



Knowledge Base

[Secure Enterprise Search](#)[Advanced](#)[Saved Searches](#)

Did this article help solve your problem?

Would you recommend this document to others?

TIP: Click [help](#) for a detailed explanation of this page.[Bookmark](#)[Go to End](#)Subject: **Running Oracle Database in Solaris 10 Containers - Best Practices**Doc ID: [Note:317257.1](#)Type: **WHITE PAPER**Last Revision Date: **17-FEB-2007**Status: **PUBLISHED**

Running Oracle Database in Solaris 10 Containers

Best Practices

05/03/06

Version 1.1

Table of Contents

- [1 Executive Summary](#)
- [2 Document Scope](#)
- [3 Overview](#)
 - [3.1 Solaris Containers](#)
- [3.2 Solaris Zones](#)
- [3.3 Solaris Resource Manager](#)
 - [3.3.1 Workload identification](#)
- [3.3.2 Resource Controls](#)
- [3.3.3 CPU and Memory Management](#)
- [4 Oracle license model for Solaris 10 Containers](#)
- [5 Creating a local container](#)
 - [5.1 Requirements](#)
- [5.2 Creating a resource pool](#)
- [5.3 Creating a local zone](#)
- [6 Special considerations](#)
 - [6.1 File Systems](#)
- [6.2 Raw Devices](#)
- [6.3 Backup with Recovery Manager](#)
- [6.4 Volume Management](#)
- [6.5 CPU visibility](#)
- [7 Oracle features not available in local containers](#)
 - [7.1 Oracle RAC](#)
- [7.2 Solaris Dynamic Intimate Shared Memory \(DISM\)](#)
- [8 Appendix](#)
 - [8.1 Appendix-1 Script to create a local container](#)
 - [8.1.1 README.txt](#)

- [8.1.2 setenv.sh](#)
 - [8.1.3 zone_cmd_template.txt](#)
 - [8.1.4 pool_cmd_template.txt](#)
 - [8.1.5 create_zone_cmd.sh](#)
 - [8.1.6 create_pool_cmd.sh](#)
 - [8.1.7 create_container.sh](#)
 - [8.2 Appendix-2 Setting System V IPC kernel parameters](#)
 - [9 References](#)
-

1 Executive Summary

This document provides an overview of Solaris Containers and guidelines for running a non-RAC Oracle database in a non-global Solaris 10 Container. Oracle databases (RAC and non-RAC) have been certified to run in a global Solaris 10 Container. This document concentrates on running non-RAC Oracle database in non-global Solaris Container and explains in detail the process of creating a non-global Solaris Container appropriate for deploying an Oracle database. Additionally, it captures special considerations for running non-RAC Oracle databases in a non-global Solaris 10 Container. Oracle RAC does not work in non-global Solaris 10 Containers.

For the remainder of this document a `non-global Solaris 10 Container` will be referred as a `local container`, and a `non-RAC Oracle database` will be referred to simply as an `Oracle database`.

2 Document Scope

A new licensing agreement between Sun and Oracle recognizes Solaris 10 capped Containers as hard partitions ([6], [7]). The scope of this document is to define a Solaris 10 Container appropriate for running an Oracle database. In addition, it captures limitations and special cases of running Oracle databases in local Solaris Containers.

It is beyond the scope of this document to explain how Solaris Container 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 Container technology for server consolidation.

3 Overview

This section provides a brief overview of Solaris Container technology. It also gives an introduction to Solaris Zones and Solaris Resource Manager, which are the two major components of Solaris Containers.

3.1 Solaris Containers

Solaris Containers are designed to provide a complete, isolated and secure runtime environment 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 application uses, allowing multiple applications to operate on a single server while maintaining specified service levels.

Solaris Containers are a management construct intended to provide a unified model for defining and administering the environment within which a collection of Solaris processes executes. Solaris Containers use Solaris Resource Manager (SRM) features along with Solaris Zones to deliver a virtualized environment that can have fixed resource boundaries for application workloads.

3.2 Solaris Zones

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. Non-global zones are also referred to as local zones. The underlying original 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 local zones can be created by an administrator of a global zone. Once created, these local zones can be administered by individual zone administrators, whose privileges are confined to that local zone.

Two types of local zones can be created using different root file system models: sparse and whole root. The sparse root zone model optimizes the sharing of objects by only installing a subset of the root packages and using read-only loopback file system to gain access to other files. By default with this model, the directories `/lib`, `/platform`, `/sbin` and `/usr` will be mounted 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. The whole root zone model provides for maximum configurability by installing the required and any selected optional Solaris packages into the private file systems of the zone. The advantages of this model include the ability for zone administrators to customize their zones 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 on Solaris Zones. However, applications running in local zones need to be aware of local zones behavior, in particular:

- All processes running in a zone have a reduced set of privileges. The set of privileges in a local zone is a subset of privileges available in the global zone. Hence, processes that require a privilege which is not available in a local zone, can either fail or experience poor performance.
- Each local zone has 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.
- Local 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.

3.3 Solaris Resource Manager

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

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 system resources the workload is actually using.
- The ability to control the workloads so they do not interfere with one another and also get required system resources to meet predefined service level agreement.

SRM provides three types of workload control mechanisms:

- The *constraint mechanism* allows the Solaris system administrator to limit the resources a workload is allowed to consume.
- The *scheduling mechanism* 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* ensures that pre-defined system resources are assigned to a given workload.

3.3.1 Workload identification

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 several 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 files, NIS or LDAP depending on the definition of *projects* database source in `/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 like `ps`, `pgrep`, `pkill`, `prstat` and `cron`.

3.3.2 Resource Controls

Resource usage of workloads can be controlled by placing bounds on resource usage. These bounds can be used to prevent a workload from over-consuming a particular resource and interfering with other workloads. The Solaris Resource Manager provides a resource control facility to implement constraints on resource usage.

Each resource control is defined by 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 light weight processes to any process in that task.

3.3.3 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* and *Resource Capping Daemon* facilities respectively.

Fair Share Scheduler

The default scheduler in the Solaris operating system 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 (FSS) 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 CPUs 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.

Oracle Database 9iR2 (9.2.0.6 onwards) and subsequent database releases are certified with the Solaris 10 Operating Environment using the Fair Share Scheduler (FSS) on all supported instruction sets (SPARC, x86 and x86-64).

- * The corresponding non-RAC databases are certified to work in Solaris 10 global and non-global (local) Containers.
- * The corresponding RAC databases are certified to work in Solaris 10 global Containers, with both Oracle Clusterware and Sun Cluster.
- * The Fair Share Scheduler(FSS) requires Solaris 10 11/06 or higher. If using a prior update of Solaris 10, install the following patches:
 - * Download patch # 118833-36 from sunsolve.sun.com before installing Oracle 10gR2 for Solaris 10 on SPARC.
 - * Download patch # 118855-36 from sunsolve.sun.com before installing Oracle 10gR2 for Solaris 10 on x86-64
 - * Patches can be downloaded from the following URL: www.sunsolve.sun.com. Click on the Patch Portal link to find the above patches.

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 specifiedsystem's physical memory utilization 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.

Note that the *rcapd* daemon cannot determine which pages of memory are shared with other processes or which are mapped multiple times within the same process. Hence, it is not recommended that shared memory intensive applications, like Oracle database, run under projects that uses *rcapd* to limit physical memory usage.

4 Oracle license model for Solaris 10 Containers

Oracle now recognizes capped Solaris 10 Containers as licensable entities, known as hard partitions. Oracle licensing policy defines hard partitioning as "A physical subset of a server that acts like a self-contained server". Oracle customers running Oracle in a Solaris 10 environment can now license only the CPUs or cores that are in a capped Solaris Container as discussed below.

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 Solaris Zone to this Resource Pool. The license is driven by the number of CPUs or cores in this Pool.

5 Creating a local container

This section provides instructions for creating a Solaris 10 local container appropriate for installing and running Oracle. These instructions have been followed in the sample scripts documented in Appendix-A, which provide a convenient way of creating such containers.

5.1 Requirements

1. Ensure that the file system in which the root directory for the local container will be placed has at least 6 GB of physical disk space. This will be enough to create the container and install Oracle.
2. Identify the physical interface that will be used to bring up the virtual interface for the local container. To find the physical interfaces available in the global container execute the command:

```
/usr/sbin/ifconfig -a.
```

Example of common interfaces are ce0, bge0 or hme0.

3. Obtain an IP address and a hostname for the local container.. This IP address must be in the same subnet as the IP assigned to the physical interface selected in the previous step.
4. Ensure that the netmask for the IP address of the local container can be resolved in the global container 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. Determine the number of CPUs to be reserved for the local container. To find the number 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 1 CPU.

5.2 Creating a resource pool

The resource pool can be created by the root user in the global container following these steps:

1. Enable the pool facility with the command

```
pooladm -e
```

2. Use the file *pool_cmd_template.txt* provided in the appendix as a template to create a command file. Replace the strings *PSET_NAME*, *NUM_CPUS_MIN*, *NUM_CPUS_MAX* and *POOL_NAME* with appropriate values. This command file has instructions to create a resource pool and a processor set (pset), and then to associate the pset to the resource pool.
3. If the default configuration file */etc/pooladm.conf* does not exist, create it by executing the command

```
pooladm -s /etc/pooladm.conf
```

4. Create the resource pool by executing the command

```
poolcfg -f pool_commands.txt ,
```

where *pool_commands.txt* is the file you created two steps before.

5. Instantiate the changes made to the static configuration by executing the command

```
pooladm -c
```

6. Validate your configuration by executing the command

```
pooladm -n
```

5.3 Creating a local zone

Once the resource pool is available, a local zone can be created and bound to it. For the purposes of installing and running Oracle the local zone can have either a whole root model or a sparse root model. Unless it conflicts with specific requirements we recommend the use the sparse root model. A local Solaris zone can be created as follows:

1. Create as root a directory where the root of the local zone will be placed (for example */export/home/myzone*) and set the access permissions to 700. The name of this directory should match the name of the zone (*myzone*, in this example).
2. Unless special instructions are added, the directory */usr* in the local container will be a loopback file system (lofs). This means that the local container will mount in read-only mode */usr* from the global container into */usr* of its own files systems tree. By default the Oracle installer requires the root to create files in */usr/local* directory. Since */usr* will be read-only in the local container we will create a

special mount point in `/usr/local` to allow root to create such files in the local container . Check if the directory `/usr/local` exist 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 local container. The recommendation is to create it in `/opt/<ZONE_NAME>/local`.
4. Use `zone_cmd_template.txt` file in Appendix-1 as a model to create a command file to create the zone and bind it to the resource pool previously created. Replace the strings `ZONE_DIR`, `ZONE_NAME`, `POOL_NAME`, `NET_IP` and `NET_PHYSICAL` with appropriate values. Notice that this template file expects the special mount point for `/usr/local` to be found in `/opt/<ZONE_NAME>/local`. In this file, the command `create` is used to create a sparse root. Replacing this command with `create -b` would create a whole root. Also, in this file the zone is bound to the pool with the command `set pool=POOL_NAME` . Another way to bind a pool to a zone is to use the command `poolbind(1M)` once the pool and the zone have been created.

5. Create the zone by executing as root the command

```
zonecfg -z <ZONE_NAME> -f zone_commands.txt
```

where `zone_commands.txt` is the file you create in the previous step.

6. Install the zone by executing as root the command

```
zoneadm -z <ZONE_NAME> install
```

7. Boot the zone with the command

```
zoneadm -z <ZONE_NAME> boot
```

8. Finish the configuration of your local container with the command

```
zlogin -C <ZONE_NAME>
```

6 Special considerations

In this section we point out some special consideration when running Oracle database inside a local container.

6.1 File Systems

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

Here we present 4 different ways of mounting a file system from global zone to local zone:

1. Create a file system in global zone and mount it in local zone as loopback file system (lofs)

- Log in as global zone administrator

- Create a file system in global zone

```
global# newfs /dev/rdisk/c1t0d0s0
```

- Mount the file system in the global zone

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

- Add the file system of type lofs to the local 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
```

2. Create a file system in the global zone and mount it to the local zone as UFS.

- Log in as global zone administrator.
- Create a file system in the global zone

```
global# newfs /dev/rdisk/c1t0d0s0
```

- Add the file system of type ufs to the local 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/rdisk/c1t0d0s0
zonecfg:my-zone:fs>set type=ufs
zonecfg:my-zone:fs>end
```

3. Export a device from global zone to local zone and mount it from the local zone.

- Log in as global zone administrator.
- Export a raw device to the local zone

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

- Log in as root in local zone
- Create a file system in the local zone
 - Mount the file system in the local zone

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

4. Mount a UFS file system directly into the local zone's directory structure. This assumes that the device is already made available to the local zone.

- Log in as local zone administrator.
- Mount the device in local zone.

```
my-zone#mount /dev/dsk/c1t1d0s0 /usr/mystuff
```

6.2 Raw Devices

To guarantee that security and isolation is not compromised, certain restrictions regarding devices are

placed in local zones. These restrictions are:

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

The zone administrator can make physical devices available to local zones. It is the administrator's responsibility to ensure that security of the system is not compromised by doing so, mainly because: 1) Placing a physical device into more than one zone can create a covert channel between zones and 2) Global zone applications that use such a device risk the possibility of compromised data or data corruption by a local zone.

The global zone administrator can use the `'add device'` sub-command of `zonecfg` to include additional devices in local zone. For example to add `/dev/dsk/c1t1d0s0` device node to the local zone, the administrator execute 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> set match=/dev/rdisk/c1t1d0s0
zonecfg:my-zone>end
zonecfg:my-zone>exit
zoneadm -z my-zone reboot
```

All the slices of `/dev/dsk/c1t1d0` could be added to the local zone by using `/dev/dsk/c1t1d0*` in the match command.

6.3 Backup with Recovery Manager

Recovery manager (`rman`) is an Oracle utility in Oracle database version 8.0 and higher to backup, restore and recover database files. To store backup on tape, RMAN requires a media manager. A media manager is a third party software that manages sequential media such as tape drives to backup and restore database files. Although `rman` works fine on local zones, the media managers may not be certified to work on local zones. This would potentially imply that currently `rman` utility may only be used to backup and restore database files on disks. The backup taken by `rman` on disks can however be copied over to tape devices from the global zone.

6.4 Volume Management

Volume managers are third party products, so their usability in local containers cannot be generalized. As of this writing, Volume Managers cannot be installed or managed from local containers. Currently, the recommended approach is to install and manage Volume Managers from global containers. Once the devices, file systems and volumes are created in the global container, they can be made available to the local container by using `zonecfg` subcommands.

In the case of Veritas, the following features are **not** supported in local containers:

- Admin `ioctls`
- Administration commands
- Veritas volumen manager Software
- VSM
- VFS/VxMS
- Quick I/O and CQIO

- Cluster File System

The following Veritas features are supported in local zones:

- Veritas file systems
- Access to VxFS file system in the global zone from the local zone of a lofs file system
- Access to ODM files from local zones
- Concurrent I/O with files from local zones

Veritas commands will be accessible in local zones, but they have been enhanced to ensure they are not executed in a local zone. When Veritas Volume Manager commands detect local zone execution, the following error message will be presented:

```

VxVM command_xxx ERROR msg_id: Please execute this operation in global zone.

```

In a similar way, Solaris Volume Manager (SVM) should also be installed and managed from global containers. Once the storage has been configured in the desired way from the global zone, the metadevices can be made available to the local zones.

6.5 CPU visibility

Users in a local Solaris Container can expect a virtualized view of the system 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.

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 (i.e., the number of CPUs in the resource pool) if the zone is bound to a resource pool with a pset associated with it. The interfaces that have been modified in Solaris and that will return the expected value in this scenario are:

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

sar(1M)	Command
---------	---------

In addition to these interfaces, there are certain kernel statistics (kstats) used commonly by tools such as *psrinfo(1M)*, *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.

7 Oracle features not available in local containers

This section covers some of the Oracle database features that are not available in local Containers.

7.1 Oracle RAC

An Oracle RAC installation is composed of several nodes, shared storage, and a private interconnect. A local Solaris Container cannot be used as an Oracle RAC node, mainly because of the following two reasons:

Cluster manager: One of the software components that must be present in a RAC installation is a cluster manager. The cluster manager is responsible for isolating failed systems from the shared storage, to avoid corruption of data, and enabling communication between nodes in the cluster via a private interconnect. In order to use local Solaris Containers as RAC nodes the cluster manager would need to become capable of managing local Containers. As of this writing no cluster solution is capable of using local containers as cluster members or nodes.

Oracle RAC VIP: Another limitation to running Oracle RAC in local Solaris Container is the Virtual IP. Oracle RAC uses a virtual interface in each node to distribute client connections to all the active nodes in the cluster. When a node (node A) leaves the cluster one of the other active members of the cluster (node B) will take over its virtual IP address by bringing up an extra virtual interface with the IP address used by node A. In this way node B will service all new connections sent to node A until node A re-joins the cluster. For security reasons, a local Container does not have the privileges required for managing virtual interfaces. Therefore, the VIP mechanism used by Oracle RAC conflicts with the local container security limitations.

7.2 Solaris Dynamic Intimate Shared Memory (DISM)

This section provides a brief overview of DISM and explains why it is not supported in local Solaris Containers.

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 Solaris Operating system. 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 (i.e., it cannot be paged out). This greatly reduces the overhead due to process context switches, which improves Oracle's performance linearity under load.

ISM has been a feature of Solaris since Solaris 2.2 and it is enabled by default in Solaris 2.4 and above. ISM is used by default by Oracle from 7.2.2 onwards. Prior to Oracle 7.2.2, ISM could be turned on in Oracle by adding the directive *use_ism=true* in *init.ora* file. Oracle uses ISM by default and recommends it use. ISM is supported in local containers.

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 shutdown and restarted. DISM overcomes this limitation as it provides shared memory that is dynamically resizable. DISM is supported in Oracle database since Oracle 9i. Oracle uses DISM instead of ISM if the maximum SGA size set by *sga_max_size* parameter in *init.ora* is larger than the sum of its component.

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 "proc_lock_memory" privilege. This privilege is not available in a Solaris local zone. Although Oracle comes up when DISM is invoked in a local zone, it is not able to lock SGA memory in Solaris local Zones. This is likely to cause performance to degrade, therefore Oracle should not be configured to use

DISM in local Containers.

8 Appendix

8.1 Appendix-1 Script to create a local container

The scripts documented in this Appendix can be used to create a local Solaris 10 Container appropriate for installing and running non-RAC instances of Oracle database. These scripts do not represent the only way in which the local container can be created. They are provided as a sample code and should be modified to fit specific requirements and constraints.

These scripts will first create a resource pool and a processor set (pset) resource with a minimum and maximum number of CPUs in it (both values specified by the user). These new resources will be configured in the default configuration file `/etc/pooladm.conf` (see `pooladm(1M)` for details). Once created, the pset will be associated with the resource pool. Next, a sparse root zone will be created with the root directory, IP address and physical 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. Once created, the zone will be bound to the resource pool. The combination of the zone bound to the resource pools will define the local Container in which Oracle can be installed and executed. This local container (and all processes running in it) will only have access to the CPUs associated to the resource pool. 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
- `NET_IP`: IP address 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 in the pset resource
- `NUM_CPUS_MIN`: minimum number of CPUs in the pset resource

2) Issue the following command from the global container:

```
./create_container.sh
```

3) Configure this local container by executing the following command from global container:

```
zlogin -C <zone_name>
```

The files composing this scripts are presented and explained next.

8.1.1 README.txt

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

The scripts in this directory can be used to create a local 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 local container for Oracle

These scripts will first create a resource pool and a processor set (pset) resource with a minimum and maximum number of CPUs in it (both values

specified by the user). These new resources will be configured in the default configuration file `/etc/pooladm.conf` (see `pooladm(1M)` for details). Once created, the pset will be associated with the resource pool. Next, 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. Once created, the zone will be bound to the resource pool; The combination of the zone bound to the resource pools will define the local container in which Oracle can be installed and used. This local container (and all processes running in it) will only have access to the CPUs associated to the resource pool. 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 in the pset resource
 - `NUM_CPUS_MIN`: minimum number of CPUs in the pset resource
- b) from the global container run `./create_container.sh`
- c) Once the local container has been created run `"zlogin -C <zone_name>"` from the global container to finish configuring the zone.

2) giving your local container access to raw devices

If you need to give your local 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:myzone> add device
zonecfg:myzone:device> set match=/dev/rdisk/c3t40d0s0
zonecfg:myzone:device> end
zonecfg:myzone> exit
zonecfg -z my_zone halt
zonecfg -z my_zone boot
```

3) giving your local container access to a file system

If you need to give your local container access to a file system created in the global container follow this example once the local container has been created:

```
global# newfs /dev/rdisk/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/rdisk/c1t0d0s0
```

```
zoncfg:my-zone> set type=ufs
zoncfg:my-zone> end
```

4) to remove pool resources previously created by operating directly on the kernel (see poolcfg(1M) for details) use these commands:

```
poolcfg -d -c 'destroy pool my_pool'
poolcfg -d -c 'destroy pset my_pset'
```

5) 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
```

8.1.2 setenv.sh

This file is where the used defines the parameters to create the local container

```
#!/usr/bin/sh

#host name for the zone
ZONE_NAME=myzone
#directory where to place root dir for the zone
ZONE_DIR=/export/home
#IP for the zone (make sure netmask can be resolved for this IP according to
# the databases defined in nsswitch.conf)
NET_IP=129.146.182.199
#interface used by the zone
NET_PHYSICAL=bge0
#min and max CPUs for the pool bound to the zone
NUM_CPUS_MIN=1
NUM_CPUS_MAX=1

# do not make changes beyond this point
POOL_NAME=pool_$ZONE_NAME
PSET_NAME=ps_$ZONE_NAME
export ZONE_NAME ZONE_DIR NET_IP NET_PHYSICAL
export POOL_NAME PSET_NAME NUM_CPUS_MIN NUM_CPUS_MAX
```

8.1.3 zone_cmd_template.txt

This file contains a template set of commands to create the zone. After replacing some strings by user defined values it will be used to create the zone.

```
create
set zonepath=ZONE_DIR/ZONE_NAME
set autoboot=true
set pool=POOL_NAME
add net
```



```

s/NET_IP/$NET_IP/
}
/NET_PHYSICAL/ {
s/NET_PHYSICAL/$NET_PHYSICAL/
}
/POOL_NAME/ {
s/POOL_NAME/$POOL_NAME/
}
./ {
p
d
}"

```

8.1.6 create_pool_cmd.sh

This script will use the sed utility to create a command file that creates the resources. It replaces the user given parameters in the pool command template file. It is called by create_container.sh

```

#!/bin/sh
# Copyright (c) 2005 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.

sed -e "
/NUM_CPUS_MIN/ {
s/NUM_CPUS_MIN/$NUM_CPUS_MIN/g
}
/NUM_CPUS_MAX/ {
s/NUM_CPUS_MAX/$NUM_CPUS_MAX/g
}
/POOL_NAME/ {
s/POOL_NAME/$POOL_NAME/
}
/PSET_NAME/ {
s/PSET_NAME/$PSET_NAME/
}
./ {
p
d
}"

```

8.1.7 create_container.sh

This is the main script. It will use the parameters given in setenv.sh to create the container.

```
#!/usr/bin/ksh
# Copyright (c) 2005 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 local container to run oracle RDBMS.
# to use this script follow the instructions in teh README.txt file
# located in this directory.

. ./setenv.sh

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

#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
chmod 700 $ZONE_DIR/$ZONE_NAME

#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

#pset alreedy created?
```

```

pooladm -e
pooladm | grep pset | grep $PSET_NAME > /dev/null 2>&1
if [ $? -eq 0 ]
then
echo "ERROR: pset $PSET_NAME already exists. Please choose a different pset name " exit 1
fi

#/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
./create_pool_cmd.sh < pool_cmd_template.txt > /tmp/pool_commands.txt
pooladm -e # enable facility
#check for default config file exists, if not there
#create one with active configuration
if [ ! -f /etc/pooladm.conf ]
then
pooladm -s /etc/pooladm.conf
fi
poolcfg -f /tmp/pool_commands.txt # configure
pooladm -n > /tmp/pool.out 2>&1 # validate
if [ ! $? -eq 0 ]
then
echo ERROR: invalid pool configuration. please see /tmp/pool.out
exit 1
fi
#instantiate config at /etc/pooladm.conf
pooladm -c

# 3)..... zone creation
./create_zone_cmd.sh < 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"

```

8.2 Appendix-2 Setting System V IPC kernel parameters

Prior to Solaris 10, 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 shortcoming:

- 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 Solaris 10 all these limitations were addressed. The System V IPC implementation in Solaris 10 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 a per segment limit. This means that an administrator can give a user the ability to allocate a lot of segments and large segments, without having to give the user the ability to create a lot of large segments.
- Because resource controls are the administrative mechanism, this configuration can be persistent using `project(4)`, as well as be made via the network.

In Solaris 10 the following changes were made which resulted in removing certain `/etc/system` parameters:

- 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 allocated dynamically, and are per-process and per-semaphore arrays. 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 used to allocate objects from a fixed size namespace, allocated at module load time. All facility namespaces are now resizable, and will grow as demand increases.

The following related parameters have been removed. If these parameters are included in the `/etc/system` file on a Solaris system, the parameters are ignored..

Parameter Name	Brief Description
	Maximum number of System V semaphores on the

semsys:seminfo_semmns	system.
semsys:seminfo_semmnu	Total number of undo structures supported by the System V semaphore system.
semsys:seminfo_semume	Maximum number of System V semaphore undo structures that can be used by any one process.
semsys:seminfo_semmap	Number of entries in semaphore map
semsys:seminfo_sevmx	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	The number of entries in a message map
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.

The following table describes the default value of the remaining /etc/system parameters.

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 :

The following table identifies the values recommended for /etc/system parameters by the Oracle Installation Guide and the corresponding Solaris resource controls.

Parameter	Oracle Recommendation	Required in Solaris 10	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	project.max-sem-nsems	512
SHMMAX (shmsys:shminfo_shmmax)		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 control that might need to be set is project.max-shm-memory. 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 value of resource control.

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 *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.staff

```
prctl -n project.cpu-shares -v 10 -r -l project group.staff
```

9 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/817-159>

[3] Sun BluePrints OnLine: Solaris Container Cookbook, Menno Lageman, Sun Microsystems,2005

<http://www.sun.com/blueprints/0505/819-2679.html>

[4] BigAdmin System Administration Portal-Solaris Zones, Sun Microsystems ;

<http://www.sun.com/bigadmin/content/zones>.

[5] Sun Blueprints online : Dynamic Reconfiguration and Oracle 9i Dynamically Resizable SGA ;

<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>.

[Bookmarks](#) [Admin](#) [Profile](#) [Feedback](#) [Sign Out](#) [Help](#)