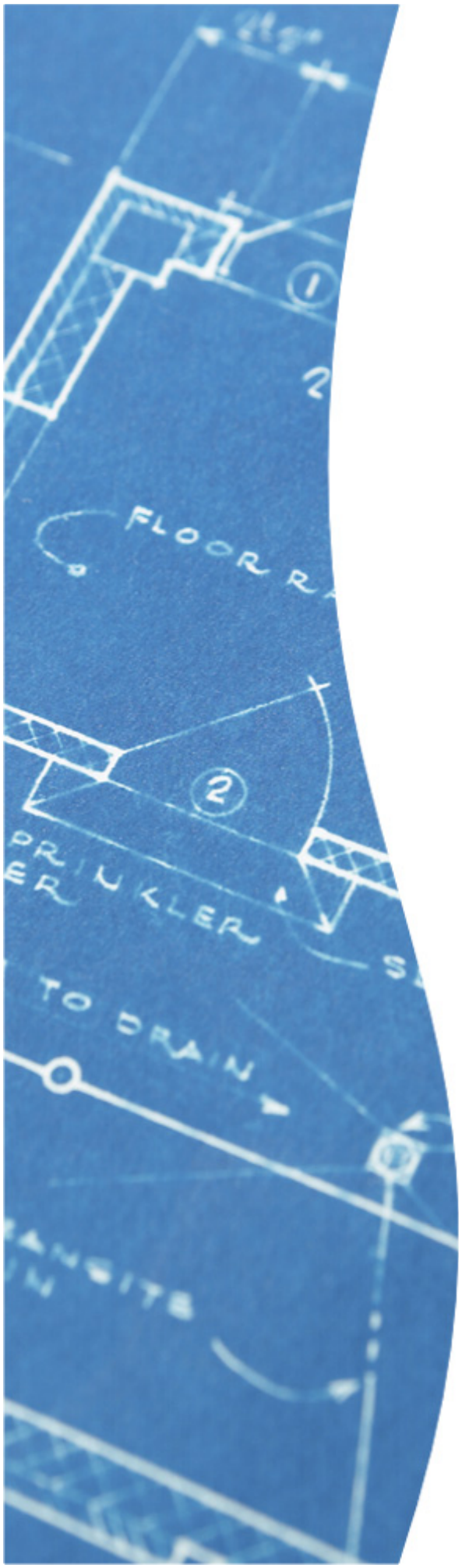

BEST PRACTICES FOR RUNNING ORACLE DATABASES IN SOLARIS™ CONTAINERS

Ritu Kamboj, ISV Engineering
Roman Ivanov, ISV Engineering

Sun BluePrints™ Online

Part No 820-7195-10
Revision 1.0, 1/16/09

Table of Contents

Best Practices for Running Oracle Databases in Solaris™ Containers

The Solaris™ Operating System (Solaris OS) includes support for *Solaris Containers*, a virtualization technology that provides isolated and secure runtime environments within a single Solaris OS instance. Using Solaris Containers, administrators can manage separate workloads, control resource usage, and maintain IP network separation. These features can enable multiple applications, or even multiple instances of the same application, to securely coexist on a single system, providing potential server consolidation savings.

Both Oracle 9i R2 and 10g R2 databases have been certified to run in a Solaris Container. A licensing agreement between Sun and Oracle recognizes Solaris 10 OS capped containers as hard partitions. The ability to license only the CPUs or cores configured in a Solaris Container provides flexibility, consolidation opportunities, and possible cost savings.

This article addresses the following topics:

Note – All Solaris OS features described in this document require the use of Solaris 10 10/08 (U6) or later releases. This document does not address Oracle Real Application Cluster (RAC), but concentrates solely on non-RAC Oracle databases. It is also beyond the scope of this document to explain how Solaris Containers technology can be used to consolidate multiple Oracle database instances in separate containers on the same system. See references [1] and [3] for detailed information about using Solaris Containers technology for server consolidation.

Solaris Containers

Solaris Containers, Sun's operating system level virtualization technology, provide complete, isolated, and secure runtime environments for applications. This technology allows application components to be isolated from each other using flexible, software-

defined boundaries. Solaris Containers are designed to provide fine-grained control over resources that the applications use, allowing multiple applications to operate on a single Solaris 10 OS instance while maintaining specified service levels (Figure 1).

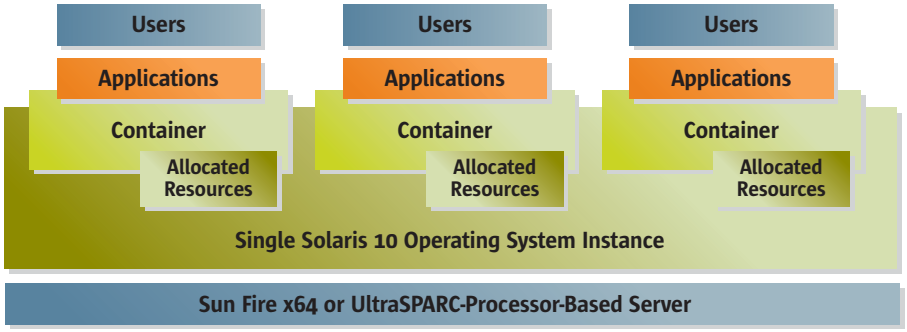

Figure 1. Multiple Solaris Containers on a single Solaris 10 OS instance.

Solaris Containers use *Solaris Resource Manager (SRM)* features along with *Solaris Zones* software partitioning technology to deliver a virtualized environment that can have fixed resource boundaries for application workloads. These two major components of Solaris Containers are discussed in the following sections. For more detailed information about these technologies, see references [2] and [4].

Solaris Zones Partitioning Technology

Solaris Zones, a component of the Solaris Containers environment, is a software partitioning technology that virtualizes operating system services and provides an isolated and secure environment for running applications. Solaris Zones are ideal for environments that consolidate multiple applications on a single server.

There are two types of zones: *global zones* and *non-global zones*. The underlying OS, which is the Solaris instance booted by the system hardware, is called the global zone. There is only one global zone per system, which is both the default zone for the system and the zone used for system-wide administrative control. One or more non-global zones can be created by an administrator of a global zone. Once created, these non-global zones can be administered by individual non-global zone administrators, whose privileges are confined to that non-global zone.

Two types of non-global zones can be created using different root file system models: *sparse root* and *whole root*.

- *Sparse root model* — The sparse root zone model optimizes the sharing of objects by only installing a subset of the root packages and using a read-only loopback file system to gain access to other files. In this model, the directories `/lib`, `/platform`, `/sbin`, and `/usr` are mounted by default as loopback file systems. The advantages of this model are improved performance due to efficient sharing of executables and shared libraries, and a much smaller disk footprint for the zone itself.

- *Whole root model* — The whole root zone model provides for maximum configurability by installing the required packages and any selected optional zones into the private file systems of the zone. The advantages of this model include the ability for zone administrators to customize their zone's file system layout and add arbitrary unbundled or third-party packages.

Solaris Zones provide the standard Solaris interfaces and application environment; they do not impose a new ABI or API. In general, applications do not need to be ported to Solaris Zones. However, applications running in non-global zones may need to be aware of non-global zone behavior, depending on the Solaris interfaces they use. In particular:

- All processes running in a zone have a reduced set of privileges, which is a subset of the privileges available in the global zone. This set of privileges is available to the root user. Non-root users of a zone have a subset of those privileges. By default, non-global zone non-root users have privileges that are the "logical AND" of the privileges available to non-root users in the global zone and the privileges available to that zone.

  Processes that require a privilege not available in a non-global zone can fail to run, or in a few cases fail to achieve full performance.

- Administrators can modify the privileges that a zone has, reducing or expanding the set. This provides the ability to enhance security by removing privileges not needed by applications running in that zone, or to give a zone a non-default privilege in order to improve the functionality or performance of an application. The privilege `proc_lock_memory`, required to use Dynamic Intimate Shared Memory (DISM), is now in the default privileges set of zones.
- Each non-global zone may have its own logical network and loopback interface. Bindings between upper-layer streams and logical interfaces are restricted such that a stream may only establish bindings to logical interfaces in the same zone. Likewise, packets from a logical interface can only be passed to upper-layer streams in the same zone as the logical interface.
- Each zone can be configured with exclusive-IP privileges which allow it to have its own IP resources. This gives full functionality and independence from the global zone's IP. Specifically, an exclusive-IP zone can manage its own network interfaces, routing table, IPQoS configuration, and IP Filter rules.
- Non-global zones have access to a restricted set of devices. In general, devices are shared resources in a system. Therefore, restrictions within zones are put in place so that security is not compromised.
- Two categories of iSCSI storage are supported with zones. A zone can be installed into a directory which is mounted in the global zone and is backed by iSCSI storage. (See a description of *Network-Attached Containers* by Jeff Victor, documented at `http://blogs.sun.com/jeffv/date/20080408`.) Alternatively, iSCSI storage can be mounted into the global zone, and a directory from the file system can be loopback mounted into a zone.

Solaris Resource Manager

By default, the Solaris OS provides all workloads running on the system equal access to all system resources. This default behavior of the Solaris OS can be modified by Solaris Resource Manager, which provides a way to control resource usage.

Solaris Resource Manager (SRM) provides the following functionality:

- A method to classify a workload, so the system knows which processes belong to a given workload.
- The ability to measure the workload to assess how much of the system resources the workload is actually using.
- The ability to control the workloads so they do not interfere with one another and also get the required system resources to meet predefined service-level agreements.

SRM provides three types of workload control mechanisms:

- The *constraint mechanism*, which allows the Solaris system administrator to limit the resources a workload is allowed to consume.
- The *scheduling mechanism*, which refers to the allocation decisions that accommodate the resource demands of all the different workloads in an under-committed or over-committed scenario.
- The *partitioning mechanism*, which ensures that pre-defined system resources are assigned to a given workload.

Workload Identification

Solaris Resource Manager uses two levels of granularity, *projects* and *tasks*, to identify a workload:

- *Projects*
  Projects are a facility that allow the identification and separation of workloads. A workload can be composed of several applications and processes belonging to different groups and users. The identification mechanism provided by projects serves as a *tag* for all the processes of a workload. This identifier can be shared across multiple machines through the project name service database. The location of this database can be in files, NIS, or LDAP, depending on the definition of projects database source in the `/etc/nsswitch.conf` file. Attributes assigned to the projects are used by the resource control mechanism to provide a resource administration context on a per-project basis.

- *Tasks*
  Tasks provide a second level of granularity in identifying a workload. A task collects a group of processes into a manageable entity that represents a workload component. Each login creates a new task that belongs to the project, and all the processes started during that login session belong to the task. The concept of projects and tasks has been incorporated in several administrative commands such as `ps`, `pgrep`, `pkill`, `prstat` and `cron`.

Resource Controls

It is possible to place bounds on resource usage, to control the resource usage of a workload. These bounds can be used to prevent a workload from over-consuming a particular resource and interfering with other workloads. The Solaris Resource Manager provides a resource control facility to implement constraints on resource usage.

Each resource control is defined by the following three values:

- Privilege level
- Threshold value
- Action that is associated with the particular threshold

The privilege level indicates the privilege needed to modify the resource. It must be one of the following three types:

- *Basic*, which can be modified by the owner of the calling process
- *Privileged*, which can be modified only by privileged (superuser) callers
- *System*, which is fixed for the duration of the operating system instance

The threshold value on a resource control constitutes an enforcement point where actions can be triggered. The specified action is performed when a particular threshold is reached. Global actions apply to resource control values for every resource control on the system. Local action is taken on a process that attempts to exceed the control value.

There are three types of local actions:

- *None*—No action is taken on resource requests for an amount that is greater than the threshold.
- *Deny*—Deny resource requests for an amount that is greater than the threshold.
- *Signal*—Enable a global signal message action when the resource control is exceeded.

For example, `task.max-lwp=(privileged, 10, deny)` would tell the resource control facility to deny more than 10 lightweight processes to any process in that task.

CPU and Memory Management

SRM enables the end user to control the available CPU resources and physical memory consumption of different workloads on a system by providing *Fair Share Scheduler (FSS)*, *Resource Capping Daemon, CPU Caps* and *Dedicated CPUs* facilities.

- *Fair Share Scheduler*
  The default scheduler in the Solaris OS provides every process equal access to CPU resources. However, when multiple workloads are running on the same system one workload can monopolize CPU resources. Fair Share Scheduler provides a mechanism to prioritize access to CPU resources based on the importance of the workload.

With FSS the importance of a workload is expressed by the number of shares the system administrator allocates to the project representing the workload. Shares define the relative importance of projects with respect to other projects. If project A is deemed twice as important as Project B, project A should be assigned twice as many shares as project B.

It is important to note that FSS only limits CPU usage if there is competition for CPU resources. If there is only one active project on the system, it can use 100% of the system CPUs resources, regardless of the number of shares assigned to it.

- *Resource Capping Daemon*
  The resource capping daemon (`rcapd`) can be used to regulate the amount of physical memory that is consumed by projects with resource caps defined. The `rcapd` daemon repeatedly samples the memory utilization of projects that are configured with physical memory caps. The sampling interval is specified by the administrator. When the system's physical memory utilization *soft cap* exceeds the threshold for cap enforcement and other conditions are met, the daemon takes action to reduce the memory consumption of projects with memory caps to levels at or below the caps.

  Virtual memory (swap space) can also be capped. This is a *hard cap*. In a Container which has a swap cap, an attempt by a process to allocate more virtual memory than is allowed will fail.

  With the Oracle Database it may be not appropriate to set the physical memory and swap limitation since the swapping of Oracle processes or System Global Area (SGA) is not desirable.

  The third new memory cap is locked memory. This is the amount of physical memory that a Container can lock down, or prevent from being paged out. By default a Container now has the `proc_lock_memory` privilege, so it is wise to set this cap for all Containers.

- *CPU Caps*
  CPU caps provide absolute fine-grained limits on the amount of CPU resources that can be consumed by a project or a zone. CPU caps are provided as a `zonecfg` resource, and as project and zone-wide resource controls.

  Administrators can use this feature to control upper limit of CPU usage by each zone. This is in contrast to FSS, which sets the minimum guaranteed portion of CPU time to a given zone if there is a competition for CPU.

  For example: consider the following commands to set a CPU cap for a zone

```
zonecfg:myzone> add capped-cpu
zonecfg:myzone:capped-cpu> set ncpus=3.75
zonecfg:myzone:capped-cpu> end
```

The `ncpus` parameter indicates the percentage of a single CPU that can be used by all user threads in a zone, expressed as a fraction (for example, .75) or a mixed number (whole number and fraction, for example, 3.25). An `ncpu` value of 1 means 100% of a CPU, a value of 3.25 means 325%, .75 mean 75%, and so forth. When projects within a capped zone have their own caps, the minimum value takes precedence.

- *Dedicated CPUs*
  Administrators can use this feature to assign CPUs to zones dynamically within specified minimum and maximum limits per each zone. This eliminates the need to create CPU pools and assign pools to zones, leading to better resource usage and much simple administration.

  For example, consider the following commands to set dedicated CPUs for a zone:

```
zonecfg:myzone> add dedicated-cpu
zonecfg:myzone:dedicated-cpu> set ncpus=8-12
zonecfg:myzone:dedicated-cpu> end
```

  With this example, when the zone boots the system creates a temporary dedicated pool for this zone by taking CPUs from the global zone. If the zone will need more CPUs and there will be available CPUs, then the system will assign them to the zone within specified limits.

More examples of CPU and Memory Management in Solaris Containers are included in “New Zones Features” by Jeff Victor [10].

Note – Oracle users should consult with the current status of dynamic cpu_count changes supported by Oracle software. At the time of publication, some bugs prevented Oracle software from working correctly with dynamic cpu_count changes.

Oracle License Model for Solaris Containers

Oracle now recognizes capped Solaris 10 Containers as licensable entities, known as hard partitions. Oracle customers running an Oracle database in a Solaris 10 OS environment can now license only the CPUs or cores that are in a capped Solaris container.

Oracle licensing policy defines hard partitioning as “a physical subset of a server that acts like a self-contained server” (for more details see reference [2]). The following example (Figure 2) illustrates how an 8-processor system can be partitioned into a 3-processor sub-system using Solaris Containers technology in the Solaris 10 OS.

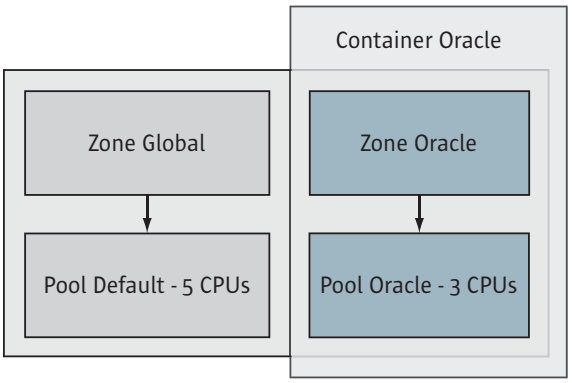

Figure 2. Example of a Container for Oracle in the Solaris 10 OS.

To create a Solaris 10 container that fits the licensing requirements set by Oracle, the Solaris system administrator needs to create a resource pool with the desired number of CPUs or cores and bind a zone to this resource pool. Alternatively, the administrator may set up a zone to use dynamic pool with specified CPU maximum limit. The license is driven by the number of CPUs or cores in this pool.

Creating a Solaris 10 Container

This section provides instructions for creating a Solaris 10 container appropriate for installing and running an Oracle database. These instructions have been followed in the sample scripts documented in Appendix A, “Scripts to Create a Solaris Container” on page 22, which provide a convenient way of creating such containers.

Requirements

1. Ensure that the file system in which the root directory for the container will be placed has at least 6 GB of physical disk space. This disk space is required to create the container and install Oracle database binaries. This example uses a sparse root zone.
2. Identify the physical interface that will be used to bring up the virtual interface for the container. Examples of common interfaces are `ce0`, `bge0` and `hme0`.To find the physical interfaces available in the global container execute the command:

```
# /usr/sbin/ifconfig -a
```

   Alternatively, to configure exclusive-IP for a zone, consider using an interface that is listed among the output of the following `dladm` command, but is not listed by the output of the `ifconfig -a` command:

```
# /usr/sbin/dladm show-link
```

   Examples may include `hme1` or `e1000g1`.

3. Obtain an IP address and a hostname for the container.
   a. If exclusive-IP is *not* being used, this IP address must be in the same subnet as the IP assigned to the physical interface selected in the previous step.
   b. Or, if exclusive-IP is being used, then select any IP address (this address is not required to belong to the same subnet as global zone's IP).
4. If exclusive-IP is not being used, ensure that the netmask for the IP address of the container can be resolved in the global zone according to the databases used in the `/etc/nsswitch.conf` file. If this is not the case, update the file `/etc/netmasks` in the global container with the netmask desired for the subnet to which the IP address belongs.
5. If partitioning the CPUs into pools or using dedicated-CPUs, determine the quantity of CPUs to be reserved for the container. To find the quantity of CPUs available in the default pool, execute the command `poolstat`. The default pool will indicate the number of CPUs available. Keep in mind that the default pool must always have at least one CPU. If your system is not configured with pools then use `psrinfo` to find all available CPUs.

Enabling Resource Pools

Resource pools and dynamic resource pools have been integrated into the Solaris service management facility (SMF). Dynamic resource pools are enabled separately of the resource pools service.

The fault management resource identifier (FMRI) for the dynamic resource pools service is `svc:/system/pools/dynamic`. The FMRI for the resource pools service is `svc:/system/pools`.

To enable the pool facility, use the following commands:

```
# svcadm enable svc:/system/pools:default
# svcadm enable svc:/system/pools/dynamic:default
```

To check the status, use the following command:

```
# svcs -a | grep pool
STATE          STIME    FMRI
online         11:21:40 svc:/system/pools:default
online         11:21:45 svc:/system/pools/dynamic:default
```

Creating a Non-Global Zone

Once the resource pools services are available, a non-global zone can be created and bound to it. For the purposes of installing and running an Oracle database, the non-global zone can have either a whole root model or a sparse root model. Unless it conflicts with specific requirements, it is recommended to use the sparse root model as this creates a much smaller disk footprint for the zone.

A non-global Solaris zone can be created as follows:

1. As root user, create a directory where the root directory of the non-global zone will be placed (for example, `/zones/myzone`) and set the access permissions to 700. The name of this directory should match the name of the zone (`myzone`, in this example). This directory can be also on ZFS filesystem (for example `rpool/myzone` or `rpool/zones/myzone`).
2. Unless special instructions are added, the directory `/usr` in the container will be a loopback file system (lofs). This means that the container will mount in read-only mode `/usr` from the global container into `/usr` of its own file system tree. By default, the Oracle installer requires root to create files in the `/usr/local` directory. Since `/usr` will be read-only in the container, this example creates a special mount point in `/usr/local` to allow root to create such files in the container. Check if the directory `/usr/local` exists in the global container, and if it is not present, create it.
3. Create a directory in the global container to mount `/usr/local` in the container. The recommendation is to create it in `/opt/`*`ZONE_NAME`*`/local`.
4. Use the `setenv.sh` file in Appendix A (“The setenv.sh File” on page 24) to create a settings file to create the zone and bind it to the resources. Set the values of the variables `ZONE_DIR`, `ZONE_NAME`, `IP_TYPE`, `NET_IP`, and `NET_PHYSICAL` with appropriate values. In the file `zone_cmd_template.txt`, the command `create` is used to create a sparse root. Replacing this command with `create -b` would create a whole root. The zone will be created with a dynamic CPU pool within the `NUM_CPUS_MIN` and `NUM_CPUS_MAX` limits. If you want to have exclusive IP in your new zone then set `IP_TYPE=EXCLUSIVE`. The scheduling class and `max-shm-memory` for the zone can also be set in this file.
5. Create the zone by executing the following command as root (see “The create_container.sh Script” on page 26 of Appendix A):

```
# sh ./create_container.sh
```

6. Finish the configuration of the container with the following command:

```
# zlogin -C ZONE_NAME
```

Special Considerations

This section contains special considerations when running an Oracle database inside a Solaris container.

Devices in Containers

To guarantee that security and isolation are not compromised, certain restrictions regarding devices are placed on non-global zones:

- By default, only a restricted set of devices (which consist primarily of pseudo devices) such as `/dev/null`, `/dev/zero`, `/dev/poll`, `/dev/random`, and `/dev/tcp`, are accessible in the non-global zone.
- Devices that expose system data like `dtrace`, `kmem`, and `ksyms` are not available in non-global zones.
- By default, physical devices are also not accessible by non-global zones.

The global zone administrator can make physical devices available to non-global zones. It is the administrator's responsibility to ensure that the security of the system is not compromised by doing so, mainly for two reasons:

- Placing a physical device into more than one zone can create a covert channel between zones.
- Global zone applications that use such a device risk the possibility of compromised data or data corruption by a non-global zone.

The global zone administrator can use the `add device` sub-command of `zonecfg` to include additional devices in non-global zone. For example to add the block device `/dev/dsk/c1t1d0s0` to the non-global zone my-zone, the administrator executes the following commands:

```
# zonecfg -z my-zone
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/dsk/c1t1d0s0
zonecfg:my-zone:device> end
zonecfg:my-zone> exit
# zoneadm -z my-zone reboot
```

All slices of `/dev/dsk/c1t1d0` could be added to the non-global zone by using `/dev/dsk/c1t1d0*` in the `match` command. This same procedure can be used for character devices (also known as raw devices) or any other kind of device.

If it is planned to install an Oracle database in a non-global zone by using Oracle installation CDs, the CD-ROM device must be made visible to the non-global zone. One recommended method is to loopback mount the `/cdrom` directory from the global zone. An alternative method of exporting the physical device from the global zone to the non-global zone is discouraged, as `/cdrom` is nearly always a system-wide shared device (and, furthermore, exporting the cdrom device requires stopping/disabling the Volume Management daemon `vold` in the global zone). For details about how to gain access to the CD-ROM device from a non-global zone, see reference [8].

UFS File Systems in Solaris Containers

Each zone has its own file system hierarchy, rooted at a directory known as *zone root*. Processes in a zone can access only files in the part of the file system hierarchy that is located under the zone root.

Three different ways of mounting a UFS file system in a non-global zone are described in the following examples:

- *Method 1: Mount using loopback file system (lofs)*
  Create a file system in a global zone and mount it in a non-global zone as a loopback file system (lofs). This method is used to share a file system between global and non-global zones.

  1. Log in as global zone administrator.
  2. Create a file system in global zone:

```
global# newfs /dev/rdsk/c1t0d0s0
```

  3. Mount the file system in the global zone:

```
global# mount /dev/dsk/c1t0d0s0 /mystuff
```

  4. Add the file system of type `lofs` to the non-global zone:

```
global# zonecfg -z my-zone
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/usr/mystuff
zonecfg:my-zone:fs> set special=/mystuff
zonecfg:my-zone:fs> set type=lofs
zonecfg:my-zone:fs> end
```

- *Method 2: Mount file system as UFS*
  Create a file system in the global zone and mount it to the non-global zone as UFS.

  1. Log in as global zone administrator.
  2. Create a file system in the global zone:

```
global# newfs /dev/rdsk/c1t0d0s0
```

  3. Add the file system of type `ufs` to the non-global zone:

```
global# zonecfg -z my-zone
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/usr/mystuff
zonecfg:my-zone:fs> set special=/dev/dsk/c1t0d0s0
zonecfg:my-zone:fs> set raw=/dev/rdsk/c1t0d0s0
zonecfg:my-zone:fs> set type=ufs
zonecfg:my-zone:fs> end
```

- *Method 3: Export a device from a global zone*
  Export a device from a global zone to a non-global zone and mount it from the non-global zone. Using this method, the file system that is created will not be shared between zones.

  1. Log in as global zone administrator.

2. Export a raw device to the non-global zone:

```
global# zonecfg -z my-zone
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/rdsk/c1t0d0s0
zonecfg:my-zone:device> end
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/dsk/c1t0d0s0
zonecfg:my-zone:device> end
```

3. Log in as root in non-global zone.

4. Create a file system in the non-global zone:

```
my-zone# my-zone# newfs /dev/rdsk/c1t0d0s0
```

5. Mount the file system in the non-global zone:

```
my-zone# mount /dev/dsk/c1t0d0s0 /usr/mystuff
```

ZFS File Systems in Containers

The Solaris Zones partitioning technology supports Solaris ZFS components, such as adding Solaris ZFS file systems and storage pools into a zone. The following sections describe how to add a ZFS file system and delegate a dataset to a non-global zone. For more detailed information, see the Solaris ZFS Administration Guide [11].

Adding a Single ZFS File System

This example illustrates adding a ZFS file system to a non-global zone.

1. From the global zone, make sure the mount point is set to legacy:

```
global# zfs set mountpoint=legacy tank/home
```

2. From the global zone, make sure that mount point is set to legacy. Use the following command to check the mount point, and confirm it is set to legacy:

```
global# zfs get mountpoint tank/home
```

3. Use the following commands to add a ZFS file system to the zone:

```
global# zonecfg -z myzone
zonecfg:myzone> add fs
zonecfg:myzone:fs> set type=zfs
zonecfg:myzone:fs> set dir=/export/share
zonecfg:myzone:fs> set special=tank/home
zonecfg:myzone:fs> end
```

This syntax adds the ZFS file system `tank/home` to the `myzone` zone, mounted at `/export/share`. With this configuration, the zone administrator can create and destroy files within the file system. The file system cannot be remounted in a different location, nor can the zone administrator change properties on the file system (such as compression, read-only, and so on). The global zone administrator is responsible for setting and controlling properties of the file system.

Delegating Complete ZFS Dataset

If the primary goal is to delegate the administration of storage to a zone, a complete ZFS dataset can be delegated to a non-global zone. In the following example, a ZFS file system is delegated to a non-global zone by the global zone administrator.

1. First, create a ZFS file system in a global zone:

```
# zfs create distr/vol1
```

2. Second, add it to the zone as a dataset:

```
# zonecfg -z myzone
zonecfg:myzone> add dataset
zonecfg:myzone:dataset> set name=distr/vol1
zonecfg:myzone:dataset> end
zonecfg:myzone> verify
zonecfg:myzone> commit
zonecfg:myzone> end
```

3. Third, restart the zone to get the ZFS file system in it. After restarting, the ZFS file system can be found inside a zone:

```
# zfs list
NAME          USED  AVAIL  REFER  MOUNTPOINT
distr        48,9G  18,3G  30,5K  /distr
distr/vol1   24,5K  18,3G  24,5K  /distr/vol1
```

Because the directory `distr/vol1` was imported in this example, it cannot be managed on the `distr` level. But, additional file systems can be created beneath the `distr/vol1` directory, as seen in the following example:

```
# zfs create distr/vol2
cannot create 'distr/vol2': permission denied
# zfs create distr/vol1/vol2
```

Unlike the previous example of adding a single file system, this example causes the ZFS file system `/distr/vol1` to be visible within the zone `myzone`. The zone administrator can set file system properties, create children, take snapshot, create clones, and otherwise control the entire ZFS file system hierarchy. This allows the administration of a ZFS file system (and any subordinate file systems created in the zone) to be delegated to a non-global zone.

After the ZFS dataset is delegated to a zone, the dataset property `zoned=on` will be set. This means that most of the ZFS properties, including `mountpoint`, cannot be used in global zone anymore. Effectively, it will look like `mounpoint=legacy` even if the `mountpoint` is set to any specific value.

Limiting ZFS cache

The ZFS cache limit in global zone can be set by specifying the `zfs_arc_max` parameter in the `/etc/system` file. For example, the following example sets the ZFS cache limit to 8 GB (`0x200000000` in hexadecimal):

```
set zfs:zfs_arc_max = 0x200000000
```

While ZFS sets its own limits on cache used, administrators can choose to monitor ZFS cache usage and set explicit limits. The blog article "Monitoring ZFS Statistic on the system loaded with Oracle database," available online at `http://blogs.sun.com/pomah/entry/monitoring_zfs_statistic_con_t`, contains a script that can be used to monitor cache usage, and the cache limit should be planned to be below the value of the physical memory size less the sum of all SGAs for all Oracle instances.

Note – The "ZFS Evil Tuning Guide" provides additional information on limiting the ZFS cache and other ZFS tuning best practices:
`http://www.solarisinternals.com/wiki/index.php/ZFS_Evil_Tuning_Guide`

System V Resource Controls for Zones

The *Fair Share Scheduler* (FSS) can be used to control the allocation of available CPU resources among zones, based on their importance.

```
zonecfg:myzone> set scheduling-class=FSS
```

If it is planned to set shares for each zone, they can be specified using the `set cpu-shares` command. For example:

```
zonecfg:myzone> set cpu-shares=10
```

The more shares that are assigned, the more CPU cycles this zone's processes will get if there is competition for CPU.

Additionally, the maximum shared memory segment size needed can be set for an Oracle instance. For example:

```
zonecfg:myzone> set max-shm-memory=4G
```

It is recommended to set FSS globally (unless specific configuration requirements make it undesirable to do so) and then reboot the global zone:

```
global# dispadmin -d FSS
```

In this case, it is not required to explicitly set the scheduling class to FSS for each zone (`set scheduling-class=FSS`) since it is inherited from global zone.

Solaris Dynamic Intimate Shared Memory (DISM)

DISM provides shared memory that is dynamically resizable. A process that makes use of a DISM segment can lock and unlock parts of a memory segment while the process is running. By doing so, the application can dynamically adjust to the addition of physical memory to the system or prepare for the removal of it.

By default, Oracle uses intimate shared memory (ISM) instead of standard System V shared memory on the Solaris OS. When a shared memory segment is made into an ISM segment, it is mapped using large pages and the memory for the segment is locked (that is, it cannot be paged out). This greatly reduces the overhead due to process context switches, which improves Oracle's performance linearity under load. For more details about Oracle and DISM, see reference [5].

ISM certainly has benefits over standard System V shared memory. However, its disadvantage is that ISM segments cannot be resized. To change the size of an ISM database buffer cache, the database must be shut down and restarted. DISM overcomes this limitation as it provides shared memory that is dynamically resizable. DISM has been supported in Oracle databases since Oracle 9i. Oracle uses DISM instead of ISM if the maximum SGA size set by the `sga_max_size` parameter in `init.ora` is larger than the sum of its components (that is, `db_cache_size`, `shared_pool_size`, and other smaller SGA components).

In ISM, the kernel locks and unlocks memory pages. However, in DISM the locking and unlocking of memory pages is done by the Oracle process `ora_dism_`*`$ORACLE_SID`*. In order to lock memory, a process needs the `proc_lock_memory` privilege. This privilege is now available in a Solaris 10 non-global zone by default.

The following example shows configuring Oracle for DISM:

```
SQL> alter system set sga_max_size=1000M scope=spfile;
SQL> alter system set sga_target=600M scope=spfile;
SQL> shutdown immediate
SQL> startup
$ ps -ef|grep ism
root 28976  5865   0 10:54:00 ?           0:17 ora_dism_mydb
```

Dedicated CPU

With the introduction of the dedicated CPU feature, system administrators now have a much easier and also more effective way of managing pools of CPUs. In reality, system administrators do not even have to care about creating CPU pools. Of course, they must know how many CPUs are available and how they are used. With the dedicated CPU feature, administrators can assign CPUs right when they are creating a new zone and they can manage assigned CPUs by changing zone properties.

For example, the following commands specifies a CPU range for use by the zone `myzone`:

```
zonecfg:myzone> add dedicated-cpu
zonecfg:myzone:dedicated-cpu> set ncpus=8-12
zonecfg:myzone:dedicated-cpu> end
```

IP Instances: LAN and VLAN Separation for Non-Global Zones

IP networking can now be configured in two different ways, depending on whether the zone is assigned an exclusive IP instance or shares the IP layer configuration and state with the global zone. IP types are configured by using the `zonecfg` command.

The shared-IP type is the default. These zones connect to the same VLANs or same LANs as the global zone and share the IP layer.

Full IP-level functionality is available in an exclusive-IP zone. If a zone must be isolated at the IP layer on the network, then the zone can have an exclusive IP. The exclusive-IP zone can be used to consolidate applications that must communicate on different subnets that are on different VLANs or different LANs.

Network interfaces should not be used by any other zone, including the global zone.

The following example configures the zone `myzone` for exclusive IP usage using the physical interface `e1000g1`:

```
zonecfg:myzone> set ip-type=exclusive
zonecfg:myzone> add net
zonecfg:myzone:net> set physical=e1000g1
zonecfg:myzone:net> end
```

Moving Solaris Containers

Solaris Containers can be moved, or migrated, from one server to another. This capability allows administrators to rapidly provision zones and relocate workloads as needed to meet changing requirements.

In order to move a Solaris Container, the zone must be halted. In addition, both the old and new systems must have compatible levels of Solaris patches, and the location of devices must be the same if these devices were used in the configuration.

Basic steps include the following:

1. Create the container on the first system.
2. Detach the container.
3. Create an archive.
4. Move the archive to a new system, and unpack the archive.
5. Attach the container to the new system.

The procedure for moving a Solaris Container depends, in part, on the type of file system used to hold the container's private files. For example, the container can reside on a root file system; the container can reside on a ZFS file system; or, the container can reside in a UFS file system built on a Solaris Volume Manager metadevice. For detailed descriptions on procedures for these three scenarios, see [9].

Volume Management

Because various options are available for volume management, including third-party volume managers products, usability in containers cannot be completely generalized. As of this writing, volume managers cannot be installed or managed from non-global zones. Currently, the recommended approach is to install and manage volume manager software from a system's global zone. Once the devices, file systems, and volumes are created in the global zone, they can be made available to the non-global zones by using `zonecfg` subcommands.

For example, Solaris Volume Manager (SVM) should be installed and managed from global zones. Once the storage has been configured in the desired way from the global zone, the metadevices can then be made available to the non-global zones or used through mounted file systems.

Note – For the latest status of support and any published work-arounds for technical issues related to third-party volume management software, it is advised to consult with the third-party support Web sites.

CPU Visibility

Users can expect a virtualized view of the system in a Solaris container with respect to CPU visibility when the zone is bound to a resource pool. In these cases the zone will only see those CPUs associated with the resource pool it is bound to.

By default a container is bound the pool `pool_default`, which is the same set of processors that the global zone's processes use.

The Oracle database application mainly calls `pset_info(2)` and `sysconf(3c)` with the `_SC_NPROCESSORS_ONLN` argument to procure the number of CPUs it has available. Based on this number, Oracle will size internal resources and create threads for different purposes (for example, parallel queries and number of readers). These calls will return the expected value (that is, the number of CPUs in the resource pool) if the zone is bound to a resource pool with a pset associated with it. Table 1 shows the interfaces that have been modified in the Solaris OS and that will return the expected value in this scenario.

Table 1. List of CPU-Related Solaris 10 Interfaces That Are Container-Aware

INTERFACE	TYPE
p_online(2)	System Call
processor_bind(2)	System Call
processor_info(2)	System Call
pset_list(2)	System Call
pset_info(2)	System Call
pset_getattr(2)	System Call
pset_getloadavg(3c)	System Call
getloadavg(3c)	System Call
sysconf(3c)	System Call
p_online(2)	System Call
_SC_NPROCESSORS_CONF	sysconf(3c) arg
_SC_NPROCESSORS_ONLN	sysconf(3c) arg
pbind(1M)	Command
psrset(1M)	Command
psrinfo(1M)	Command
mpstat(1M)	Command
vmstat(1M)	Command
iostat(1M)	Command
sar(1M)	Command

In addition to these interfaces, certain kernel statistics (`kstats`) are used commonly by tools such as `psrinfo(1M)` and `mpstat(1M)` to retrieve information about the system. All consumers of these `kstats` will only see information for a `pset` in the pool bound to the zone.

Summary

Solaris Containers provide a very flexible and secure method of managing multiple applications on a single Solaris OS instance. Solaris Containers use Solaris Zones software partitioning technology to virtualize the operating system and provide isolated and secure runtime environments for applications. Solaris Resource Manager can be used to control resource usage, such as capping memory and CPU usage, helping to ensure workloads get required system resources. By utilizing Solaris containers, multiple applications, or even multiple instances of the same application, can securely coexist on a single system, providing potential server consolidation savings.

Oracle 9i R2 and 10g R2 databases have been certified to run in a Solaris container. This paper provides step-by-step directions for creating a non-global zone in a Solaris container that is appropriate for running a non-RAC Oracle database. In addition, it describes special considerations that apply when running an Oracle database within a Solaris container.

About the Authors

Ritu Kamboj is a Staff Engineer in ISV Engineering's Open Source Team at Sun Microsystems. She has over 12 years of experience in software development with expertise on database design, performance, and high availability. She has worked extensively on Sybase, Oracle, and MySQL databases. Recently her primary focus has been making MySQL run best on the Solaris platform.

Roman Ivanov joined Sun in January 2006. He is working in the ISV Engineering department helping Independent Software Vendors adopt Sun's technology and improve performance on Sun's hardware. His blog is available online at `http://blogs.sun.com/pomah/`.

Acknowledgements

The author would like to thank Alain Chéreau, Jeff Victor, and Fernando Castano for their contributions to this article.

References

[1] *Consolidating Applications with Solaris 10 Containers,* Sun Microsystems, 2004.
`http://www.sun.com/datacenter/consolidation/solaris10_whitepaper.pdf`

[2] *Solaris 10 System Administrator Collection — System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*, Sun Microsystems, 2005.
`http://docs.sun.com/app/docs/doc/817-1592`

[3] Lageman, Menno. “Solaris Containers—What They Are and How to Use Them,” *Sun BluePrints OnLine*, 2005.
`http://www.sun.com/blueprints/0505/819-2679.html`

[4] Solaris Zones section on BigAdmin System Administration Portal, Sun Microsystems.
`http://www.sun.com/bigadmin/content/zones`

[5] Vanden Meersch, Erik and Hens, Kristien. "Dynamic Reconfiguration and Oracle 9i Dynamically Resizable SGA," *Sun BluePrints OnLine*, 2004.
http://www.sun.com/blueprints/0104/817-5209.pdf

[6] Oracle Pricing with Solaris 10 Containers.
http://www.sun.com/third-party/global/oracle/consolidation/solaris10.html

[7] Oracle's Partitioning document
http://www.oracle.com/corporate/pricing/partitioning.pdf

[8] Lovvik, Paul and Balenzano, Joseph. "Bringing Your Application Into the Zone," Sun Developer Network article, 2005.
http://developers.sun.com/solaris/articles/application_in_zone.html

[9] Victor, Jeff. "How to Migrate a Container to Another System," Sun Microsystems,
http://www.sun.com/software/solaris/howtoguides/moving_containers.jsp

[10] Victor, Jeff. "New Zones Features," Sun Microsystems,
http://blogs.sun.com/JeffV/entry/new_zones_features

[11] *Solaris ZFS Administration Guide*, Sun Microsystems.
http://opensolaris.org/os/community/zfs/docs/zfsadmin.pdf

Ordering Sun Documents

The SunDocs℠ 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
Scripts to Create a Solaris Container

The scripts documented in this Appendix can be used to create a container appropriate for installing and running non-RAC instances of an Oracle database. These scripts do not represent the only way in which the container can be created. They are provided as sample code and should be modified to fit specific requirements and constraints.

In this example, a sparse root zone will be created with the root directory, IP address, and physical interface provided by the user. Users may choose to have exclusive IP and be independent from global zone. A special mount point for `/usr/local` will be created in `/opt/`*`zone_name`*`/local` to facilitate the Oracle installation, since `/usr/local` is the default directory for the installation of some of the Oracle utilities. To use these scripts, save all the files in the same directory and follow these steps:

1. Edit the file `setenv.sh` with appropriate values for the following variables:
   - `ZONE_NAME`: hostname for the zone
   - `ZONE_DIR`: directory for the root directory of the zone or ZFS pool name
   - `NET_IP` : IP address for the zone, it is not necessary to set if using exclusive IP
   - `IP_TYPE`: You may wish to have exclusive IP
   - `NET_PHYSICAL` : physical interface in which the virtual interface for the zone will be created
   - `NUM_CPUS_MAX` : maximum number of CPUs for the zone
   - `NUM_CPUS_MIN`: minimum number of CPUs for the zone
   - `SCHEDULING_CLASS=FSS`: To have FSS as default scheduling for the zone
   - `MAX_SHM_MEMORY=4GB`: To specify Maximum Shared memory segment
2. Issue the following command from the global zone:

```
# ./create_container.sh
```

3. Configure this container by executing the following command from global zone, substituting the correct `zone_name`:

```
# zlogin -C zone_name
```

The files composing these scripts are presented and explained next.

README.txt

This file describes how a container is created when these scripts are used. It also gives some tips about how to execute some common operations such as giving the zone access to raw devices or removing resource pools.

```
The scripts in this directory can be used to create a container
suitable for installing and running non-RAC instances of Oracle database.
These scripts do not represent the only way in which you can create an
appropriate container for Oracle; depending on your requirements and
constrains you can modify these scripts to fit your needs.

1) creating a container for Oracle
A sparse root zone will be created with the root directory, IP and
interface provided by the user.  A special mount point for /usr/local will
be created in /opt/<zone_name>/local to facilitate the oracle installation,
since /usr/local is the default directory for the installation of some of
the oracle utilities. To use these scripts follow these
steps:

a) edit the file setenv.sh with appropriate values for:
    - ZONE_NAME: hostname for the zone
    - ZONE_DIR: directory for the root directory of the zone
    - NET_IP:  IP for the zone
    - NET_PHYSICAL: physical interface in which the virtual interface for
                    the zone will be created
    - NUM_CPUS_MAX: maximum number of CPUs for the zone
    - NUM_CPUS_MIN: minimum number of CPUs for the zone
    - IP_TYPE: You may wish to have exclusive IP
    - SCHEDULING_CLASS=FSS: To have FSS as default scheduling for the zone
    - MAX_SHM_MEMORY=4GB: To specify Maximum Shared memory segment

b) from the global container run ./create_container.sh

c) Once the container has been created run "zlogin -C <zone_name>"
     from the global container to finish configuring the zone.

2) giving your container access to raw devices
If you need to give your container access to a raw device
follow this example once the container has been created
(these commands must be issued from the global container):

zonecfg -z my-zone
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/rdsk/c3t40d0s0
zonecfg:my-zone:device> end
zonecfg:my-zone> exit
zonecfg -z my-zone halt
zonecfg -z my-zone boot
```

```
3) giving your container access to a file system
If you need to give your container access to a file system
created in the global container follow this example once the non-global
container has been created:

global# newfs /dev/rdsk/c1t0d0s0
global# zonecfg -z my-zone
zoncfg:my-zone> add fs
zoncfg:my-zone> set dir=/usr/mystuff
zoncfg:my-zone> set special=/dev/dsk/c1t0d0s0
zoncfg:my-zone> set raw=/dev/rdsk/c1t0d0s0
zoncfg:my-zone> set type=ufs
zoncfg:my-zone> end
zonecfg -z my-zone halt
zonecfg -z my-zone boot

4) to uninstall and delete a previously created zone use these commands:
zoneadm -z $ZONE_NAME halt
zoneadm -z $ZONE_NAME uninstall -F
zonecfg -z $ZONE_NAME delete -F
```

The setenv.sh File

The `setenv.sh` file is used to define parameters used to create the Solaris container.

```
#!/usr/bin/sh

# host name for the zone
ZONE_NAME=myzone

# directory where to place root dir for the zone
# or ZFS pool
#ZONE_DIR=/zones
ZONE_DIR=rpool

#IP for the zone (make sure netmask can be resolved for this IP according to
# the databases defined in nsswitch.conf)
# or use Exclusive-IP with its own IP stack
#NET_IP=129.146.182.199
IP_TYPE=EXCLUSIVE

#interface used by the zone
NET_PHYSICAL=e1000g1

#min and max CPUs for the dynamic pool bound to the zone
NUM_CPUS_MIN=8
NUM_CPUS_MAX=12
SCHEDULING_CLASS=FSS
MAX_SHM_MEMORY=4G YSICAL=e1000g1

# do not make changes beyond this point
export ZONE_NAME ZONE_DIR NET_IP NET_PHYSICAL
export NUM_CPUS_MIN NUM_CPUS_MAX
export MOUNT_ZFS SCHEDULING_CLASS MAX_SHM_MEMORY IP_TYPE
```

The zone_cmd_template.txt File

The `zone_cmd_template.txt` file contains a template set of commands to create the zone. After replacing some strings by user-defined values, this file is used to create the zone.

```
create
set zonepath=MOUNTPOINT
set autoboot=true
SCHEDULING_CLASS
MAX_SHM_MEMORY
IP_TYPE
add net
NET_IP
set physical=NET_PHYSICAL
end
add fs
set dir=/usr/local
set special=/opt/ZONE_NAME/local
set type=lofs
end
MOUNT_ZFS
add dedicated-cpu
set ncpus=NUM_CPUS_MIN-NUM_CPUS_MAX
end
verify
commit
```

The create_zone_cmg.pl Script

This script uses the `perl` utility to create a command file that creates the zone. This script replaces the user-given parameters in the zone command template file. It is called by `create_container.sh`.

```
#!/usr/bin/perl
# Copyright (c) 2005,2008 Sun Microsystems, Inc. All Rights Reserved.
#
# SAMPLE CODE
# SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT
# THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESS
# OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS
# FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.
# SUN SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED
# BY LICENSEE AS A RESULT OF USING, MODIFYING OR
# DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.


while (<>){
    s/MOUNTPOINT/$ENV{'MOUNTPOINT'}/;
    s/NET_IP/$ENV{'NET_IP'}/;
    s/IP_TYPE/$ENV{'IP_TYPE'}/;
    s/MOUNT_ZFS/$ENV{'MOUNT_ZFS'}/;
    s/ZONE_NAME/$ENV{'ZONE_NAME'}/;
    s/NET_PHYSICAL/$ENV{'NET_PHYSICAL'}/;
    s/NUM_CPUS_MIN/$ENV{'NUM_CPUS_MIN'}/;
    s/NUM_CPUS_MAX/$ENV{'NUM_CPUS_MAX'}/;
    s/SCHEDULING_CLASS/$ENV{'SCHEDULING_CLASS'}/;
    s/MAX_SHM_MEMORY/$ENV{'MAX_SHM_MEMORY'}/;

    print;
}
```

The create_container.sh Script

This is the main script. It uses the parameters given in the `setenv.sh` file to create the container.

```
#!/usr/bin/ksh
# Copyright (c) 2005,2008 Sun Microsystems, Inc. All Rights Reserved.
#
# SAMPLE CODE
# SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT
# THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESS
# OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS
# FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.
# SUN SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED
# BY LICENSEE AS A RESULT OF USING, MODIFYING OR
# DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.

# script to create a container to run oracle RDBMS.
# to use this script follow the instructions in the README.txt file
# located in this directory.

. ./setenv.sh
```

```
#zone already exists?
zonecfg -z $ZONE_NAME info > /tmp/z.$$ 2>&1
cat /tmp/z.$$ | grep "No such zone" > /dev/null 2>&1

if [ $? -eq 1 ]
then
  echo "ERROR: zone $ZONE_NAME already exists. IF you want to remove it do:"
  echo "use zoneadm -z $ZONE_NAME halt"
  echo "use zoneadm -z $ZONE_NAME uninstall -F"
  echo "use zonecfg -z $ZONE_NAME delete -F"
  exit 1
fi

rm -rf /tmp/z.$$ > /dev/null 2>&1

# 1).............................  validate setenv.sh values

if [ `expr ${ZONE_DIR} : '\(.\)'` = '/' ]; then
   echo Using standard directory for the zone

   #zone path exists?
   if [ ! -d $ZONE_DIR/$ZONE_NAME ]
   then
     mkdir -p $ZONE_DIR/$ZONE_NAME
     if [ $? = 1 ]
     then
       echo ERROR: could not create root directory
       exit 1
     fi
   fi
   MOUNTPOINT=$ZONE_DIR/$ZONE_NAME

else
   echo Using ZFS for the zone

   MOUNTPOINT=`zfs get -H -o value mountpoint $ZONE_DIR/$ZONE_NAME
2>/dev/null`
   if [ x$MOUNTPOINT = 'x' ]; then
     echo zfs not exists. I will create $ZONE_DIR/$ZONE_NAME for you.
     zfs create $ZONE_DIR/$ZONE_NAME
     if [ $? -ne 0 ]; then
       echo Failed to create $ZONE_DIR/$ZONE_NAME
       exit 1
     fi
     MOUNTPOINT=`zfs get -H -o value mountpoint $ZONE_DIR/$ZONE_NAME
2>/dev/null`
   else
     echo $ZONE_DIR/$ZONE_NAME filesystem already exists
   fi

   if [ $MOUNTPOINT = 'legacy' ]; then
     echo Legacy mounted ZFS is not supported. Exiting.
     exit 1
   fi
fi

if [ x`ls $MOUNTPOINT` != 'x' ]; then
   echo ERROR:Directory $MOUNTPOINT is not empty. exiting.
   exit 1
fi

export MOUNTPOINT
chmod 700 $MOUNTPOINT
```

```
# Enabling Resource Pools
svcadm enable svc:/system/pools:default
svcadm enable svc:/system/pools/dynamic:default

#/usr/local directory exists?
if [ ! -d /usr/local ]
then
  mkdir /usr/local
fi

#special mnt point for /usr/local exists?
if [ ! -d /opt/$ZONE_NAME/local ]
then
  mkdir -p /opt/$ZONE_NAME/local
fi

# 2)............................... pool creation
# this section does not exist anymore since dynamic CPU pools is used

if [ `expr $NUM_CPUS_MIN \* 2` -le $NUM_CPUS_MAX ]; then
   echo WARNING: Having NUM_CPUS_MIN less then a half of NUM_CPUS_MAX is not
recommended by Oracle.
   echo Oracle bugs: 6309685 and 6309691 and ...
   echo You may adjust values before running Oracle or continue at your own
risk
fi

if [ -n "$SCHEDULING_CLASS" ]; then
   SCHEDULING_CLASS="set scheduling-class=\"$SCHEDULING_CLASS\""
fi

if [ -n "$MAX_SHM_MEMORY" ]; then
   MAX_SHM_MEMORY="set max-shm-memory=$MAX_SHM_MEMORY"
fi

if [ x"$IP_TYPE" = "xEXCLUSIVE" ]; then
   NET_IP=""
   IP_TYPE="set ip-type=exclusive"
else
   NET_IP="set address=$NET_IP"
fi

# 3)............................... mounting zfs inside a zone

MOUNT_ZFS=''

if [ -n "$MOUNT_ZFS" ]; then
   echo WARNING: mounting ZFS before the zone is started is not supported
   echo ignoring MOUNT_ZFS settings
   echo
#  MOUNT_ZFS=''

   TMP_MOUNT=''
   for z in $MOUNT_ZFS; do
     echo Changing mountpoint to legacy for $z
     zfs set mountpoint=legacy `echo $z|awk -F= '{print $1}'`
     TMP_MOUNT=${TMP_MOUNT}`echo $z|awk -F= '{print "\nadd fs\nset
type=zfs\nset dir="$2"\nset special="$1"\nend\n"}'`
   done
   MOUNT_ZFS="$TMP_MOUNT"
fi

# 4)...............................  zone creation
```

```
perl ./create_zone_cmd.pl < zone_cmd_template.txt > /tmp/zone_commands.txt
zonecfg -z $ZONE_NAME -f /tmp/zone_commands.txt
echo $ZONE_NAME was configured with this information:
echo ------------------------------------------------------
zonecfg -z $ZONE_NAME info

echo ------------------------------------------------------
zoneadm -z $ZONE_NAME install
zoneadm -z $ZONE_NAME boot

echo "to finish configuring your container please run: zlogin -C
$ZONE_NAME"
```

Appendix B
Setting System V IPC Kernel Parameters

Prior to the Solaris 10 OS, the System V IPC resources, consisting primarily of shared memory, message queues, and semaphores, were set in the `/etc/system` file. This implementation had the following shortcomings:

- Relying on `/etc/system` as an administrative mechanism meant reconfiguration required a reboot.
- A simple typo in setting the parameter in `/etc/system` could lead to hard-to-track configuration errors.
- The algorithms used by the traditional implementation assumed statically-sized data structures.
- There was no way to allocate additional resources to one user without allowing all users those resources. Since the amount of resources was always fixed, one user could have trivially prevented another from performing its desired allocations.
- There was no good way to observe the values of the parameters.
- The default values of certain tunables were too small.

In the Solaris 10 OS, all these limitations were addressed. The System V IPC implementation in the Solaris 10 OS no longer requires changes in the `/etc/system` file. Instead, it uses the resource control facility, which brings the following benefits:

- It is now possible to install and boot an Oracle instance without needing to make changes to `/etc/system` file (or to resource controls in most cases).
- It is now possible to limit use of the System V IPC facilities on a per-process or per-project basis (depending on the resource being limited), without rebooting the system.
- None of these limits affect allocation directly. They can be made as large as possible without any immediate effect on the system. (Note that doing so would allow a user to allocate resources without bound, which would have an effect on the system.)
- Implementation internals are no longer exposed to the administrator, thus simplifying the configuration tasks.
- The resource controls are fewer and are more verbosely and intuitively named than the previous tunables.
- Limit settings can be observed using the common resource control interfaces, such as `prctl(1)` and `getrctl(2)`.
- Shared memory is limited based on the total amount allocated per project, not per segment. This means that an administrator can give a user the ability to allocate many segments and large segments, without having to give the user the ability to create many large segments.
- Because resource controls are the administrative mechanism, this configuration can be persistent using `project(4)` and be made via the network.

In the Solaris 10 OS, the following changes were made:

- Message headers are now allocated dynamically. Previously all message headers were allocated at module load time.
- Semaphore arrays are allocated dynamically. Previously semaphore arrays were allocated from a `seminfo_semmns` sized `vmem` arena, which meant that allocations could fail due to fragmentation.
- Semaphore undo structures are dynamically allocated per-process and per-semaphore array. They are unlimited in number and are always as large as the semaphore array they correspond to. Previously there were a limited number of per-process undo structures, allocated at module load time. Furthermore, the undo structures each had the same, fixed size. It was possible for a process to not be able to allocate an undo structure, or for the process's undo structure to be full.
- Semaphore undo structures maintain their undo values as signed integers, so no semaphore value is too large to be undone.
- All facilities were used to allocate objects from a fixed size namespace, and were allocated at module load time. All facility namespaces are now resizable, and will grow as demand increases.

As a consequence of these changes, the following related parameters have been removed (see Table 2). If these parameters are included in the `/etc/system` file on a Solaris system, the parameters are ignored.

Table 2. System parameters no longer needed in the Solaris 10 OS.

Parameter Name	Brief Description
`semsys:seminfo_semmns`	Maximum number of System V semaphores on the system
`semsys:seminfo_semmnu`	Total number of undo structures supported by the System V semaphore system
`semsys:seminfo_semmap`	Number of entries in semaphore map
`semsys:seminfo_semvmx`	Maximum value a semaphore can be set to
`semsys:seminfo_semaem`	Maximum value that a semaphore's value in an undo structure can be set to
`semsys:seminfo_semusz`	The size of the undo structure
`shmsys:shminfo_shmseg`	Number of segments, per process
`shmsys:shminfo_shmmin`	Minimum shared memory segment size
`msgsys:msginfo_msgmap`	Minimum shared memory segment size
`msgsys:msginfo_msgssz`	Size of the message segment
`msgsys:msginfo_msgseg`	Maximum number of message segments
`msgsys:msginfo_msgmax`	Maximum size of System V message

As described above, many `/etc/system` parameters are removed simply because they are no longer required. The remaining parameters have more reasonable defaults, enabling more applications to work out-of-the-box without requiring these parameters to be set.

Table 3 describes the default value of the remaining `/etc/system` parameters.

Table 3. Default values for system parameters in the Solaris 10 OS.

Resource Control	Obsolete Tunable	Old Default Value	New Default Value
process.max-msg-qbytes	msginfo_msgmnb	4096	65536
process.max-msg-messages	msginfo_msgtql	40	8192
process.max-sem-ops	seminfo_semopm	10	512
process.max-sem-nsems	seminfo_semmsl	25	512
project.max-shm-memory	shminfo_shmmax	0x800000	1/4 of physical memory
project.max-shm-ids	shminfo_shmmni	100	128
project.max-msg-ids	msginfo_msgmni	50	128
project.max-sem-ids	seminfo_semmni	10	128

Setting System V IPC Parameters for Oracle Installation

Table 4 identifies the values recommended for `/etc/system` parameters by the Oracle Installation Guide and the corresponding Solaris resource controls.

Table 4. Recommended values for system parameters when running Oracle 10g.

Parameter	Oracle Rec. Value	Required in Solaris 10 OS	Resource Control	Default Value
SEMMNI (semsys:seminfo_semmni)	100	Yes	project.max-sem-ids	128
SEMMNS (semsys:seminfo_semmns)	1024	No	N/A	N/A
SEMMSL (semsys:seminfo_semmsl)	256	Yes	process.max-sem-nsems	512
SHMMAX (shymsys:shminfo_shmmax)	4294967295	Yes	project.max-shm-memory	1/4 of physical memory
SHMMIN (shmsys:shminfo_shmmin)	1	No	N/A	N/A
SHMMNI (shmsys:shminfo_shmmni	100	Yes	project.max-shm-ids	128
SHMSEG (shmsys:shminfo_shmseg)	10	No	N/A	N/A

Since the default values are higher than Oracle recommended values, the only resource controls that might need to be set are `project.max-shm-memory` and `process.max-sem-nsems`. When setting `project.max-shm-memory`, set it higher than sum of SGAs of all Oracle instances. If it is planned to have great number of Oracle processes (process parameter in `init.ora`) then consider increasing `process.max-sem-nsems`. Its value should be higher then what is planned for the process parameter.

The following section details the process of setting a particular value using resource control.

Using Resource Control Commands to Set System V IPC Parameters

The `prctl` command can be used to view and change the value of resource controls of running processes, tasks and projects. The `prctl` command is invoked with the `-n` option to display the value of a certain resource control. The following command displays the value of the `max-file-descriptor` resource control for the specified process:

```
# prctl -n process.max-file-descriptor pid
```

The following command updates the value of `project.cpu-shares` in the project `group.dba`:

```
# prctl -n project.cpu-shares -v 10 -r -i project group.dba
```

The following commands create an Oracle project that is persistent across reboots, with a `SHMMAX` value of 32 GB and `SEMMSL` set to 4096:

```
# projadd -c "Oracle project" group.dba
# projmod -sK "project.max-shm-memory=(privileged,32G,deny)" group.dba
# projmod -sK "process.max-sem-nsems=(privileged,4096,deny)" group.dba
```

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN (9786) **Web** sun.com

Printed in USA 01/09