Sun microsystems

SOLARIS™ CONTAINERS TECHNOLOGY ARCHITECTURE GUIDE

Jeff Victor, Sun Microsystems

Sun BluePrints™ OnLine — May 2006

SUN BLUEPRINTS APPROVED

Part No 819-6186-11
Revision 1.1, 6/2/06
Edition: May 2006

Table of Contents

Chapter 1
An Introduction to Solaris™ Containers Technology

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. 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 gives organizations a great deal of flexibility when designing computer architectures. An integral part of the Solaris™ Operating System (Solaris™ OS), Solaris Containers isolate software applications and services using flexible, software-defined boundaries. A breakthrough approach to virtualization and software partitioning, Solaris Containers allow 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. As a result, each zone behaves as if it is running on its own system, making consolidation simple, safe, and secure. While each zone appears to be separate, all zones on a system access, and vie for, shared hardware resources. To enable the consolidation of workloads in data center environments, tools are included to manage hardware resource utilization. Furthermore, the resource accounting tools in the Solaris OS undertand Containers, and can export account records that enable per-Containers chargeback reporting.

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.

The Philosophy and Features of Solaris Containers Technology

Solaris Containers technology is designed with several principles in mind:

- *Security isolation*, ensuring intruders that break into one zone do not have access to other zones on the system.
- *Application isolation*, ensuring applications in one zone cannot interact with applications in other zones. Application interaction is permitted for network Internet protocol (IP) communication, or when granted specifically by the global zone administrator.

- *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.
- *Granularity*, 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.
- *Transparency*, ensuring a virtual environment does 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.

These principles enable several features of Solaris Containers technology, including:

- Multiple virtualized operating systems with secure boundaries, reducing system complexity
- Safe application consolidation, reducing administrative costs
- Application isolation, reducing conflicts between applications running on the same system
- Resource allocation, including CPU, physical memory, network bandwidth, and more based on workload and business conditions
- Resource containment and control, including the ability to constrain CPU usage when the system is experiencing CPU contention
- Assignment of virtual environments to zones, enabling each environment to have its own settings rather than relying on system-wide settings
- Error isolation, helping to minimize unplanned service downtime
- Predictable service levels
- Private and shared file systems
- Separate IP address(es) per Container, with the ability to multiplex several Containers onto a network interface card (NIC) or segregate network traffic by NIC
- Support for multiplexing disk I/O from multiple Containers onto a host bus adapter (HBA), or segregating disk I/O
- Ability to boot Containers manually, programmatically, or automatically when the system is booted
- Ability to create Containers manually or programmatically
- Ability to halt the Container from within the Container or the global zone
- Support for the Dynamic Tracing (DTrace) facility, enabling the interactions of applications running in different Containers to be observed
- Ability to use package and patch tools to modify the global zone, a subset of existing Containers, or all Containers and the global zone, depending on patch requirements

Solaris Containers technology provides extensive flexibility, and selecting an appropriate set of these features for a given environment can be challenging. This Sun BluePrints™ article 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.

Chapter 2
Using Solaris Containers Technology

Terminology

The Container functionality available in the Solaris 10 OS is the result of innovative additions made to the operating system since the Solaris 8 OS. Solaris 10 Containers comprise two technologies: Solaris Zones partitioning technology and resource management features. *Solaris Zones* provide virtualized operating environments that have their own hostname, IP address(es), users, file systems, and so on, while giving applications isolation and protection from each other. *Resource management* controls how system resources are spread across specified workloads.

A Solaris 10 system can have two types of Solaris Zones: the global zone and non-global zones. The standard environment when Solaris 10 is first installed is also called the *global zone*. *Non-global zones* are the optional virtual operating environments that can be created to house applications.

Designed to meet the most common needs of server virtualization, Solaris Containers 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.

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.

Non-global zones are zones which do not reside in the global zone. In addition to differences in directory structure, 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.

- *ZFS file system*

  Introduced in Solaris 10 6/06, ZFS is a new file system that includes three different ZFS objects which can be used within a Solaris Container: a filesystem, 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.[1] 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.

```
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
```

[1] A ZFS volume cannot be made available to a Container.

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 (UFS)*

  When using 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
```

Global zone administrators can accomplish this manually, even when the zone is running. Directories and files available to the non-global zone can be managed from the global zone. However, the zone cannot boot while the file system is mounted.

```
global# mount  /dev/dsk/c1t0d0s6   /zones/zone1/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
   zone1# newfs /dev/rdsk/c1t0d0s6
   zone1# mount /dev/dsk/c1t0d0s6 /opt/local
```

  This can be accomplished manually with the following command.

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

- *Network file system (NFS)*

  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.

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 2-1).

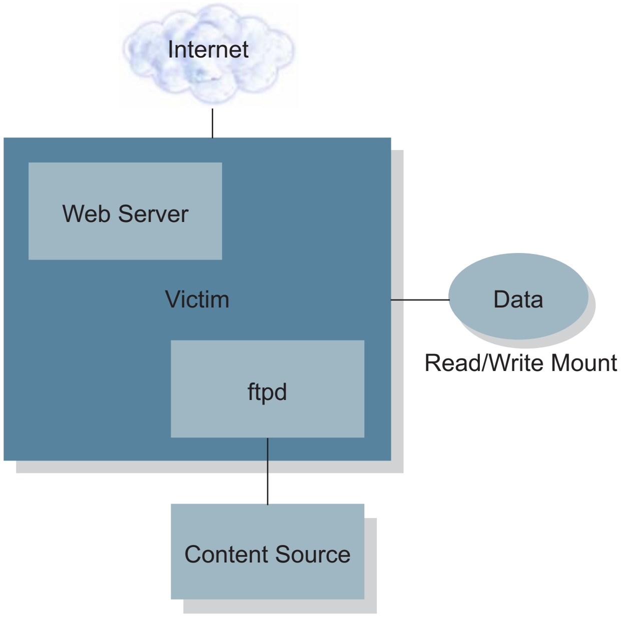


Figure 2-1. A typical Web server that is prone to attack

  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 2-2).

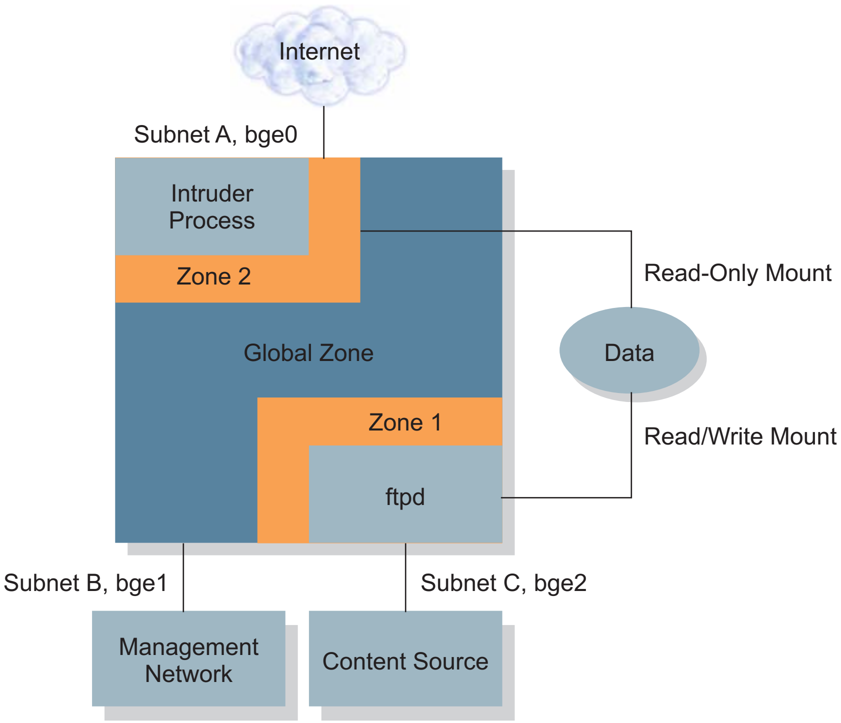


Figure 2-2. The use of Solaris Containers technology can help prevent attacks

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.

- *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 Container'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 that has been 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 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 2-3 depicts a simple diagram of a common NetBackup architecture.

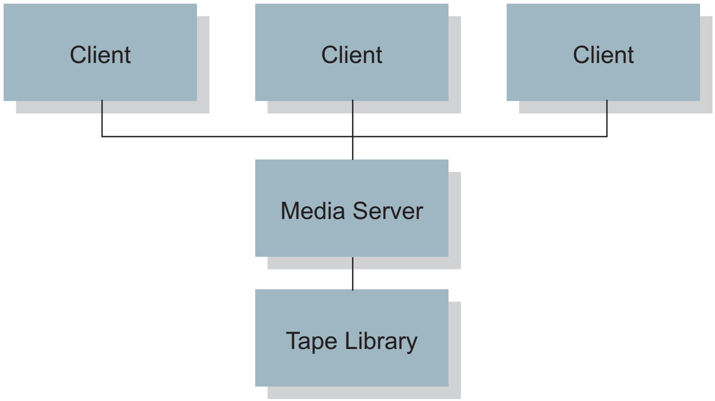


Figure 2-3. A typical NetBackup architecture

Figure 2-4 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.

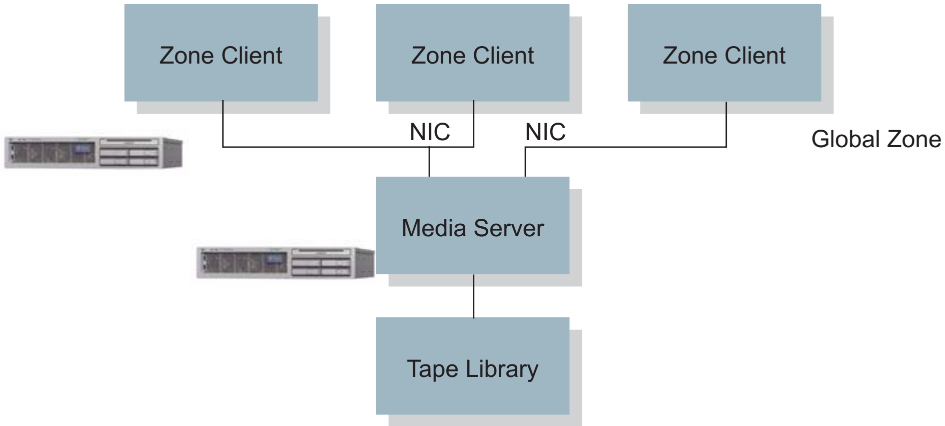


Figure 2-4. A backup architecture that uses Solaris Zones technology and a separate media server

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 2-5). 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.

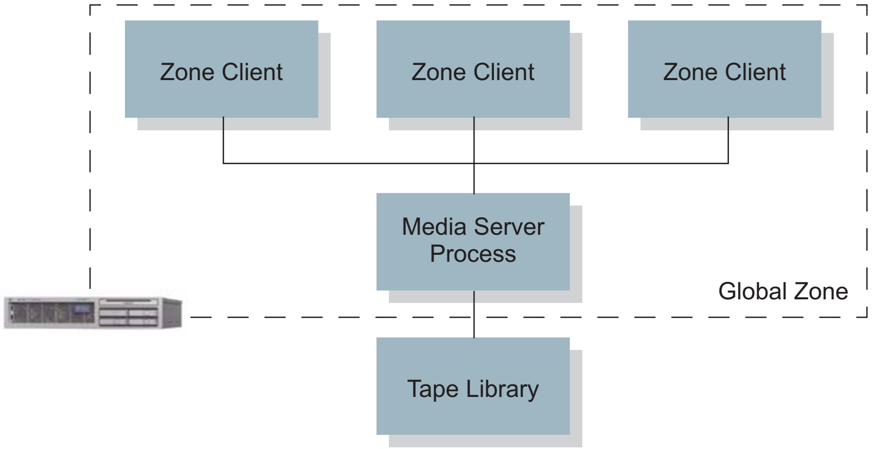


Figure 2-5. A backup architecture that places the media server in the global zone

In all cases, knowledge of the data transmission rates along the backup stream can prove helpful. Table 2-1 lists well-known data rates, along with inter-zone network transmission rates measured on some systems.

Table 2-1. 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 2-4, 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 2-5 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. Currently, these IP filters cannot be used to filter traffic passing between zones, 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

The ability to minimize cross-workload performance compromises, combined with facilities that monitor resource usage and utilization, are collectively referred to as *resource management*. The following sections provide an overview of resource management considerations. A thorough discussion of this topic can be found in *Solaris Containers—What They Are and How to Use Them* located at *http://www.sun.com/blueprints/0505/819-2679.pdf*

Processor Sets, Pools, and Projects

A *processor set* is a grouping of processors. One or more workloads can be assigned to a processor set using *Dynamic Resource Pools*, collections of resources that are reserved for exclusive use by an application or set of applications. Other workloads may not run on the processors assigned to the processor set. Dynamic Resource Pools enable a workload, such as a Container, to be assigned to a processor set.

- The Solaris OS can increase or decrease the number of processors in a Dynamic Resource Pool according to guidelines set by administrators, including a minimum and maximum quantity.
- Multiple Containers may be assigned to a pool.

  The processor utilization of each Container in the pool can be managed with the *Fair Share Scheduler*, a tool that gives administrators the ability to specify that certain processes be given more resources than others, and guarantee CPU resource availability to applications. To ensure the global zone has sufficient shares when using the Fair Share Scheduler, be sure to assign a specific number of shares to the global zone. This value can be set using the `prctl(1M)` command on a running system.

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

  Note this setting is not retained after rebooting. Appendix A provides a Service Management Facility (SMF) service to automatically set the number of shares assigned to the global zone when booting.

CPU Resources

Processor sets can be useful when limiting the maximum processor utilization consumed by a workload. This can reduce the number of CPUs or CPU cores specified when licensing some software products. To guarantee a minimum amount of processor usage, use the Fair Share Scheduler software.

Resource Capping Explained

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. 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.

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 2-2 lists these privileges.

Table 2-2. 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.

Solaris Containers and the Service Management Facility

The Service Management Facility (SMF) includes the */system/zones* service. This service automatically boots all zones that have `autoboot` set to `true` (`autoboot=true`). By default, this service is enabled. However, it can be enabled or disabled manually. While disabling this service disables automatic zone booting, global zone administrators can manually boot zones, if desired.

The combination of Solaris Containers and the SMF enables the creation of services which manage individual zones. This enables SMF to create dependencies between zones. Consider two applications, A and B. These applications must be deployed in separate zones, and application A must complete its initialization before application B starts the initialization process. Service A can be created in the global zone and defined so that SMF restarts the zone if needed. Similarly, service B can be created in the global zone such that it depends on service A. If one or both of those services fail, the SMF system in the global zone reboots zone A. After application A completes its initialization, zone B is rebooted.

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 (SunMC) and the Solaris Container Manager, 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. The use of the Sun Management Center software is outside the scope of this document.

Solaris Container Manager and Webmin

Solaris Container Manager is a graphical user interface for managing projects, resource pools, zones, and alarms. A third-party system administration tool, Webmin, provides a subset of this functionality. Webmin supports the following functions on a zone: create, boot, reboot, halt, reconfigure, and uninstall. More information on the Webmin software is available at *http://www.webmin.com*

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.

Chapter 3
Server Virtualization and Consolidation—An Example

This chapter provides a consolidation planning example using an Information Technology group for a fictional organization named Galactic Department of Information Technology (GDIT). It discusses the motivation for consolidating some servers, and the reasons for not consolidating others. The recommendations described here can be implemented individually or in combination. The implementation can be accomplished by in-house staff, or through engagement of Sun or third-party partners.

The broad rationale for consolidation is reviewed, and recommendations are provided for the consolidation of specific groups of servers. It does not describe current hardware utilization, or make recommendations concerning the sizing of specific servers. The relevant hardware and its utilization should be examined closely in order to identify opportunities to enhance consolidated servers. Doing so helps ensure servers are able to handle the new combined load without undesirable side effects, such as lengthy response times or poor availability levels.

Current Situation

GDIT currently manages the following environments. All environment utilize Sun systems with UltraSPARC® processors running the Solaris OS.

- Galactic Information Service

  The Galactic Information Service (GIS) provides information regarding Galactic services for residents and businesses. This service provides government and non-emergency services information.

- GalactiCollab

  Galactic employs over 300,000 personnel. GalactiCollab is the employee intranet, and provides a community in which workers can share information, access resources and forms, and access productivity tools.

- GalactiMap

  GalactiMap is a central mapping data repository and set of applications which provide access to the data repository. It serves as a vehicle for facilitating the sharing of Information between and within various internal agencies.

- Galaxy.gov Portal

  Galaxy.gov is the official Galactic Web site. This site aims to provide the public with quick and easy access to information on more than 60,000 galactic agencies, programs, and services. The homepage also provides information about cultural, educational, and recreational activities in the galaxy through links to external sites.

- Libra

  Enables criminal justice and law enforcement agencies to share information about criminals in real time.

Each environment consists of a set of three-tier architectures. Each set member supports a phase in the application life cycle.

- *Sandbox*, a small system used to test basic software functionality and compatibility
- *Development*, a system used to create or modify existing software that contains a small three-tier architecture
- *Test*, a system that contains all three software tiers and is used to evaluate the functionality of new or modified software
- *Staging*, an environment identical to the production environment that is used to test performance, scalability, and availability
- *Production*, the deployment environment for the service, usually consisting of multiple systems

Not every application includes every set. For example, the GalactiCollab environment does not include test systems.

It is important to note that databases are commonly consolidated, either in a single compute environment or in separate virtual compute environments. GDIT consolidated a number of database instances, and few situations exist that can benefit from further consolidation.

Consolidation Goals

GDIT wants to consolidate a portion of its 200+ Sun servers to achieve higher resource utilization rates and enable future hardware acquisitions to be hosted in existing systems, where appropriate.

The Solaris 10 Operating System

The Solaris 10 OS offers several significant features, including Solaris Containers, Dynamic Tracing, the Service Management Facility, and Process Rights Management. Many of these features can be combined to achieve an ideal environment for server consolidation. Benefits of hardware consolidation include:

- Less hardware to acquire
- Fewer systems to administer
- Lower technical support fees
- Fewer software licenses
- Improved hardware resource utilization and return on investment

GDIT would like to utilize Solaris 10 OS technology to achieve these benefits.

Big Rules for the Solution

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

Generic Rules

- 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. These 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 Containers, network addresses, and file systems. Solaris Containers permit flexibility in each of these areas. Local conventions make it easier to deploy and administer 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 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.

GDIT Rules and Factors

- Upcoming projects provide an excellent opportunity to begin consolidation efforts, as initial hardware acquisitions can be reduced at the start of the project.
- Sandbox and development systems can be consolidated with minimal effort and risk.
- Currently a staging system is only used for final testing, and sits idle most of the time. An opportunity exists to consolidate staging systems from different environments. Doing so assumes the environments can be scheduled to avoid simultaneous use.
- All network interfaces use Gigabit Ethernet connections that are under-utilized. Multiplexing the network traffic from multiple Containers onto a single physical interface should not negatively impact performance.
- Few systems considered here have average CPU utilization greater than 30 percent, or physical memory usage greater than 50 percent.
- Except for systems that already host multiple database instances, all systems use internal storage.
- Internet-facing Web servers cannot be consolidated with intranet-facing Web servers until security implications are completely understood by GDIT technical staff.
- Unlike other systems, the infrastructure supporting the GIS environment is housed in the call center building. Consolidating mission-critical GIS systems with non-GIS systems is not permitted.

Consolidation Plan

The consolidation plan is implemented in two phases.

Phase 1

The first phase of the consolidation plan recognizes that GDIT is researching the acquisition of several dozen servers to host the Libra environment. This provides an excellent opportunity to reuse existing system capacity and reduce hardware and software acquisition, license, and support costs. In preparation, it is beneficial to consolidate several existing servers and prepare them to host the Libra software. GDIT staff identified the existing GalactiCollab environment as the best candidate for this phase of the project.

Using the rules identified above, it is clear that the simplest two systems to consolidate are the GalactiCollab sandbox and development systems. The sandbox system is used to test basic functionality and compatibility of new operating system revisions and patches, and new application software and patches. The sandbox system does not get much use. However, the sandbox system needs configuration flexibility, and requires the use of the whole root Container model which stores multiple copies of system software rather than sharing a single per-OS copy. While this model consumes disk space, it provides added flexibility for maintaining multiple software levels.

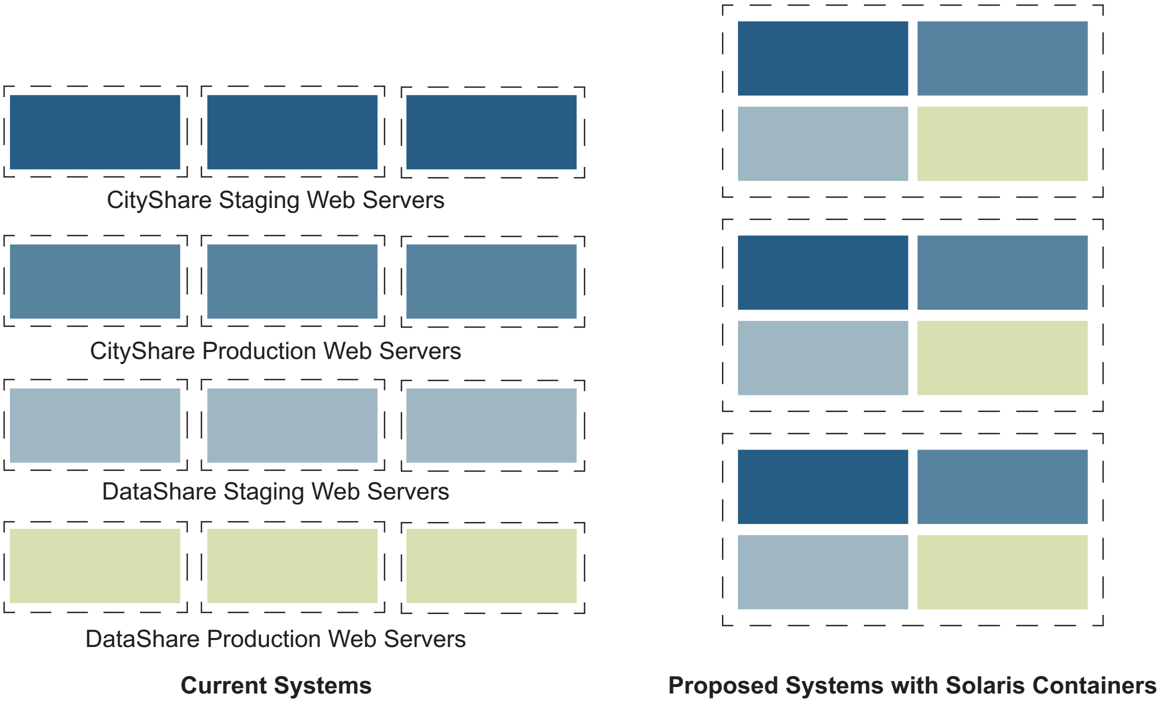
The current GalactiCollab development system houses all three tiers: Web, application, and database. These tiers can coexist on the consolidated system, however, deploying them in separate Containers provides:

- Increased resource management functionality
- Greater flexibility for reboot of one virtual environment without disturbing the other tiers
- Preparation for hosting the sandbox and development environments for future Libra projects

The staging and production Web servers are also candidates for consolidation. These servers all run the Sun Java Enterprise System Web Server software and the BEA WebLogic application connector. Because these servers do not have Internet-facing connections, the organization is not concerned about introducing new security issues.

The staging and production Web servers utilize ten servers. Each area maintains three or four servers to improve availability—and availability cannot be sacrificed in the consolidation effort. However, one server from each of the four environments can be consolidated and placed into one Container on a system. This process can be repeated for each of the other servers in each area, as shown Figure 3-1. Each shaded box in the Current Systems section represents a Container within a server. The same manner of consolidation can be applied to the application servers in those four areas.

The database instances for these four areas are already consolidated into one or more Sun Fire 15K Server domains. Further consolidation is not required. However, the Oracle database instance should be consolidated and moved to the GIS database cluster secondary node. This should result in immediate cost savings, as software licenses and support contracts can be reduced.


CityShare Staging Web Servers
CityShare Production Web Servers
DataShare Staging Web Servers
DataShare Production Web Servers

Current Systems

Proposed Systems with Solaris Containers

Figure 3-1. The staging and production Web servers can be consolidated onto fewer systems

Phase 2

The second phase involves consolidating servers in the GIS and Portal environments. The same general approach for identifying targets is applied to these systems, leading to several areas being selected for consolidation. However, the analysis yields a significant difference: the Internet-facing GIS servers are Sun Java Enterprise System proxy servers, and the GIS Web servers reside in the same security zone as the application servers. Fortunately, the Internet-facing Portal Web servers can be consolidated with the GIS proxy servers in the staging and production areas. However, the GIS and Portal staging and production systems should not be consolidated due to different availability requirements and the need to stage new operating system versions in the production environment. When taken into account, these factors lead to the following servers being consolidated using Containers:

- All sandbox and development systems in the GIS and Portal environments consolidated onto a single system
- All GIS staging proxy servers and all Portal staging Web servers consolidated onto three servers
- All GIS production proxy servers and all Portal production Web servers consolidated onto three servers
- GIS and Portal staging application servers consolidated onto two to four servers
- GIS and Portal production application servers consolidated onto two to four servers

Other systems are also prime candidates for consolidation with Solaris Containers technology:

- Separate GIS test systems that are used for two different sets of tests.
- The Interwoven development system can be moved to a Container within the combined GIS and Portal sandbox and development system.

Final Result

Tables 3-1 and 3-2 summarize the current and proposed consolidated environments. Table 3-1 describes the current mix of systems. Each cell in the Sandox, Development, Test, Staging, and Production columns represents one server. The `Shared Domain` notation indicates a database instance that has already been consolidated into a Sun Fire 15K domain with other instances. The `Dual Environment` notation denotes two separate software test areas, and `Combined Tiers` means the Web, application, and database tiers are already combined on one system.

Table 3-2 depicts the result after consolidation. Note that colored cells with the same name have been consolidated, along with the systems shown by merged cells in the table. These results indicate GDIT is in a position to consolidate a few dozen servers into onto fewer systems. Doing so can reduce the cost of operating these environments through reduced software license fees, and hardware and software support costs. In addition, consolidation frees dozens of computers for future projects, reducing or eliminating the need to acquire new hardware.

Table 3-1. Current systems

	Tier	Sandbox	Development	Test	Staging	Production
Information Service	Web			Dual Environment	Web	Web
	Application			Dual Environment	Application	Application
	Database			Dual Environment	Shared Domain	Shared Domain
GallactiCollab	Web	Combined Tiers	Combined Tiers	▨	Web	Web
	Application				Application	Application
	Database				Shared Domain	Shared Domain
Libra	Web	Combined Tiers	Combined Tiers	▨	Web	Web
	Application				Application	Application
	Database				Shared Domain	Shared Domain
GalactiMap	Web	Combined Tiers	Combined Tiers	Web	Web	Web
	Application			Application	Application	Application
	Database			Shared Domain	Shared Domain	Shared Domain
Portal	Web	Combined Tiers	Combined Tiers	Web	Web	Web
	Application			Application	Application	Application
	Database			Shared Domain	Shared Domain	Shared Domain

Table 3-2. Systems after consolidation

	Tier	Sandbox	Development	Test	Staging	Production
Information Service	Web	Containers for Sandbox, and Web, Application, and Database Tiers		Combined Dual Environment	Web	Web
	Application				Application	Application
	Database				Shared Domain	Shared Domain
GallactiCollab	Web	Containers for Sandbox, and Web, Application, and Database Tiers		▨	Web 1	
	Application				Application 1	Application 2
	Database				Shared Domain	Shared Domain
Libra	Web			▨	Web 1	
	Application				Application	Application
	Database				Shared Domain	Shared Domain
GalactiMap	Web	Containers for Sandbox, and Web, Application, and Database Tiers		Web	Web 2 (Proxy)	Web 3
	Application			Application	Application 3 and Web	Application 4
	Database			Shared Domain	Shared Domain	Shared Domain
Portal	Web			Web	Web 2	Web 3
	Application			Application	Application 3	Application 4
	Database			Shared Domain	Shared Domain	Shared Domain

Chapter 4
References

About the Author

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* (see References below). Jeff holds a B.S. in Computer Science from Rensselaer Polytechnic Institute in Troy, New York.

Acknowledgments

The author thanks the following for their advice and contributions to this article:

- Mark Huff
- Menno Lageman
- James Carlson
- Dan Price
- David Comay
- Isaac Rozenfeld
- Margaret Bierman

References

Best Practices for Running Oracle Databases in Solaris Containers
`http://www.sun.com/bigadmin/features/articles/db_in_containers.html`

OpenSolaris Web Site:
`http://www.opensolaris.org`

Practical Security Using Solaris Containers in the Solaris 10 OS
`http://www.sun.com/bigadmin/features/articles/container_security.html`

Solaris Containers—What They Are and How to Use Them
`http://www.sun.com/blueprints/0505/819-2679.pdf`

System Administration Guide: Security Services
`http://docs.sun.com/app/docs/doc/816-4557`

System Administration Guide: Solaris Containers-Resource Management and Solaris Zones
`http://docs.sun.com/app/docs/doc/819-2450`

Zones and Containers FAQ:
`http://www.opensolaris.org/os/community/zones/faq/`

Ordering Sun Documents

The SunDocs^{SM} program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You 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`

Appendix A

Assigning CPU Shares to the Global Zone with the Service Management Facility

If Containers are running in the same resource pool as the global zone—including in the default pool— and the Fair Share Scheduler is used in the pool, the number of shares assigned is used to determine the amount of processing capacity available to the global zone. By default, the global zone has one share. To ensure the global zone is given sufficient processing power, the number of shares assigned to likely needs to be set to a higher value.

Unfortunately, the number of shares assigned to the global zone is reset to one at boot time. However, the Service Management Facility (SMF) can be used to set the number of shares at boot time. First, define a new service called `svc:/system/global-cpu-shares`, as follows.

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<!--
        Service manifest for global-cpu-shares.
        Menno Lageman November 2004
-->

<service_bundle type='manifest' name='globalzone'>

<service
        name='system/global-cpu-shares'
        type='service'
        version='1'>

        <create_default_instance enabled='false' />

        <single_instance />

        <dependency
                name='multi-user-server'
                type='service'
                grouping='require_all'
                restart_on='none'>
                <service_fmri value='svc:/milestone/multi-user-server' />
        </dependency>

        <!-- We want to run before any containers are started -->
        <dependent
                name='zones'
                grouping='optional_all'
                restart_on='none'>
                <service_fmri value='svc:/system/zones' />
        </dependent>

        <exec_method
                type='method'
                name='start'
```

```
exec='/lib/svc/method/global-cpu-shares'
                timeout_seconds='60' />

        <exec_method
                type='method'
                name='refresh'
                exec='/lib/svc/method/global-cpu-shares'
                timeout_seconds='60' />

        <exec_method
                type='method'
                name='stop'
                exec=':true'
                timeout_seconds='60' />

        <property_group name='startd' type='framework'>
                <propval name='duration' type='astring'
                        value='transient' />
        </property_group>

        <!-- zone.cpu-shares -->
        <property_group name='application' type='application'>
                <propval name='cpu-shares' type='count'
                        value='1000' />
        </property_group>

        <stability value='Unstable' />

        <template>
                <common_name>
                        <loctext xml:lang='C'>
                                Set zone.cpu-shares for the global zone
                        </loctext>
                </common_name>
        </template>
</service>

</service_bundle>
```

This service executes the program */lib/svc/method/global-cpu-shares* whenever it is started or manually refreshed. As an SMF service, it is automatically started at system boot unless it is manually disabled. Starting or refreshing the service runs the script located at */lib/svc/method/global-cpu-shares*.

```
#!/sbin/sh
#
# global-cpu-shares
#
# smf(5) transient service that sets zone.cpu-shares for the global zone
#
# Menno Lageman November 2004
#

# only relevant for the global zone
if [ "${_INIT_ZONENAME:=`/sbin/zonename`}" = "global" ]; then

        shares=`/usr/bin/svcprop -p application/cpu-shares \
                        system/global-cpu-shares`

        if [ -n "${shares}" -a "${shares}" -gt 0 ]; then
                /usr/bin/prctl -n zone.cpu-shares -t privileged -r \
                        -v ${shares} -i zone global
                exit $?
        fi
fi
exit 1
```

The number of shares can be changed with the following commands:

```
# svccfg -s global-cpu-shares
svc:/system/global-cpu-shares> setprop application/cpu-shares=100
svc:/system/global-cpu-shares> ^D
# svcadm refresh global-cpu-shares
```

Appendix B

Limiting the Number of Processes Zones Can Create

It may be desirable to limit the number of processes a zone can create in some situations. The DTrace script shown below can be used to accomplish this task, and can be tailored for particular environments. It is important to understand this script prior to use. To use the script, login to the global zone as the `root` user, or another sufficiently privileged user. Save the script in a file, such as `zprocs.d`, and run the script with the with the `dtrace -w -s zprocs.d` command.

```
#! dtrace

/* Clause to initialize an array entry and report on it.
/  The 'trace()' call can be deleted if you don't want output.
/*/
fbt::zone_uniqid:entry
{
  pcount[zonename]=0;
  trace(zonename);
}

/* Clause to increment the appropriate array value whenever
/  a new process is created in that zone.  */
proc:::create
/zonename != "global"/
{
 pcount[zonename]++;
 trace(pcount[zonename]);
}

/* Clause to mark a process for destruction before it is
/  fully created.  The printf() calls are optional.  */
proc:::create
/zonename != "global" && pcount[zonename]>500/
{
 printf ("parent execname: %s\n", execname);
 printf ("child pid: %d\n", args[0]->pr_pid);
 killnext[pid, args[0]->pr_pid] = 1;
}

/* Clause to actually kill the process. The printf() is
/  optional. */
proc:::start
/killnext[curpsinfo->pr_ppid, pid]/
{
 printf ("killing execname %s\n", execname);
 raise (9);
 killnext[curpsinfo->pr_ppid, pid] = 0;
}

/* Clause to decrement the appropriate array value whenever
/  a process in a zone exits. */
proc:::exit
/zonename != "global"/
{ pcount[zonename]--; trace(pcount[zonename]);}
```

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN **Web** sun.com