ORACLE
SOLARIS

An Oracle White Paper
June 2011

Effective Resource Management Using Oracle Solaris Resource Manager

ORACLE

Introduction

Application workloads on a server need to be balanced for the system efficiency. Granular resource management is a necessity to achieve the anticipated performance and service levels in any environment including virtual and consolidated environments. Without good resource management, faulty runaway workloads can bring progress to a halt causing unwanted delays to priority jobs.

In addition, efficient resource management helps organizations economize by consolidating servers. Server consolidation is one of the effective ways to maximize return on investment (ROI) by cutting unnecessary costs on underutilized servers in a datacenter. Resource management allows controlled allocation of resources to different workloads. An OS process and an active database session are examples of a basic unit of workload.

This paper is part 2 of a four-part series. It provides details about Oracle Solaris Resource Manager and how it can be used to manage system resources effectively. Although most of the topics in this paper are discussed from a consolidated point of view, resource management features and the content in this paper are equally applicable, with few exceptions, in consolidated and isolated environments running Oracle Solaris and Oracle Database.

For more information about Oracle Solaris Resource Manager and Oracle Database Resource Manager, plus a case study that provides examples that demonstrate their features, see the other parts of this series:

- Part 1: “Introduction to Resource Management in Oracle Solaris and Oracle Database”
- Part 3: “Effective Resource Management Using Oracle Database Resource Manager”
- Part 4: “Resource Management Case Study for Mixed Workloads and Server Sharing”

The target audience of this paper is Oracle Solaris System Administrators and Oracle Database Administrators. For the sake of simplicity, the acronym "CPU" was used in many places in reference to virtual processors and hardware threads.

Resource Management in Oracle Solaris

Resource management refers to the features, facilities, and infrastructure available in the Oracle Solaris to manage hardware resources, such as processors, memory, disks, and network I/O. Out of the box, Oracle Solaris allows multiple users and applications to share the system resources with equal priority. However, running multiple workloads within a single instance of Oracle Solaris can impose risk when applications are ill-behaved or usage patterns are unpredictable. In a shared environment, an application with memory leaks or serious software bugs might consume an inordinate amount of resources at the expense of other applications, resulting in suboptimal performance of the system. By default, Oracle Solaris grants every process relatively equal access to the available resources on the system, which might not be ideal when different workloads have different levels of priorities in a shared environment. In such environments, effective resource management ensures that service levels and performance goals are sustained.

Oracle Solaris Resource Manager

Oracle Solaris Resource Manager provides specific software components and utilities that are used to manage hardware resources. It is integrated into the operating system, and it is available on SPARC and x86/x64 platforms running Solaris 9 or later.

Oracle Solaris Resource Manager is a key enabler for server and workload consolidation and increased resource utilization. It provides the ability to allocate and control major system resources, such as CPU, virtual memory, physical memory, I/O bandwidth, and number of processes. It also implements administrative policies that govern which resources different users can access and, more specifically, how much of a particular resource each user is permitted to use. Based on the implemented policies, all users can receive resources commensurate with their service levels and the relative importance of their work.

However, note that Oracle Solaris Resource Manager is *not* any of the following:

- A product, application, or process
- A capacity planning tool (although it helps manage capacity and its accounting functions construct usage records for trend analysis)
- A job scheduler (it controls how a process runs on its host system but not when or where it runs)
- A load balancing mechanism across multiple systems in a cluster (it operates only on a single system, so it can be used to manage workloads individually on each member of a cluster)

Building Blocks of Oracle Solaris Resource Manager

The basic building blocks of Oracle Solaris Resource Manager are tasks, projects, and resource controls. Resource limits can be established on a per-task, per-project, and per-process basis.

A *task* is a collection of related processes. A *project* identifies related work or classifies a service, such as a database instance. A *project* might consist one or more tasks that represent a *workload*. That is, a workload is an aggregation of all the processes of an application or group of applications. Every OS process is associated with a project and a task.

A *resource control* dictates how the operating system manages a controlled resource and how it reacts when the imposed resource limit is reached. For example, an Oracle Solaris system administrator at a university could limit the number of light-weight processes (LWPs) in each task to 50 for all tasks in a project created for all undergraduate students, thereby instructing the OS not to allocate further tasks when the established limit is reached. This might help prevent runaway processes from exhausting system resources.

You can use the extended accounting facility to determine the resource consumption of workloads on the target system and then use the resource control facility to place bounds on resource usage. Bounds that are placed on resources prevent workloads from overconsuming resources.

The resource control facility adds the following benefits.

•Dynamic adjustment of resource controls while the system is running

•Containment-level granularity

Resource controls are arranged in a containment level of zone, project, task, or process. The containment level simplifies the configuration and aligns the collected values closer to the particular zone, project, task, or process.

Refer to System Administration Guide: Oracle Solaris Containers--Resource Management and Oracle Solaris Zones for the complete list of resource controls available in Oracle Solaris.

Basic Resource Management

Oracle Solaris has several features and facilities that provide control over different types of resources. Some features, such as real-time scheduling, `nice(1)`, quotas, and processor sets, provide a limited form of resource management.

Real-Time Scheduling

By default, Oracle Solaris uses the Time-Sharing (TS) scheduling class for conventional work. It also offers the real-time (RT) scheduling class, which implements a weighted scheduling policy to ensure that specific workloads or processes get immediate access to the processor. Oracle Solaris Resource Manager has no control over any process running in the RT class. The resource manager's Fair Share Scheduler (FSS) is capable of managing the CPU resources of only those processes that are not running in the RT scheduling class.

For example, on a four-processor system, a single-threaded CPU-bound real-time process might consume one entire processor leaving other time-sharing processes to compete for the remaining three CPUs that are not being used by the real-time process. Because the real-time process might not use the CPU continually, Oracle Solaris Resource Manager might utilize all four processors when the real-time process is idle.

For example, the following `ps` output indicates that the `java` process was running under the TS scheduling class by default. The scheduling class was then changed to RT using the `priocntl(1)` command, making it a high-priority process.

```
% ps -c  -p 18733
   PID  CLS PRI TTY         TIME CMD
 18733   TS  59 ?          24:32 java

# priocntl -s -c RT -i pid 18733

% ps -c  -p 18733
   PID  CLS PRI TTY         TIME CMD
 18733   RT 100 ?          24:32 java
```

For more information, see the man page of `priocntl(1)`.

Processor Binding

Using the processor binding feature in Oracle Solaris, a specific process or a thread or group of threads within a process can be bound to a processor in such a way that all the threads in the process or the specified threads execute only on the designated processor. The binding is not exclusive; the kernel might schedule other threads from other processes on the processor targeted by the bind operation.

Processor binding is stateless meaning binding does not persist across reboots, for example:

```
% pgrep java
1003
16317

/* bind process 16317 to processor 24 */

# pbind -b 24 16317
process id 16317: was not bound, now 24

/* query bindings
   the following example demonstrates that process 16317
   is bound to processor 24, and process 1003 has no bindings */

# pbind -q 16317 1003
process id 16317: 24
process id 1003: not bound

/* unbind process 16317 */

# pbind -u 16317
process id 16317: was not bound, now not bound
```

For more information, see the man page of `pbind(1M)`.

Processor Sets

Processor sets allow partitioning multiprocessor, multicore systems into logical groups or sets where each set has one or more physical or virtual processors assigned to it. Once a processor set is created, the kernel does not schedule threads on the processors in the set (in Oracle Solaris, every process has at least one thread or LWP by default). Explicit binding is required to bind one or more processes or threads to the processor set. Only user-mode threads that have been bound get scheduled within processor sets. Hence, at least one processor must remain available to run operating system kernel threads.

Processor sets are dynamic. The creation and deletion of sets, adding and removing processors to and from sets, and process/thread binding can be performed without requiring a system reboot. Processor set configurations and bindings do not persist across reboots. However, as of Oracle Solaris 10, processor sets can be managed by resource pools allowing processor set configurations and bindings to be persistent across reboots.

Isolation is one of the benefits of using processor sets. Workloads running in one processor set are protected from CPU activity taking place in any other processor set.

Processor sets control only CPU activity. The control is at a relatively coarse-grained hardware level because processors might belong to exactly one processor set at a time. Especially on small systems, the granularity might be quite high. For example, on a four-processor system, the minimum resource that can be assigned is one processor, which is 25 percent of the CPU resources. Fine-grained control can be achieved with the help of the Fair Share Scheduler (FSS) class.

The following example creates a processor set, disables interrupts for all processors within the processor set, binds three PSAPPSRV processes to the processor set, queries the processor set bindings, and finally removes the process set.

```
% psrinfo -pv
The physical processor has 64 virtual processors (0-63)
  UltraSPARC-T2 (chipid 0, clock 1582 MHz)

# psrset -c 1-3
created processor set 1
processor 1: was not assigned, now 1
processor 2: was not assigned, now 1
processor 3: was not assigned, now 1

# psrset
user processor set 1: processors 1 2 3

# psrset -f 1

# psrset -b 1 `pgrep PSAPPSRV`
process id 5665: was not bound, now 1
process id 5680: was not bound, now 1
process id 5686: was not bound, now 1

# psrset -q
process id 5665: 1
process id 5680: 1
process id 5686: 1

# psrset -d 1
removed processor set 1
```

For more information, see the man pages of `psrset(1)` and `pbind(1M)`.

Process Interrupt Management

Interrupts are asynchronous events that allow a device or software subsystem to notify a processor that it needs attention. An interrupt temporarily suspends the execution flow of the thread running on the interrupted processor so an interrupt service routine can be executed to handle the interrupt. The suspended thread resumes execution as soon as the interrupt is handled. To balance system performance, Oracle Solaris tries to distribute interrupts evenly across all available processors.

If some applications and workloads demand running application threads without periodic interruptions, consider disabling interrupts on a specific processor or processor set upon which application threads are scheduled by the operating system. Combining interrupt management with processor sets and processor set binding provides a powerful environment for CPU-bound workloads and real-time applications.

The following example disables interrupts on processors 0, 1, and 2.

```
% psrinfo -pv
The physical processor has 64 virtual processors (0-63)
  UltraSPARC-T2 (chipid 0, clock 1582 MHz)

# psradm -i 0-2

# psrinfo | grep no-intr
0       no-intr   since 07/23/2010 01:00:54
1       no-intr   since 07/23/2010 01:00:54
2       no-intr   since 07/23/2010 01:00:54
```

For more information, see the man pages of `psradm(1M)`, `psrset(1)`, and `psrinfo(1)`.

The `nice(1)` Command

The `nice(1)` command permits users to manipulate program execution priority. Unless superuser privilege is invoked, this command only permits the user to lower the priority. Oracle Solaris Resource Manager enforces administrative policies even without the co-operation of the user. Hence, administrative policies should be used when more advanced resource management and control is required.

The following `renice(1)` command increases the scheduling priority by altering the `nice` value of a running `java` process from the base scheduling priority 0 to -10.

```
# prstat -p `pgrep java`

   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 18733 oracle    660M  423M sleep   59    0   0:24:05 0.0% java/160
```

```
# renice -10 -p 18733

# prstat -p `pgrep java`

   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 18733 oracle    660M  423M sleep   59  -10   0:24:05 0.0% java/160
```

For more information, see the man page of `nice(1)`.

Quotas

The file systems in Oracle Solaris have quota mechanisms to restrict the disk consumption of individual users.

The following example demonstrates how to set disk quota for an OS user on a ZFS file system.

```
/* create a ZFS storage pool */

# zpool create -f userpool c0t1d0s6

# zpool list
NAME       SIZE   USED  AVAIL    CAP  HEALTH  ALTROOT
userpool   278G  76.5K   278G     0%  ONLINE  -

/* create a ZFS file system for the engineers in an organization */

# zfs create userpool/engineers

/* set 5 GB quota for individual users */

# useradd -g staff -u 6700 billyeng
# zfs set userquota@billyeng=5G userpool/engineers

/* check the quotas and currant usage */

# zfs userspace userpool/engineers
TYPE        NAME       USED  QUOTA
POSIX User  billyeng  1.95G     5G
POSIX User  root         3K   none
```

Network Resource Management

The Internet Protocol Quality of Service (IPQoS) feature in Oracle Solaris provides the means to manage network resources to maintain Quality of Service (QoS) for network users. It allows network traffic to be allocated to separate classes of service, so urgent traffic gets higher priority than less-important traffic. Different classes of service can be guaranteed a portion of the network bandwidth leading to more predictable network loads and overall system behavior.

In commercial environments, different levels of network service can be provided for selected customers and selected applications by enabling IPQoS. It is also possible to provide differentiated services based on the priorities that are set for applications or users on the network.

Providing QoS involves the following activities:

•Delegating levels of service to different groups, such as customers or departments in an enterprise

•Prioritizing network services for particular groups or applications

•Discovering and eliminating areas of network bottlenecks and other forms of congestion

•Monitoring network performance and providing performance statistics

•Regulating bandwidth to and from network resources

Oracle Solaris Resource Manager and IPQoS have different and separate management domains. Oracle Solaris Resource Manager operates on a per-user or per-application basis, whereas IPQoS manages on a per-port, per-service, or per-protocol basis.

For information on configuring and monitoring IPQoS, see the System Administration Guide: IP Services for Oracle Solaris 10.

Domain Resource Management

Dynamic Domains and Dynamic Reconfiguration

Dynamic domains are created by physically partitioning a server. Each domain contains a subset of the total processors, memory, and I/O channels on the system, and each domain is configured with a dedicated boot disk to run its own Oracle Solaris kernel instance with a unique IP identity in the network. For example, a machine with 16 multicore processors on four system boards might be operated as one system with eight CPUs and two other systems with four CPUs each. Three instances of Oracle Solaris would be running in this configuration.

The Dynamic Domains facility is available in a subset of Oracle's Sun server product line, such as the high-end Sun SPARC Enterprise M-series servers.

The Dynamic Reconfiguration feature in Dynamic Domains enables dynamically adding and removing system boards that contain hardware resources, such as processors, memory, and I/O devices. Oracle Solaris automatically recognizes those dynamically added system resources and makes them available for use without requiring a system reboot.

In addition to resource control, the Dynamic Domains facility provides fault isolation so a failure in one domain does not impact the operation of other domains configured on the same server.

Oracle Solaris Resource Manager is similar to Dynamic Domains in the sense that it also provides software tools to dynamically allocate resources, but it does so in a different way. Oracle Solaris Resource Manager operates within a single instance of Oracle Solaris and provides a finer degree of administrative control over the system resources. The Dynamic Domains facility divides a large hardware system into multiple Oracle Solaris servers and provides tools to manage the transfer of resources between instances of Oracle Solaris running on the same Sun SPARC Enterprise M-series server frame.

Fine-Grained Resource Management

When basic resource controls are inadequate to perform complex tasks, such as dynamic changes to the resource allocations in a consolidated environment, more-advanced resource management and control is required. For example, two workloads might be consolidated onto a single system using processor sets to manage the CPU resource by allocating 10 processors to one workload and 6 processors to the other workload. Although this would provide processor limits and isolate each workload from another, resources could be potentially wasted if either of the workloads is not using all of its share of the processors, because the spare CPU cannot be used by any other workload.

As of Oracle Solaris 10, advanced resource management controls, namely the Fair Share Scheduling class (FSS), resource pools, and virtualized execution environments such as zones, are integrated into the operating system.

Projects

A project is a tag or an identifier that is used to classify a service, such as a database instance. The projects framework in Oracle Solaris provides a stateful namespace for binding users, processes, and applications to resource allocations and limits. The framework is hierarchically structured. A project might have one or more tasks associated with it that represent a workload, and a task might have one or more processes associated with it. Projects are defined in the `project(4)` database, which might be in a local `/etc/project` file, in a Network Information Service (NIS) map, or in a Lightweight Directory Access Protocol (LDAP) name service.

Share allocation is done through attributes in the project definition file. In addition to allowing the allocation of CPU shares, the projects framework provides for settings resource limits at the project, task, and process level. For example, System V IPC resources for shared memory, semaphores, and message queues are defined at the project level. The maximum number of LWPs can be set either at the project level or task level, and traditional UNIX resource limits are defined in the projects database on a per-process basis.

Any user or group can belong to one or more projects. These projects can be used to represent the workloads in which the user or group of users is allowed to participate. This membership can then be the basis of chargeback that is based on usage or initial resource allocations. For example, an administer could define a project called "Cloud" and associate that project with the OS groups called "platinum," "gold," "silver," and "bronze." When any of the members of these groups log in to the system, their processes are associated with the "Cloud" project.

Although an OS user might have a default project assigned, the processes that the user launches can be associated with any of the projects of which that user is a member.

Tasks

A task is simply a collection of related processes. Tasks are assigned task Ids, which are similar to process IDs. For example, when a user logs in to the system, the user's shell is placed into a newly created task. If the user logs in a second time, a second task is created. Invoking `newtask(1)` creates additional tasks.

Each process is a member of one task and each task is associated with one project.

It is possible to bind tasks to processor sets and set their scheduling priorities and classes.

The extended accounting facility can provide accounting data for processes that is aggregated at the process and task levels.

The following example demonstrates configuring the maximum shared memory segment to a desired value using various command options in projects and tasks.

```
/* create a new project called "FMS90" for "oracle" user */

# projadd -p 4321 -c 'PeopleSoft Financials Database' -U oracle -G dba FMS90

# grep FMS90 /etc/project
FMS90:4321:PeopleSoft Financials Database:oracle:dba:

/* find the default project that the user belongs to */

% id -p
uid=20000(oracle) gid=98194051(dba) projid=3(default)

% projects
default FMS90

/* check the default limit for the maximum shared memory segment that can be
        created under the FMS90 project */
```

```
% prtconf | grep Mem
Memory size: 65536 Megabytes

% newtask -p FMS90 ksh

% id -p
uid=20000(oracle) gid=98194051(dba) projid=4321(FMS90)

% prctl -n project.max-shm-memory -i project FMS90
project: 4321: FMS90
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
project.max-shm-memory
        privileged      15.6GB      -   deny                                 -
        system          16.0EB    max   deny                                 -

% grep -i "MEMORY" $ORACLE_HOME/dbs/init$ORACLE_SID.ora
MEMORY_TARGET = 16G

/* start the Oracle Database instance */

% sqlplus / as sysdba

SQL*Plus: Release 11.2.0.1.0 Production on Sun Jul 25 03:47:25 2010
Copyright (c) 1982, 2009, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> startup
ORA-27102: out of memory
SVR4 Error: 22: Invalid argument

/* increase the shared memory segment limit to > 16 GB */

# projmod -K 'project.max-shm-memory=(privileged,32G,deny)' FMS90

/* exit the old task (shell) and launch a new shell to pick up the
        new value for project.max-shm-memory */

% exit

% newtask -v -p FMS90 ksh
154
```

```
% prctl -n project.max-shm-memory -i project FMS90
project: 4321: FMS90
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
project.max-shm-memory
        privileged      32.0GB      -   deny                                 -
        system          16.0EB    max   deny

/* try starting the database instance again */

SQL> startup
ORACLE instance started.

Total System Global Area 1.7108E+10 bytes
Fixed Size                  2165208 bytes
Variable Size            8220841512 bytes
Database Buffers         8824815616 bytes
Redo Buffers               60448768 bytes
Database mounted.
Database opened.
SQL>

/* a few more relevant commands with no outputs */
/* check the existing/current task and project IDs */

% ps -o user,pid,uid,projid,taskid

/* find all the processes that are associated with a project id */

% pgrep -J PROJID

/* find all the processes that are associated with a task id */

% pgrep -T TASKID

/* display various statistics for processes and projects
        that are currently running on the system */

% prstat -J

/* display various statistics for processes and tasks
        that are currently running on the system */

% prstat -T
```

For more information, see the man pages of `ps(1)`, `id(1M)`, `pgrep(1)`, `prstat(1M)`, `projects(1)`, `priocntl(1)`, `projadd(1M)`, `projmod(1M)`, and `projdel(1M)`.

Fair Share Scheduler Class

The Fair Share Scheduler (FSS) scheduling class can be used to control the allocation of available CPU resources among workloads based on their importance. The importance is expressed by the number of shares of CPU resources that are assigned to each workload. The FSS must be the default scheduler on the system to have CPU share assignments take effect.

CPU shares are not the same as CPU percentages. A CPU share indicates a project's entitlement to available CPU resources. Shares define the relative importance of projects with respect to other projects. If project A is twice as important as project B, project A should be assigned twice as many shares as project B. Because shares are meaningful only in comparison with another project's shares, the absolute quantity of shares is not important. Two shares for project A versus one share for project B yields the same results as 18 shares for project A versus nine shares for project B. Project A is entitled to twice the amount of CPU as project B in both cases. The importance of project A relative to project B can be increased by assigning more shares to project A while keeping the same number of shares for project B.

FSS limits the CPU usage only if there is competition for the CPU resources. A single active project on a system can use 100 percent of the CPU irrespective of the number of shares it was allocated. CPU cycles are never wasted. If a project does not use all the CPU it is entitled to because it has no work to perform, the remaining CPU resources are distributed between other active processes. Active projects are the projects with at least one process using the CPU.

FSS continually adjusts priorities of all processes to make each project's relative CPU usage converge with its entitlement.

While FSS is designed to fairly allocate CPU resources over a long-term period, it is possible that projects will not receive their allocated shares of CPU cycles due to uneven demand. This makes one-time instantaneous analysis of FSS performance data unreliable.

Projects with an undefined number of shares are given one share each. This means that such projects are treated with equal importance. Projects with zero shares run only when there are no projects with non-zero shares competing for the same processor set. The maximum number of shares that can be assigned to one project is 65535.

By default, project "system" (project id 0) includes all system daemons started by initialization scripts and has an unlimited amount of shares. Hence, it is always scheduled first no matter how many shares are given to other projects.

FSS can be assigned as the default scheduling class. However, without share assignment, it would behave like the timeshare class. Shares can be assigned in an ad-hoc fashion to running processes and can also be defined as a project attribute.

FSS and Processor Sets

FSS can be used with processor sets to provide more fine-grained control over allocation of CPU resources among projects that run in each processor set. When processor sets exist, FSS treats every processor set as a separate partition. CPU entitlement for a project is based on CPU usage in that processor set only. The CPU allocations of projects running in one processor set are not affected by the CPU shares or activity of projects running in another processor set, because the projects are not competing for the same resources. Projects compete with each other only if they are running within the same processor set.

The share allocation is system-wide. When processor sets are used, project CPU allocations are calculated for active projects that run within each processor set.

The performance and availability of applications that run within the boundaries of their processor sets are not affected by the introduction of new processor sets. The applications are also not affected by changes that are made to the share allocations of projects that run on other processor sets.

Empty processor sets (sets without processors in them) or processor sets without processes bound to them do not have any impact on FSS behavior.

Combining FSS with Other Scheduling Classes

By default, the FSS class uses the same range of priorities (0 to 59) as the timesharing (TS), interactive (IA), and fixed priority (FX) scheduling classes. Therefore, you should avoid having processes from these scheduling classes share the same processor set. A mix of processes in the FSS, TS, IA and FX classes could result in unexpected scheduling behavior.

By using processor sets, you can mix TS, IA, and FX with FSS in one system. However, all the processes that run on each processor set must be in one scheduling class so they do not compete for the same CPUs. The FX scheduler, in particular, should not be used with the FSS class unless processor sets are used. This action prevents applications in the FX class from using priorities high enough to starve applications in the FSS class.

It is possible to mix processes in the TS and IA classes in the same processor set or on the same system without processor sets.

Oracle Solaris also offers a real-time (RT) scheduler to users with superuser privileges. By default, the RT scheduling class uses system priorities in a different range (usually from 100 to 159) than FSS. Because RT and FSS are using disjoint or nonoverlapping ranges of priorities, FSS can coexist with the RT scheduling class within the same processor set. However, the FSS class does not have any control over processes that run in the RT class.

Run the following command to find out which scheduling classes the processor sets are running in and to ensure that each processor set is configured to run either TS, IA, FX, or FSS processes.

```
# ps -ef -o pset,class | grep -v CLS | sort | uniq
  -  FSS
  -   TS
  -  SYS
```

The following example creates two projects, `highprof` and `lowprof`, and allocates twice as many CPU shares to the `highprof` project than to the `lowprof` project.

```
/* create two new projects called highprof and lowprof for oracle user */

# projadd -p 5555 -c 'high profile project' -U oracle -G dba highprof
# projadd -p 4444 -c 'low profile project' -U oracle -G dba lowprof

/* find the current/default share assignment for any of the
       newly created projects */

# newtask -p highprof ksh

# prctl -n project.cpu-shares -i project highprof
project: 5555: highprof
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
project.cpu-shares
        privileged          1       -   none                                 -
        system          65.5K     max   none                                 -


/* allocate 50 shares to lowprof project and 100 shares to highprof project
       that is, lowprof gets ~33.33% CPU while highprof project gets 66.66% */

# projmod -K 'project.cpu-shares=(privileged,100,none)' highprof
# projmod -K 'project.cpu-shares=(privileged,50,none)' lowprof

# newtask -p highprof ksh

#  prctl -n project.cpu-shares -i project highprof
project: 5555: highprof
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
project.cpu-shares
        privileged        100       -   none                                 -
        system          65.5K     max   none                                 -
```

```
/* determine the current scheduling class in use */

# dispadmin -d
dispadmin: Default scheduling class is not set

/* Set the default scheduler for the system to be FSS.
   FSS must be the default scheduler on the system to have
        CPU shares assignment take effect. */

# dispadmin -d FSS

/* The above change takes effect on the next reboot.
   After reboot, every process on the system runs in the FSS class.

   To make this configuration take effect immediately without rebooting, run the
following command. */

# priocntl -s -c FSS -i all

% dispadmin -d
FSS     (Fair Share)

% ps -cafe | grep FMS
  oracle  3121     1  FSS  29 02:49:09 ?     0:00 ora_w000_FMS90
  oracle 26570     1  FSS  29 04:01:33 ?     0:28 ora_vktm_FMS90
  oracle 26618     1  FSS  29 04:01:35 ?     0:02 ora_dbwh_FMS90
  ...
```

To manually move the processes in a project into the FSS class, run the following command:

```
# priocntl -s -c FSS -i projid project
```

For more information, see the man pages of `FSS(7)`, `priocntl(1)`, `dispadmin(1)`, `ps(1)`, `prstat(1)`, and `resource_controls(5)`.

Memory Resource Management

The resource capping daemon `rcapd` allows regulating physical memory consumption by processes running in projects that have resource caps defined. A resource cap is an upper bound placed on the consumption of a resource, such as physical memory. Per-project physical memory caps are supported.

The resource capping daemon and its associated utilities provide mechanisms for memory resource cap enforcement and administration. Similar to the resource control, the resource cap can be defined by using attributes of project entries in the project database. Resource controls are synchronously enforced by the kernel, whereas resource caps are asynchronously enforced at the user level by the resource capping daemon.

The `rcapd` service is managed by the Service Management Facility, `smf(5)`, under the service identifier: `svc:/system/rcap:default`. Only one instance of `rcapd` can run at any time.

The capping daemon repeatedly samples the resource utilization of projects that have physical memory caps defined. When the system's physical memory utilization exceeds the threshold for cap enforcement and if other conditions are met, the daemon takes action to reduce the resource consumption of projects with memory caps to levels at or below the caps. To reduce resource consumption, the daemon pages out or relocates infrequently used pages to a swap device, which is an area outside of physical memory.

Enabling and Disabling the Resource Capping Daemon

Starting with Oracle Solaris 10, the resource capping daemon can be enabled or disabled using the Service Management Facility (SMF).

- Turn on resource capping using the `svcadm` command:

```
# svcadm enable rcap
```

  Enabling resource capping creates the `/etc/rcap.conf` file with default values.

- Turn off resource capping using the `svcadm` command:

```
# svcadm disable rcap
```

Limiting Physical Memory Usage for a Project

To define a physical memory resource cap for a project, establish a resident set size (RSS) cap by adding the `rcap.max-rss` attribute to the project's database entry. This attribute specifies the total amount of physical memory in bytes that is available to all the processes in the project. The operating system might round the specified cap value to align to the memory page size.

The following example demonstrates controlling physical memory using the resource capping daemon.

```
/* create a project with the memory cap set to 10 MB */

# projadd -p 9999 -K 'rcap.max-rss=10M' consolidate

/* associate the new project with the "test" OS user */

# usermod -K 'project=consolidate' test

/* for the sake of demonstration, change the operation intervals
   from the defaults to some random values */

# rcapadm -i scan=30,sample=60,report=2,config=15

/* monitor the total RSS for the test user before enforcing memory capping */

$ id -p
uid=9898(test) gid=98194051(dba) projid=9999(consolidate)

   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 25246 test     1648K 1048K cpu220   0    4   0:04:13 0.4% testproc1/1
 25247 test     1648K 1064K cpu238   0    4   0:04:08 0.4% testproc2/1
 25248 test     1712K 1128K cpu65   38    4   0:04:04 0.4% testproc3/1
 25267 test     1648K 1024K cpu10    0    4   0:00:36 0.3% testproc4/1
 25251 test     7640K 6560K sleep   59    0   0:00:00 0.0% sshd/1
 25257 test     1888K 1608K sleep   59    0   0:00:00 0.0% ksh/1
 25209 test     7640K 4832K sleep   59    0   0:00:00 0.0% sshd/1
 25130 test     1888K 1480K sleep   59    0   0:00:00 0.0% ksh/1
 25124 test     7640K 4680K sleep   59    0   0:00:02 0.0% sshd/1
 25094 test     1888K 1600K sleep   59    0   0:00:00 0.0% ksh/1
 25215 test     1888K 1592K sleep   59    0   0:00:00 0.0% ksh/1

 NPROC USERNAME  SWAP   RSS MEMORY      TIME  CPU
    12 test     8456K   13M   0.0%   0:13:04 1.5%

Total: 12 processes, 12 lwps, load averages: 3.56, 2.71, 1.80

/* now enable the resource capping daemon to cap the total
        physical memory used by “test” user to 10 MB */

# svcadm -v enable rcap
svc:/system/rcap:default enabled.
```

```
/* Monitor the resource capping daemon's activity.
        Notice how the values under the "rss" column change. */

# rcapstat 2
    id project         nproc    vm   rss   cap    at avgat    pg avgpg
  9999 consolidate        12 8456K   13M   10M   16K    0K   16K    0K
  9999 consolidate         - 8456K   13M   10M    0K    0K    0K    0K
  9999 consolidate        12 8456K   13M   10M 7600K    0K 4104K    0K
  9999 consolidate         - 8456K   13M   10M    0K    0K    0K    0K
  9999 consolidate        12 8456K 9800K   10M   12M    0K 2800K    0K
  9999 consolidate         - 8456K 9800K   10M    0K    0K    0K    0K
  9999 consolidate         - 8456K 9800K   10M    0K    0K    0K    0K
  9999 consolidate         - 8456K 9800K   10M    0K    0K    0K    0K

/* monitor the total RSS for the “test” user one more time
        and notice the difference in "RSS" size for some of the processes */

   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 25246 test     1648K 1048K cpu138   0    4   0:05:19 0.4% testproc1/1
 25247 test     1648K 1064K cpu238   0    4   0:05:14 0.4% testproc2/1
 25248 test     1712K 1128K cpu65    0    4   0:05:10 0.4% testproc3/1
 25267 test     1648K 1024K cpu10    0    4   0:01:41 0.4% testproc4/1
 25251 test     7640K 3984K sleep   59    0   0:00:00 0.0% sshd/1
 25257 test     1888K 1584K sleep   59    0   0:00:00 0.0% ksh/1
 25209 test     7640K 4616K sleep   59    0   0:00:00 0.0% sshd/1
 25130 test     1888K 1472K sleep   59    0   0:00:00 0.0% ksh/1
 25124 test     7640K 4616K sleep   59    0   0:00:02 0.0% sshd/1
 25094 test     1888K 1464K sleep   59    0   0:00:00 0.0% ksh/1
 25215 test     1888K 1464K sleep   59    0   0:00:00 0.0% ksh/1

 NPROC USERNAME  SWAP   RSS MEMORY      TIME  CPU
    12 test     8456K 9800K   0.0%   0:17:27 1.5%

Total: 12 processes, 12 lwps, load averages: 4.00, 3.00, 1.97
```

For more information, see the man pages of `rcapd(1M)`, `rcapadm(1M)`, and `rcapstat(1)`.

Resource Partitioning with Pools

In order to provide predictable service levels to applications on a shared system, system administrators need a way to ensure that applications always have access to a consistent set of resources regardless of the resource usage on the rest of the system. Using the resource pool facility in Oracle Solaris, administrators can partition system resources, such as CPUs and memory, into named groups so that workload consumption of those resources does not overlap. Resource pools provide a persistent configuration mechanism for processor set configuration and, optionally, scheduling class assignment. By default, processor sets are stateless and require a reconfiguration of the processor sets and bindings whenever the system is restarted. Resource pools address this by using the project's database to store processor set configurations and bindings. Thus, a set of CPUs can be configured as a resource pool with a specific project bound to the pool.

In short, a resource pool is a logical entity that owns a subset of the system resources. Every resource pool is associated with a processor set, so to give the pool its own unique CPUs, define a processor set with the number of processors it contains and associate it with the resource pool.

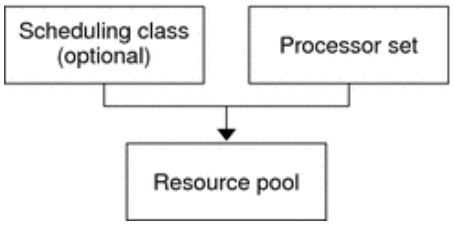

Figure 1. Resource Pool Framework

Resource pools enable creating a processor set by specifying the number of processors required rather than CPU physical IDs. The definition of a processor set is, therefore, not tied to any particular type of hardware. It is also possible to specify a minimum and maximum number of processors for a pool. Multiple configurations can be defined to adapt to changing resource requirements, such as different daily, nightly, or seasonal workloads.

Resource pools can have different scheduling classes. Scheduling classes work per resource pool. The two most common are Fair Share Scheduler and Time Sharing Scheduler. While the resource pool facility provides the ability to partition system resources, it does not allow you to specify how resources are to be shared by applications within the resource pool. FSS can be used within the resource pool to ensure higher-priority applications receive more resources. With Oracle Solaris, different resource pools can be associated with different schedulers to take advantage of the schedulers that best serve the needs of the applications running in them.

When pools are enabled, a default pool and a default processor set form the base configuration. Additional user-defined pools and processor sets can be created and added to the configuration. A CPU can belong to only one processor set. User-defined pools and processor sets can be destroyed. The default pool and the default processor set cannot be destroyed.

A resource pool can also be applied to an Oracle Solaris 10 zone.

It is not possible to enable the pools facility on a system where processor sets have been created. Use the `psrset(1M)` or `pset_destroy(2)` command to destroy processor sets manually before enabling the pools facility.

Dynamic Resource Pools

Dynamic resource pools provide a mechanism for dynamically adjusting each pool's resource allocation in response to system events and application load changes. Adjustments are automatically made to preserve the system performance goals specified by an administrator or a privileged user. The changes made to the configuration are logged. These features are primarily enacted through the resource controller `poold`, a system daemon that should always be active when dynamic resource allocation is required. Periodically `poold` examines the load on the system and determines whether intervention is required to maintain optimal performance with respect to resource consumption.

Resource Pools and Dynamic Reconfiguration Operations

This section is applicable to the SPARC platform only.

Dynamic Reconfiguration (DR) allows reconfiguring the hardware while the system is running. A DR operation can increase, reduce, or have no effect on a given type of resource. Because DR can affect available resource amounts, the pools facility must be included in these operations.

When a DR operation is initiated, the pools framework acts to validate the configuration. If the DR operation can proceed without causing the current pools configuration to become invalid, the private configuration file is updated. An invalid configuration is one that cannot be supported by the available resources. If the DR operation causes the pools configuration to be invalid, the operation fails and the administrator is notified by a message in the message log. The configuration can be forced to complete using the DR force option. The pools configuration is then modified to comply with the new resource configuration.

When dynamic resource pools are active, it is possible for a partition to move out of `poold` control while the daemon is active.

Enabling and Disabling the Pools Facility

As of the latest release of Oracle Solaris 10, resource pools and dynamic resource pools services can be enabled and disabled using the `svcadm` command or the `pooladm` command. The SMF interface is the preferred method for controlling resource pools and dynamic resource pools.

```
/* enable the resource pools service */

# svcadm enable system/pools:default
```

```
/* Enable the dynamic resource pools service.
   Due to a dependency, make sure the resource pools service is online
   before attempting to enable the dynamic resource pools service. */

# svcadm enable system/pools/dynamic:default

# svcs *pool*
STATE          STIME    FMRI
online         15:36:04 svc:/system/pools:default
online         15:36:24 svc:/system/pools/dynamic:default

/* disable the resource pools service */

# svcadm disable svc:/system/pools:default

/* disable the dynamic resource pools service */

# svcadm disable system/pools/dynamic:default

/* example demonstrating the impact of a processor set
   while enabling the resource pools service */

# psrset
#
# psrset -c 0-1
created processor set 1
processor 0: was not assigned, now 1
processor 1: was not assigned, now 1

# svcadm enable system/pools:default
svcadm: Instance "svc:/system/pools:default" is in maintenance state.
pooladm: cannot enable pools: Error 0

# svcs *pool*
STATE          STIME    FMRI
disabled       15:40:52 svc:/system/pools/dynamic:default
maintenance    16:21:17 svc:/system/pools:default

# svcs -x | grep system-pools:default.log
   See: /var/svc/log/system-pools:default.log

# tail  /var/svc/log/system-pools:default.log
..
[ Jul 26 16:21:17 Executing start method ("/lib/svc/method/svc-pools start") ]
```

```
pooladm: cannot enable pools: System has active processor sets
[ Jul 26 16:21:17 Method "start" exited with status 95 ]

# psrset -d 1

# svcadm clear svc:/system/pools:default

# svcs *pool*
STATE          STIME    FMRI
disabled       15:40:52 svc:/system/pools/dynamic:default
online         16:23:05 svc:/system/pools:default
```

The following example demonstrates the creation of two resource pools on a system that has 64 virtual processors. The `batch_pool` resource pool performs batch processing that uses the default scheduling class in Oracle Solaris, the timeshare or interactive class. The `database_pool` resource pool performs database operations that use the FSS class.

```
# psrinfo -pv
The physical processor has 64 virtual processors (0-63)
  UltraSPARC-T2 (chipid 0, clock 1582 MHz)

# pooladm
pooladm: couldn't open pools state file: Facility is not active

/* enable pools facility */

# svcadm enable system/pools:default

/* Create two input files, as shown below.
   These files will be used to configure the resource pools. */

% cat /tmp/batchcfg
create pset batch-pset ( uint pset.min = 16; uint pset.max = 16 )
create pool batch-pool
associate pool batch-pool ( pset batch-pset )

/* database-pool consists of a dynamic resource pool configuration
   and specifies that utilization should be kept between 25% and 80%.
   The logging level was set to "information." */
```

```
% cat /tmp/databasecfg
create pset database-pset ( uint pset.min = 16; uint pset.max = 32 )
create pool database-pool (string pool.scheduler = "FSS" )
associate pool database-pool ( pset database-pset )
modify pset pset_oracle (string pset.poold.objectives="utilization > 25; \
        utilization < 80; locality loose")
modify system (string system.poold.log-level="INFO")

/* /etc/pooladm.conf must exist before instantiating a new configuration */

# ls /etc/pooladm.conf
/etc/pooladm.conf: No such file or directory

# pooladm -s

# ls /etc/pooladm.conf
/etc/pooladm.conf

/* update /etc/pooladm.conf with the new pool configuration
    in /tmp/batchcfg and /tmp/databasecfg */

# poolcfg -f /tmp/batchcfg
# poolcfg -f /tmp/databasecfg

/* activate the new configuration from /etc/pooladm.conf into memory */

# pooladm -c /etc/pooladm.conf

# psrset
user processor set 1: processors 31 32 33 34 35 .. 42 43 44 45 46
user processor set 2: processors 0 1 2 3 4 5 6  .. 25 26 27 28 29 30

/* check the current pool configuration */

# pooladm
        ..
        pool database-pool
                boolean pool.active true
                string  pool.scheduler FSS
                pset    database-pset
                ..

        pset database-pset
                uint    pset.min 16
```

```
                uint    pset.max 32
                string  pset.units population
                uint    pset.load 83
                uint    pset.size 31
                ..
#

/* make the current configuration boot-persistent  */

# pooladm -s

/* create two new projects called loyaltybatch and oradb for oracle user */

# projadd -p 6789 -c 'Siebel Loyalty Batch' -U oracle -G dba loyaltybatch
# projadd -p 5678 -c 'Oracle RDBMS' -U oracle -G dba oradb

/* use the project.pool attribute of projmod to associate the
        newly created resource pools with the loyaltybatch and oradb projects */

# projmod -a -K 'project.pool=batch-pool' loyaltybatch
# projmod -a -K 'project.pool=database-pool' oradb

% id -p
uid=602324(oracle) gid=98194051(dba) projid=3(default)

% newtask -p loyaltybatch ./loyaltyeng &
% newtask -p loyaltybatch ./postdiscount &
% newtask -p oradb ./gatherstats &

% prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  5036 oracle   1600K  952K cpu0     1    0   0:01:22 1.5% gatherstats/1
  5079 oracle   1600K  952K cpu31    0    4   0:00:35 1.3% loyaltyeng/1
  5080 oracle   1600K  952K cpu37    0    4   0:00:26 1.1% postdiscount/1
        ...

PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
  6789        2  336K 1808K   0.0%   0:01:01 2.4% loyaltybatch
  5678        1  168K 1640K   0.0%   0:01:22 1.5% oradb
     1        2 1688K 8496K   0.0%   0:00:00 0.0% user.root
     3        5 6208K   16M   0.0%   0:00:00 0.0% default
     0       45  201M  298M   0.2%   0:35:48 0.0% system

/* use the poolstat utility to examine all active pools */
```

```
% poolstat
                           pset
 id pool                 size used load
  4 database-pool          31 0.00 0.89
  0 pool_default           17 0.00 0.04
  3 batch-pool             16 0.00 1.44

% ps -cafe | egrep "gather|loyal|post"
  oracle  5079  5044   TS   0 03:36:34 pts/3       2:32 ./loyaltyeng
  oracle  5036  4993  FSS   1 03:35:48 pts/2       3:18 ./gatherstats
  oracle  5080  5044   TS   0 03:36:43 pts/3       2:23 ./postdiscount

% poolstat -r pset 2
 id pool                 type rid rset                 min  max size used load
  4 database-pool        pset   2 database-pset          16   32   31 1.00 0.58
  0 pool_default         pset  -1 pset_default            1  66K   17 0.04 0.04
  3 batch-pool           pset   1 batch-pset             16   16   16 2.00 1.37
       ...

/* Dynamic reconfiguration example:
       move eight processors from database-pset to batch-pset */

# poolcfg -dc 'transfer 8 from pset database-pset to batch-pset'
```

The following error indicates that some configured resource is either not available or beyond its limits on the system. Double-check the configuration.

```
pooladm: configuration at '/etc/pooladm.conf' cannot be instantiated on current system
```

For more information, see the man pages of `pooladm(1M)`, `poolcfg(1M)`, `poold(1M)`, `poolstat(1M)`, `poolbind(1M)`, `rctladm(1M)`, and `smf(5)`.

Extended Accounting

The extended accounting facility in Oracle Solaris provides a flexible way to record system and network resource consumption on a task or process basis. Extended accounting is strictly not for monitoring system usage in real time. It enables examining historical usage for reporting purposes. You can then make assessments of capacity requirements for future workloads. With extended accounting data available, you can develop software for resource chargeback, workload monitoring, or capacity planning.

As of Oracle Solaris 10 9/10, memory usage is not accounted in extended accounting.

The extended accounting facility uses a versioned, extensible file format to store accounting data. Files that use this data format can be accessed by using the `libexacct` API. These files can then be analyzed on any platform with extended accounting enabled, and their data can be used for capacity planning and chargeback.

To activate extended accounting for tasks, processes, and flows, use the `acctadm` command. The optional final parameter indicates whether the command should act on the process, system task, or flow accounting components of the extended accounting facility.

```
# acctadm -e extended -f /tmp/acctproc proc

# acctadm process
         Process accounting: active
    Process accounting file: /tmp/acctproc
  Tracked process resources: extended
Untracked process resources: host
#
```

The `-r` option of the `acctadm` command displays the accounting resources available on the system.

To deactivate process, task, and flow accounting, turn off each of them individually by using the `acctadm` command with the `-x` option.

For example, the following command turns off process accounting.

```
#  acctadm -x process
```

To display the contents of an extended accounting (exacct) file in plain text, use the `/usr/demo/libexacct/exdump.c` file.

```
# cd /usr/demo/libexacct

/* build an executable by compiling exdump.c with any supported C compiler */

# /opt/SUNWspro/bin/cc -o exdump exdump.c -lexacct -lproject -lsocket -lnsl

/* specify the exactt file as input to exdump utility, and
        examine the output displayed in plain text */

# ./exdump  -v /tmp/acctproc
    ff  group-header                    [group of 4 object(s)]
   ...
  1022   memory-rss-avg-k               632
```

```
1023   memory-rss-max-k              21576
 100  group-proc                    [group of 34 object(s)]
1001   uid                           15000          oracle
1002   gid                           98194049       dba
1004   projid                        3              default
1003   taskid                        184
100b   cpu-user-sec                  0
100c   cpu-user-nsec                 24519400
 ...
```

For more information, see the man pages of `acctadm(1M)` and `libexacct(3LIB)`.

Resource Management in Virtualized Environments

Oracle Solaris Zones

Introduced in Oracle Solaris 10, zones provide multiple virtualized and isolated execution environments for running multiple application workloads within a single kernel instance. A zone is a virtualized operating system environment created within a single instance of the Oracle Solaris operating system. When a zone is created, all the processes executing within the zone are isolated from processes running in other zones on the system. This isolation prevents processes that are running in one zone from monitoring or affecting processes that are running in other zones.

At the center of the zones design was consolidation, which is the ability to run multiple applications, including several instances of the same type of service (for example, Web server or database server), in a contained and secure environment with a simple management framework. Installing, configuring, and running applications in a zone is no different than doing so on a standalone system. That is, no changes are required at the application level in order to install and run the software within a zone. End users might perceive the zone as a standalone server running Oracle Solaris.

Zones can be created on any server with Oracle Solaris 10 or later releases installed. The number of zones that can be effectively hosted on a single system is determined by the total resource requirements of the application software running in the zones.

Every Oracle Solaris system contains a global zone. The global zone has a dual function. The global zone is both the default zone for the system and the zone used for system-wide administrative control. The global zone has visibility into all non-global zones, which are sometimes referred as *local zones* or simply *zones*.

Two types of non-global zone root file systems can be created.

- **Sparse root** zones, which are native branded zones. The sparse root zone model optimizes the sharing of objects. Not all brands support the sparse root model.
- The **whole root** zone model, which provides maximum configurability.

Based on a few experimental observations, the CPU overhead for managing a few zones on an Oracle Solaris 10 system is minimal.

Non-Global Zone Characteristics

A zone provides isolation at almost any level of granularity required. A zone does not need a dedicated CPU, a physical device, or a portion of physical memory. These resources can either be multiplexed across a different zones running within a single domain or allocated on a per-zone basis using the resource management features available in the operating system.

Using Resource Management Features in Non-Global Zones

A zone that makes use of the resource management features in Oracle Solaris is called an Oracle Solaris Container. Resources that can be controlled in an Oracle Solaris Container include the following:

•Resource pools or assigned CPUs to partition the system resources

•Resource controls that provide a mechanism for constraining the system resources

•A scheduling class to control the allocation of CPU resources based on their importance.

Resource management in the global zone is similar to using the resource management features on any standalone Oracle Solaris server.

Resource Pool Association

If resource pools are configured on the target system, use the `pool` property to associate the zone with one of the resource pools.

If resource pools are not configured, it is still possible to specify that a subset of the system's processors be dedicated to a non-global zone by using the `dedicated-cpu` resource property. The resource manager dynamically creates a temporary pool for use while the zone is running.

A zone configuration using a persistent pool set through the `pool` property is incompatible with a temporary pool configured through the `dedicated-cpu` property. Only one of these two properties can be configured.

`dedicated-cpu` Resource Property

The `dedicated-cpu` property specifies that a subset of the system's processors should be dedicated to a non-global zone while it is running. Dynamic resource pool behavior can be achieved by specifying a range or number of CPUs using the `ncpus` property of the `zonecfg` command. It is also required to enable the `poold` service and set the `importance` property.

`capped-cpu` Resource Property

The `capped-cpu` resource property provides an absolute fine-grained limit on the amount of CPU resources that can be consumed by a project or a zone. When used with processor sets, CPU caps limit CPU usage within a set. The `capped-cpu` resource has a single property, `ncpus`, that is a positive decimal with two digits to the right of the decimal. This property corresponds to units of CPUs. The resource does not accept a range. When specifying `ncpus`, a value of 1 means 100 percent of a CPU. A value of 1.25 means 125 percent, because 100 percent corresponds to one full CPU on the system.

The `capped-cpu` resource and the `dedicated-cpu` resource are incompatible.

Scheduling Class

Use the Fair Share Scheduler (FSS) to control the allocation of available CPU resources among zones based on their importance. The importance is expressed by the number of CPU shares assigned to each zone. Even when FSS is not used to manage CPU resource allocation, it is still possible to configure the zone's scheduling class to FSS to set shares on projects within the zone.

If `cpu-shares` is not set for a zone, the zone uses the system default scheduling class. When the `cpu-shares` property is explicitly set, FSS is the default scheduling class for that zone. However, the preferred way to use FSS in this case is to set FSS to be the system default scheduling class with the `dispadmin` command. That way, all zones benefit from getting a fair share of the system CPU resources.

Physical Memory Control and the `capped-memory` Resource

The `capped-memory` resource property sets limits for physical, swap, and locked memory. Each limit is optional. Set the memory cap for a non-global zone by configuring the `physical` property of the `capped-memory` resource. Configure the `swap` and `locked` properties to specify the swap space and the amount of memory to be locked for the zone.

Consider locking memory if the applications that run inside a zone are known to lock significant amounts of memory.

Zone-Wide Resource Controls

Zone-wide resource controls limit the total resource usage of all process entities within a zone. These limits are specified by using the `zonecfg` command.

The `zone.cpu-cap` resource control puts an absolute limit on the amount of CPU resources that can be consumed by a zone. A value of 100 means 100 percent of one CPU. A value of 125 is 125 percent because 100 percent corresponds to one full CPU on the system when using CPU caps.

The `zone.cpu-shares` resource control limits the number of FSS CPU shares for a zone. CPU shares are first allocated to the zone and then further subdivided among projects within the zone, as specified in the `project.cpu-shares` entries.

The `zone.max-locked-memory` resource control limits the amount of locked physical memory available to a zone. The allocation of the locked memory resource across projects within the zone can be controlled by using the `project.max-locked-memory` resource control.

The `zone.max-swap` resource control limits swap space consumed by user process address space mappings and `tmpfs` mounts within a zone.

The `zone.max-msg-ids`, `zone.max-sem-ids`, `zone.max-shm-ids`, and `zone.max-shm-memory` resource controls are used to limit System V resources used by all processes within a zone. The allocation of System V resources across projects within the zone can be controlled by using the equivalent resource controls defined for projects.

Refer to the "Setting Zone-Wide Resource Controls" section of Oracle Solaris Containers--Resource Management and Oracle Solaris Zones for the complete list of zone-wide resource controls.

The following example shows the steps involved in creating a whole-root exclusive-IP non-global zone. The CPU, memory caps, and zone-wide resource controls are demonstrated in the example. The example also shows how to explicitly set the scheduling class, and it adds the ability to use DTrace functionality that requires the `dtrace_proc` and `dtrace_user` privileges in the zone.

```
# dladm show-dev
e1000g0         link: up        speed: 1000  Mbps       duplex: full
e1000g1         link: up        speed: 1000  Mbps       duplex: full

/* preparatory steps for zone creation */

# mkdir -p /zones/webserver
# chmod 700 /zones/webserver

/* create a "whole root" local zone */

# zonecfg -z webserv
webserv: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:webserv> create
zonecfg:webserv>
zonecfg:webserv> set zonepath=/zones/webserver
zonecfg:webserv> set autoboot=true
zonecfg:webserv> set ip-type=exclusive
zonecfg:webserv> set scheduling-class=TS
zonecfg:webserv>
zonecfg:webserv> add net
zonecfg:webserv:net> set physical=e1000g1
zonecfg:webserv:net> end
zonecfg:webserv>
```

```
zonecfg:webserv> remove inherit-pkg-dir dir=/usr
zonecfg:webserv> remove inherit-pkg-dir dir=/sbin
zonecfg:webserv> remove inherit-pkg-dir dir=/lib
zonecfg:webserv> remove inherit-pkg-dir dir=/platform
zonecfg:webserv>
zonecfg:webserv> add rctl
zonecfg:webserv:rctl> set name=zone.max-lwps
zonecfg:webserv:rctl> add value (priv=privileged,limit=6400,action=deny)
zonecfg:webserv:rctl> end
zonecfg:webserv>
zonecfg:webserv> add dedicated-cpu
zonecfg:webserv:dedicated-cpu> set ncpus=32
zonecfg:webserv:dedicated-cpu> set importance=1
zonecfg:webserv:dedicated-cpu> end
zonecfg:webserv>
zonecfg:webserv> add capped-memory
zonecfg:webserv:capped-memory> set physical=8193m
zonecfg:webserv:capped-memory> end
zonecfg:webserv>
zonecfg:webserv> set limitpriv="default,dtrace_proc,dtrace_user"
zonecfg:webserv>
zonecfg:webserv> verify
zonecfg:webserv> commit
zonecfg:webserv> exit
```

Install, boot and configure the network services along with other essential services for proper operation of the new non-global zone.

```
/* install newly configured non-global zone */

# zoneadm -z webserv install

/* boot newly created non-global zone */

# zoneadm -z webserv boot

/* configure the IP address and the network services  */

# zlogin -C -e [ webserv

/* check the state of the newly created/configured non-global zone */

% zoneadm list -cv
```

```
ID NAME             STATUS     PATH                           BRAND    IP
 0 global           running    /                              native   shared
 - webserv          running    /zones/webserver               native   excl
```

Dynamic Reconfiguration of System Resources Assigned to a Zone

It is possible to dynamically change the quantity of a system resource, such as CPU, memory, and network bandwidth assigned to an Oracle Solaris Container. To change the quantity of a resource, use the `prctl` command.

For example, use the `prctl` command to specify a new value for `cpu-shares`, as shown below. Note that the limits specified through the `prctl` command are not persistent. The limits are in effect only until the system is rebooted.

```
# prctl -i idtype -n zone.cpu-shares -r -v value
```

`idtype` is either the zone name or the zone ID. *`value`* is the new value. Reconfiguration must be performed from the global zone.

For the complete list of resources that can be dynamically reconfigured, check the "Setting Zone-Wide Resource Controls" section of Oracle Solaris Containers--Resource Management and Oracle Solaris Zones.

Branded Zones

By default, a non-global zone has the same characteristics as the operating system running in the global zone, which is Oracle Solaris 10 or a later release. These native non-global zones and the global zone share their conformance to standards, runtime behavior, command sets, and performance traits in common.

It is also possible to run any supported operating system inside a non-global zone with the help of the branded zone (BrandZ) framework. For example, the non-global zone can emulate another version of the Oracle Solaris operating system or Linux.

In other words, branded zones are zones that are capable of emulating user environments from operating systems other than Oracle Solaris 10. Branded zones achieve this by emulating the non-native operating systems' system calls. Brands are named after the operating systems whose system calls they emulate. For example, there are solaris8 and solaris9 brands that allow Oracle Solaris 10 zones to host Solaris 8 and Solaris 9 operating environments, respectively. Zones that host native Solaris user environments (that is, zones that do not emulate system calls) are native-branded zones, which are often simply called "native" zones.

Solaris 8 and 9 Containers

With Solaris 8 Containers and Solaris 9 Containers, you can simplify installation and maintenance requirements and reduce hardware by retiring older, less efficient systems and installing newer hardware systems with Oracle Solaris 10 and Containers hosting the Solaris 8 and Solaris 9 applications.

Solaris 8 Containers provide the Solaris 8 OS runtime environment on the SPARC platform creating a virtual environment for hosting Solaris 8 applications. Similarly, Solaris 9 Containers provide the Solaris 9 runtime environment. By taking advantage of existing Solaris Containers features, the Solaris 8 and Solaris 9 virtual environments bring all the benefits of Oracle Solaris 10 to applications and services so that they can do the following:

•Be cloned many times for rapid deployment

•Be migrated to remote hosts

•Control CPU and memory resources with Oracle Solaris Resource Manager

•Take advantage of key benefits of Oracle Solaris 10

It is possible to run Solaris 8, Solaris 9, and Oracle Solaris 10 applications on the same system. The same restrictions apply as with normal Oracle Solaris Containers with regard to having enough CPU, memory, I/O bandwidth, and so on.

Benefits of Using Solaris 8 Containers and Solaris 9 Containers

For customers with legacy Solaris 8 or Solaris 9 environments, Solaris 8 Containers and Solaris 9 Containers help to consolidate one or many existing Solaris 8 or Solaris 9 systems onto newer, cost-effective, and powerful Sun SPARC Enterprise systems, such as the T-series and M-series servers. You can then transition these environments to Oracle Solaris 10 at your own pace while removing the dependency on Solaris 8 or Solaris 9 support.

Oracle VM Server for SPARC

Oracle VM Server for SPARC (previously called Sun Logical Domains) provides built-in no-cost virtualization capabilities in Oracle's Sun SPARC Enterprise T-series servers. Oracle VM Server for SPARC provides the ability to split a single physical server into multiple independent virtual servers. It abstracts the hardware and can expose or hide various resources allowing the creation of resource partitions that can operate as discrete systems with virtual CPU, memory, and I/O devices. Hence, each domain or virtual machine has its own set of resources and runs its own independent instance of Oracle Solaris.

By careful architecture, an Oracle VM Server for SPARC environment can help achieve greater resource usage, better scaling, and increased security and isolation.

The Oracle VM Server for SPARC software depends on particular Oracle Solaris versions, particular software patches, and specific versions of system firmware. For the exact requirements, check "Required and Recommended Oracle Solaris OS" in the Oracle VM Server for SPARC Release Notes.

Securely isolated from one another, logical domains are still able to amortize various resources (CPUs, memory, networking, and storage) with the flexibility to change resource amounts and configurations on demand. By combining multiple separate systems into discrete, logical domains, you can increase system usage levels and reduce the real estate footprint.

Oracle VM Server for SPARC fits somewhere in the middle of Oracle Solaris Containers and Dynamic Domains. It offers isolation between the various domains, and this isolation is achieved through a firmware layer called the Hypervisor.

Oracle VM Server for SPARC provides a reliable, robust, and secure virtualization solution. Multiple guest domains provide strong workload isolation with each guest domain running an independent instance of Oracle Solaris and its own software applications. A software failure in one guest domain does not bring down other guest domains on the system. In addition, the Hypervisor, which has access to all memory and I/O devices, is not accessible from the guest domains or from even the control domain. This increases security and prevents unauthorized access of data and resources. As long as traditional OS security remains uncompromised on the server, no domain can access the content of another domain.

The lowest level of CPU resource granularity is a single virtual CPU or hardware thread, which can be assigned to only a single logical domain at any time. Hence, it is not possible for two or more logical domains to share a virtual CPU.

Only Sun SPARC Enterprise T-series servers from Oracle have support for the Oracle VM Server for SPARC technology.

From some of the experimental data, it is observed that the CPU and memory performance overhead is minimal when using Oracle VM Server for SPARC. The I/O performance impact from Oracle VM Server for SPARC software depends on the configuration and applications.

Domain Roles

A domain can have one or more of the following roles:

- **A control** *domain* creates and manages other logical domains and services by communicating with the Hypervisor. The control domain runs Logical Domains Manager software to create, manage, and map logical domains to physical resources. Only one Logical Domains Manager can run on a server.
- A *service domain* provides services, such as a virtual network switch or a virtual disk service, to other logical domains.

- An *I/O domain* has direct ownership and access to physical I/O devices, such as a PCI Express card or a network device. It can optionally share those devices to other domains by providing services.
- A *guest domain* presents a virtual machine that subscribes to services provided by service domains and it is managed by the control domain.

Setting Up a Logical Domain

The common steps for creating a Logical Domain are briefly outlined below. Refer to part 4 of this series, "Resource Management Case Study for Mixed Workloads and Server Sharing," for an example.

1. Install a supported version of Oracle Solaris 10 and apply required patches.
2. Update the firmware as required.
3. Install Logical Domain Manager software.
4. Release resources by modifying control domain resources.
5. Create a base configuration for the system to operate on.
6. Enable the virtual network terminal server daemon via SMF.
7. Define a guest domain and add system resources, including CPU and memory resources.
8. Add a virtual network device.
9. Define a virtual disk service based upon the second physical internal disk drive.
10. Add a virtual disk device from the disk service.
11. Set up boot parameters.
12. Bind the configuration.
13. Start the logical (guest) domain.
14. Install Oracle Solaris.
15. Verify the logical domain configuration.
16. Save the logical domain configuration.

Resource Management in Oracle VM Server for SPARC

Unless otherwise stated explicitly, all the resource management techniques in Oracle Solaris 10 are equally applicable in Oracle VM Server for SPARC.

CPU Resource Allocation Mechanism in Oracle VM Server SPARC

Starting with the Oracle VM Server for SPARC 2.0 release, the CPU allocation mechanism supports specifying CPU cores in addition to virtual processors.

The whole-core constraint specifies that virtual CPUs are allocated to a domain based on a specified number of CPU cores. The system must be able to allocate the specified number of cores. Otherwise, the domain fails to bind.

When a domain is assigned virtual CPUs instead of cores, the whole-core constraint is disabled. The whole-core constraint can be disabled only on an inactive domain, not on the bound or active domains.

When the whole-core constraint is set on a domain, the "maximum number of cores" constraint is automatically enabled and set to the number of cores configured when the domain is inactive. When the whole-core constraint is disabled, the maximum-core constraint is also automatically disabled. Currently, this constraint cannot be enabled independently of the whole-core constraint, and the maximum number of cores cannot be set manually.

The "core affinity hint" optimizes the assignment of virtual CPUs for better performance. This hint requests that virtual CPUs allocated to a domain come from the same CPU cores or from the fewest number of CPU cores. The system makes its best effort to honor this request. The core affinity hint is enabled by default and cannot be disabled.

The whole-core constraint and the core affinity hint address only the location of a virtual CPU on cores. They do not address the location of a core on chips or a chip on sockets. Here is an example.

```
primary# ldm add-vcpu -c 3 domX

primary# ldm list -o resmgmt domX
NAME
domX
CONSTRAINT
whole-core
max-cores=3
```

Reconfiguration in Oracle VM Server for SPARC: Adjusting CPU and Memory Resources

Logical Domains Manager provides the ability to reconfigure domains, including the primary domain, to address the ever-changing resource requirements of different workloads running in different domains. The following sections briefly focus on reconfiguring CPU and memory resources. For clarity, the examples identify the different terminal sessions by modifying the shell prompt to `primary#` and `guest#` to represent the control and guest domains, respectively.

Dynamic Reconfiguration

Dynamic Reconfiguration (DR) is the ability to add or remove resources while the operating system is running. The ability to perform dynamic reconfiguration of a particular resource type is dependent on having support in the OS running in the logical domain.

To use the DR capability, the DR daemon (`drd`) must be running in the target domain.

As of Oracle Solaris 10 9/10, DR is supported for virtual CPUs, physical memory, virtual I/O devices, and cryptographic units. DR is not supported yet for physical I/O devices.

Dynamic Reconfiguration of CPU Resources

Virtual CPUs can be dynamically reconfigured. It is possible to set the `vcpu` count to the desired number and the Oracle Solaris instance running in the guest domain will see that immediately without a guest domain reboot. The `set-vcpu` subcommand can be used from the control domain to reconfigure CPU resources. For the sake of demonstration, let's add eight additional CPUs to the guest domain, `ebizappltop`:

```
/* check the existing CPU allocation in guest domain */
guest# psrinfo -pv
The physical processor has 16 virtual processors (0-15)
  UltraSPARC-T2+ (chipid 0, clock 1596 MHz)

/* set the total #vcpu count to 24 */
primary# # ldm set-vcpu 24 ebizappltop

/* now go back to the guest domain and check the current CPU allocation */
guest# psrinfo -pv
The physical processor has 24 virtual processors (0-23)
  UltraSPARC-T2+ (chipid 0, clock 1596 MHz)
```

The whole-core constraint is fully compatible with CPU dynamic reconfiguration. If a bound or active domain is not in delayed reconfiguration mode, its number of cores cannot exceed the maximum number of cores. This maximum is set with the maximum core constraint, which is automatically enabled when the whole-core constraint is enabled. Any CPU dynamic reconfiguration operation that would not satisfy the maximum core constraint fails.

The following example dynamically reconfigures domain `domX` to use two CPU cores.

```
primary# ldm add-vcpu -c 2 domX
```

Dynamic Reconfiguration of Memory Resources

The content in this section is applicable only to those systems running Oracle Solaris 10 9/10 or later.

The Oracle VM Server for SPARC 2.0 release introduces memory dynamic reconfiguration (DR). This capacity-based feature allows adding or removing an arbitrary amount of memory to or from an active logical domain. The memory DR feature enforces 256-MB alignment on the address and size of the memory involved in a given operation. Unaligned memory that is already allocated to a domain cannot be removed by memory DR.

It is possible to perform memory DR operations on any domain. Only a single memory DR operation can be in progress in a domain at any time.

```
/* Dynamically transfer 8 GB memory from  guest domain "domA" to "domB" */

# ldm remove-memory--auto-adj 8G domA
# ldm add-memory --auto-adj 8G domB
```

Delayed Reconfiguration

In contrast to dynamic reconfiguration operations that take place immediately, delayed reconfiguration operations take effect in the following circumstances:

•After the next reboot of the OS

•After a stop and start of the logical domain

Starting with the Logical Domains Manager 1.2 software, delayed reconfiguration operations are restricted to the control domain. For all other domains, the domain must be stopped to modify the configuration unless the resource can be dynamically reconfigured.

When a delayed reconfiguration is in progress on the control domain, other reconfiguration requests for the control domain are deferred until the domain is rebooted. Also, when a delayed reconfiguration is outstanding for the control domain, reconfiguration requests for other logical domains are severely restricted and fail with an appropriate error message.

Delayed Reconfiguration of Memory Resources

The content in this section is applicable only to those systems running Oracle Solaris 10 10/09 or earlier versions.

Unlike CPU resources, memory resources cannot be dynamically reconfigured in Oracle Solaris 10 10/09 and prior versions. Hence, the delayed reconfiguration must be used to queue a change in the amount of memory allocated for the guest domain. The Oracle Solaris instance running in the guest domain must be rebooted for the allocated memory to be available for the guest domain. For the sake of demonstration, let's add four more gigabytes of memory to the guest domain, `ebizappltop`.

```
/* check the existing memory allocation in guest domain */
guest# prtconf | grep Mem
Memory size: 16384 Megabytes

/* set the total memory to 20 GB for the guest domain */
primary# # ldm set-memory 20G ebizappltop
Please perform the operation while the domain is bound or inactive

primary# ldm stop-domain ebizappltop
LDom ebizappltop stopped

primary# ldm set-memory 20G ebizappltop

primary# ldm start-domain ebizappltop
LDom ebizappltop started

/* check the memory allocation again */
# prtconf | grep Mem
Memory size: 20480 Megabytes
```

Notice the diagnostic message that advises bringing the guest domain to an inactive state to run the `set-memory` subcommand. Once the guest domain is shut down, further changes to that domain are merged into the current delayed reconfiguration operation. This reconfiguration must take effect before users are allowed to make changes, including dynamic changes, to other domains on the system.

To roll back a delayed reconfiguration change, use the `ldm` subcommand `cancel-reconf` prior to applying the change with a reboot of the guest domain.

Using Dynamic Resource Management Policies

Starting with the Logical Domains 1.3 software, system administrators can configure policies to let the resource manager automatically perform dynamic reconfiguration activities, such as adding or removing CPUs from a domain based on CPU utilization and priority. A resource management policy specifies under what conditions CPUs can be automatically added to and removed from a logical domain. At the time of this writing, only policies that govern the dynamic resource management of CPUs can be created.

The whole-core constraint is not compatible with dynamic resource management. When a dynamic resource management policy is enabled on a domain that uses the whole-core constraint, the policy is automatically disabled. The whole-core constraint remains enabled.

For example, based on the system utilization patterns, an administrator can create a "critical-hours" policy to be used during the high utilization period, as shown below.

```
primary# ldm add-policy tod-begin=10:00 tod-end=17:00 util-lower=40 util-upper=70 \
     vcpu-min=8 vcpu-max=32 attack=2 decay=1 priority=1 name=critical-hours ebizappltop
```

This policy can be interpreted in plain text as follows.

Between 10 a.m. and 5 p.m. everyday, the domain should be running with at least eight CPUs. If the CPU utilization surges beyond 70%, two CPUs are added to the domain unless the total number of CPUs exceeds 32. Similarly, if the CPU utilization is below 40%, one CPU is removed unless the total count drops below the minimum number of CPUs, which is eight. The priority property determines which policy to use if more than one policy is in effect simultaneously. A value of 1 represents the highest priority.

Refer to the Oracle VM Server for SPARC Administration Guide for detailed information.

Also see the man page of `ldm(1M)` and `drd(1M)`.

Resources

Here are resources referenced earlier in this document:

- Part 1 of this series, "Introduction to Resource Management Using Oracle Solaris Resource Manager and Oracle Database Resource Manager": http://www.oracle.com/technetwork/articles/servers-storage-admin/o11-054-intro-rm-419298.pdf
- Part 3 of this series, "Effective Resource Management Using Oracle Database Resource Manager": http://www.oracle.com/technetwork/articles/servers-storage-admin/o11-056-oracledb-rm-419380.pdf
- Part 4 of this series, "Resource Management Case Study for Mixed Workloads and Server Sharing": http://www.oracle.com/technetwork/articles/servers-storage-admin/o11-057-mixed-wl-rm-419381.pdf
- *System Administration Guide*:*Oracle Solaris Containers--Resource Management and Oracle Solaris Zones*: http://download.oracle.com/docs/cd/E19253-01/index.html
- *System Administration Guide: IP Services*: http://download.oracle.com/docs/cd/E19253-01/index.html
- *Oracle VM Server for SPARC Release Notes*: http://download.oracle.com/docs/cd/E23120_01/index.html
- *Oracle VM Server for SPARC Administration Guide*: http://download.oracle.com/docs/cd/E23120_01/index.html

And here are some additional resources:

- *Solaris Containers: Resource Management and Solaris Zones Developer's Guide*: http://download.oracle.com/docs/cd/E19253-01/index.html
- *System Administration Guide: Oracle Solaris 8 Containers*: http://download.oracle.com/docs/cd/E22645_01/index.html
- *System Administration Guide: Oracle Solaris 9 Containers*: http://download.oracle.com/docs/cd/E22645_01/index.html
- "Moving to Oracle Solaris 10": http://www.oracle.com/technetwork/server-storage/solaris/move-to-solaris10-wp-167891.pdf
- "Beginners Guide to LDoms: Understanding and Deploying Logical Domains for Logical Domains 1.0 Release" by Tony Shoumack: http://www.oracle.com/technetwork/articles/systems-hardware-architecture/beginners-vm-server-sparc-256946.pdf
- "Virtualization with Oracle Solaris 10" by Joost Pronk van Hoogeveen and Duncan Hardie: http://www.oracle.com/technetwork/server-storage/solaris/solaris-10-virtualization-167782.pdf
- "Zones and Containers FAQ: Resource Management, Performance": http://hub.opensolaris.org/bin/view/Community+Group+zones/faq#HSection3ResourceManagementPerformance
- *Solaris Internals: Solaris 10 and OpenSolaris Kernel Architecture* (second edition) by Richard McDougall and Jim Mauro (ISBN-13: 978-0131482098): http://www.amazon.com/gp/product/0131482092/
- *Resource Management* by Richard McDougall, Adrian Cockcroft, Enrique Vargas, Evert Hoodendoorn, and Tom Bialaski (ISBN-13: 978-0130258557): http://www.amazon.co.uk/Resource-Management-Blueprints-Richard-McDougall/dp/0130258555

Effective Resource Management Using Oracle Solaris Resource Manager

June 2011, Revision 1.0

Author: Giri Mandalika

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software
Engineered to Work Together