
  zoneadm 68, 70
  zoneadmd 69, 70
  zonecfg 68, 70
  zonename 70
Communications Express
  deployment example 175
Consolidation
  example 151
Consolidation Tool for Sun Fire Servers 132

Container
definition 12
Containment
approaches 13
definition 12
fault 13
overview 7, 11
security 13
services 12
software-based 15
trade-offs 18
CPU
consumption
controlling 40, 41, 83
transferring 62
CPU share
definition 28
CPU shares
assigning
to projects 41
configuration 29
overview 28
D
Differentiated services
overview 6
Direct device 107
Directory server
deployment example 158
Disk snapshot 115
domains 14
Dynamic Reconfiguration 50, 54
and resource pools 50
Dynamic resource pools 52, 56
poold 53
Dynamic System Domains
approach to containment 12, 14
combining with Solaris Containers 19
E
Examples
Access Manager deployment in zones 160
Calendar server deployment in zones 170
Communications Express deployment in zones 175
Directory server deployment in zones 158
Messaging server deployment in zones 167
Messenger Express deployment in zones 173
Portal server deployment in zones 162
Extended accounting 26
commands
acctadm 27
wracct 27
prstat 26
using 44
zones 72
F
Fair Share Scheduler
and pools 51
and processor sets 52
CPU shares 28
enabling
dispadmin 40
overview 6, 27
Zones 73
File system
sparse root 102
whole root 102
File Systems
loopback 104
Solaris ZFS 104
UFS 107

File systems
direct device 107
Network File System 108
NFS 108
shareable 109
considerations 109
Flash archives 127
G
Global zone
definition 9, 17, 66
overview 17, 66
relationship to the Solaris OS 17
H
Hardware configuration managers 13
hardware domains 14
I
id 26
IP multi-pathing 118
ipcs 26
Isolating applications 65
Isolation
process 67
zones 65
K
Kernel parameters 123
extended accounting 125
file system 124
IPQoS 125
processes 123
max_nprocs 123
maxuprc 123
pidmax 123
zone.max-lwps 123
pseudo terminal 124
scheduling 125
STREAMS 124
System V IPC 124
virtual memory 124
L
libpool 54
Load
adapting 62
Loopback file system 104
M
Messaging server
deployment example 167
Messenger Express
deployment example 173
N
Namespace
isolation 129
naming services 129
Network
configuration 115
DHCP 116
IP address 116
IP multi-pathing 118
routing 117
firewalls and filters 118
Network File System 108
newtask 26, 27, 43
Non-global zone
definition 9, 17, 66
overview 17, 66
relationship to the Solaris OS 17
O
OpenSolaris 101
Oracle
running in projects 38
P
Packages 126
management 126
partitions 14
Patches
management 126
pgrep 26

pkill 26
Pool
    creating 58
pooladm 55
poolbind 26, 55, 94
poolcfg 55, 58
poold 55
Pools
    binding 61
        Solaris Containers 94
        Solaris Zones 94
    commands
        libpool 54
        pooladm 55
        poolbind 55, 94
        poolcfg 55, 58
        poold 55
        poolstat 55
    monitoring 55
poolstat 55
Portal server
    deployment example 162
prctl 26, 30, 33
Predictive Self-Healing 132
    and Solaris Containers 133, 144
Printing 119
priocntl 26
Process
    isolation 67
    relationship to tasks and projects 5
Process Rights Management 127
    Solaris privileges 127
Processes
    binding to pools 51
Processor sets 49
    and Fair Share Scheduler 52
    commands
        psrset 49
    creating 49
    overview 49
Processors
    UltraSPARC T1 151
projadd 26
projdel 26
Project
    definition 5
    relationship to processes and tasks 5
Projects
    attributes
        project.pool 51
    classifying workloads with 24
    commands
        id 26
        ipcs 26
        newtask 26, 27, 43
        pgrep 26
        pkill 26
        poolbind 26
        prctl 26, 30, 33
        priocntl 26
        projadd 26
        projdel 26
        projects 26
        projmod 26
        prstat 26, 43
        ps 26
        rctladm 33
        useradd 26
    database 25
    defining
        example 34
    defining workloads 24
    definition 24
    overview 24
projects 26
projmod 26

prstat 26, 43
Extended accounting 26
ps 26
psrset 49
R
rctladm 33
Resource allocation
definition 13
Resource capping 120, 122
guidelines 121
Resource containment 3
definition 7, 13
relationship to Solaris Containers 7
Resource control
actions
deny 31
none 31
signal 31
privilege level
basic 31
privileged 31
system 31
Resource Controls 30
Resource controls
administering 30
overview 6
process.max-address-space 32
process.max-core-size 32
process.max-cpu-time 32
process.max-data-size 32
process.max-file-descriptor 32
process.max-file-size 32
process.max-msg-messages 32
process.max-msg-qbytes 32
process.max-port-events 32
process.max-sem-nsems 32
process.max-sem-ops 32
process.max-stack-size 32
project. max-contracts 32
project. max-sem-ids 36
project. max-sem-nsems 36
project. max-shm-ids 36
project. max-shm-memory 36
project.cpu-shares 29, 30, 32
project.max-crypto-memory 32
project.max-device-locked-memory 32
project.max-lwps 32
project.max-msg-ids 32
project.max-port-ids 32
project.max-sem-ids 32
project.max-shm-ids 32
project.max-shm-memory 32
project.max-tasks 32
System V IPC 35
SEMMNI (semsys seminfo_semmni) 36
SEMMNS (semsys seminfo_semmns) 36
SEMMSL (semsys seminfo_semmsl) 36
SHMMAX (shmsys shminfo_shmmax) 36
SHMMIN (shmsys shminfo_shmmin) 36
SHMMNI (shmsys shminfo_shmmni) 36
SHMSEG (shmsys shminfo_shmseg) 36
task.max-cpu-time 32
task.max-lwps 32
zone.cpu-shares 32
zone.max-lwps 32
Resource management
concepts 3
kernel parameters 123
extended accounting 125

file system 124
IPQoS 125
processes 123
max_nprocs 123
maxuprc 123
pidmax 123
zone.max-lwps 123
pseudo terminal 124
scheduling 125
STREAMS 124
System V IPC 124
virtual memory 124
resource capping 120, 122
guidelines 121
Solaris Containers 101
zones 72
Resource pool
creating 58
definition 4
Resource Pools
monitoring 55
overview 50
Resource pools
binding 61
creating
example 55
dynamic 52, 56
poold 53
in Containers 91
zones 72
Resource sets
definition 4
Resources
managing 49
rlimit 30
S
Scalability
definition 13
Security 127
Access control 128
Basic Security Model 128
intruder traps 129
role-based access control 128
Auditing 128
Basic Security Model 128
intruder traps 129
role-based access control 128
Process Rights Management 127
Solaris privileges 127
risks 119
Solaris Containers
Solaris Containers
security
risks 119
zones 65
Security isolation
definition 8
relationship to Solaris Containers 8
server virtualization
overview 7
Servers
Sun Fire T2000 151
Service
definition 5, 12
Service Management Facility
commands
inetadm 137
svcadm 137
svccfg 137
svcprop 137
svcs 137
compatibility 136
manifests 135
services 134
profiles 137, 142
starting 136

stopping 136
states 135
Solaris Container Manager 131
Solaris Containers
and Predictive Self-Healing 144
approach to containment 12
backup
disk snapshot 115
tape 113
backup and restore 112
consolidation
example 151
construction 91
creating 89
sparse root
Solaris Containers
creating
whole root 125
definition 7
disk snapshot 115
file system
direct device 107
file systems 102
loopback 104
Network File System 108
NFS 108
UFS 107
guidelines for deployment 20
integration 101
network 101
firewalls and filters 118
network configuration 115
DHCP 116
IP address 116
IP multi-pathing 118
routing 117
overview 2, 16
package management 126
patches 126
pools
binding 94
creating 91
printing 119
provisioning
Solaris Containers
installation 125
resource capping 120, 122
guidelines 121
resource management 101
security 119
storage 101
configuration 102
file system 102
direct attached 103
network attached 103
storage area network 103
tape backup 113
Solaris Dynamic Tracing
overview 18
Solaris Service Manager 134
Solaris ZFS 104
Solaris Zones 65
administering 67
backup
disk snapshot 115
tape 113
backup and restore 112
booting 76
commands
zlogin 69, 70
zoneadm 68, 70
zoneadmd 69, 70
zonecfg 68, 70
zonename 70
configuration 68, 154
verification 97

configuring 75
CPU consumption 83
creating 75
    sparse root 125
    whole root 125
definition 8
deployment
    example 152
devices 68
disk snapshot 115
extended accounting 72
Fair Share Scheduler 73
file systems 68, 71
installation 125
installing 68, 76
login 69
managing 157
monitoring 157
network
    firewalls and filters 118
network configuration 115
    DHCP 116
    IP address 116
    IP multi-pathing 118
    routing 117
network interface 68
overview 65, 66
pools
    binding 94
printing 119
properties 68
provisioning 125
relationship to Solaris Containers 8
resource capping 120, 122
    guidelines 121
resource controls 68
resource pools 72
tape backup 113
    Using
        example 73
    zone name 68
    zone path 68
Solaris zones
    resource management 72
Sparse root 102
    creating 125
Storage
    configuration 102
        file system 102
    direct attached 103
    file system
        direct device 107
    file systems
        Solaris ZFS 104
        UFS 107
    network attached 103
    Solaris Containers
        direct attached 103
        network attached 103
        storage area network 103
    storage area network 103
Sun Fire T2000 Server 151
Sun Java Enterprise System 152
    deployment
        example 152
    performance tuning 180
Sun Management Center 131
Sun Trunking 118
System V
    IPC resource controls 35
        SEMMNI (semsys
            seminfo_semmni) 36
        SEMMNS (semsys
            seminfo_semmns) 36
        SEMMSL (semsys
            seminfo_semmsl) 36

SHMMAX (shmsys shminfo_shmmax) 36
SHMMIN (shmsys shminfo_shmmin) 36
SHMMNI (shmsys shminfo_shmmni) 36
SHMSEG (shmsys shminfo_shmseg) 36
T
Tape backup 113
Task
definition 5
relationship to processes and projects 5
Thresholds
determining 33
Time Share scheduler 51
Troubleshooting 147
Networks 150
Zone access 147
Zones
boot single user 150
with console login 149
with safe-mode login 149
with Telnet and shells 147
with zlogin 148
U
UFS 107
UltraSPARC T1 processor 151
UNIX File System 107
useradd 26
V
Virtual machine monitor
hosted 15
Virtualization
definition 7
of file systems 66
overview 7, 11
relationship to Solaris Containers 7
trade-offs 18
zones 65
Virutal machine monitors 13
W
Whole root 102
creating 125
Workload
definition 5, 23
management 23
visibility 13
Workload classification
overview 4
Workloads
definining 24
managing
example 33
wracct 27
Z
zlogin 69, 70
zoneadm 68, 70
zoneadmd 69, 70
zonecfg 68, 70
zonename 70

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN **Web** sun.com