Solaris Resource Manager™

By Richard McDougall - Enterprise Engineering

Sun BluePrints™ OnLine - April 1999

http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300  fax 650 969-9131
Part No.: 806-3839-10
Revision 01, April 1999
In December 1998, Sun introduced the Solaris Resource Manager™ software, a flexible resource management package for the Solaris™ Operating Environment. The Solaris Resource Manager software provides the ability to allocate and control major system resources. It also implements administrative policies that govern the resources different users can access, and more specifically, the level of consumption of those resources that each user is permitted.

The Solaris Resource Manager software should be used when more advanced resource management and control is required, over and above that which can be achieved using products such as processor sets and dynamic reconfiguration.

For example, two workloads can be consolidated onto a single system using processor sets to manage the CPU resource by allocating 10 processors to workload A and 8 processors to workload B. This would provide processor limits for each workload. But resources are wasted if one of the workloads is not using all of its share of the processors because the spare CPU cannot be used by any other workload.

The Solaris Resource Manager software provides the following advantages over base Solaris resource control:

- Better utilization of system resources
- Dynamic control of system resources
- More flexible resource allocation policies
- Finer-grained control over resources
- Decayed usage of resources
- Accounting data for resource usage
TABLE 1 shows a summary of the complete range of the Solaris Resource Manager software capabilities, and the scope of each control.

### TABLE 1  
**Solaris Resource Manager Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Policy</th>
<th>Control</th>
<th>Measurement</th>
<th>Accounting</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage</td>
<td>Per User Id</td>
<td>yes</td>
<td>Per User Id</td>
<td>yes</td>
</tr>
<tr>
<td>Virtual Memory</td>
<td>per-user per-process</td>
<td>per-user per-process</td>
<td>per-user per-process</td>
<td>yes</td>
</tr>
<tr>
<td>No. of processes</td>
<td>yes per-process</td>
<td>yes per-process</td>
<td>yes per-process</td>
<td>yes</td>
</tr>
<tr>
<td>Max Logins</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Connect Time</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

A full overview of the Solaris Resource Manager software is available by browsing the following: <A HREF="http://docs.sun.com:80/ab2/coll.409.2/RSCMGNTADMIN/@Ab2PageView/385?DwebQuery=srm">Solaris Resource Manager Introduction</A>

### Mapping the Workload to the Inode Hierarchy

The key to effective resource management using Solaris Resource Manager software is a well designed resource hierarchy. Solaris Resource Manager software uses the Inode tree to implement the resource hierarchy.

Each node in the Inode tree maps to UID in the password database, which means that workloads must be mapped to align with entries in the password database. In some cases, additional users may need to be created to cater to the leaf nodes in the hierarchy. These special users will not actually run processes or jobs, but will act as an administration point for the leaf node.

### A Simple Flat Hierarchy

A simple hierarchy would, for example, control the processing resources of two users, Chuck and Mark. Both of these users are notorious for using large amounts of CPU at different times, and hence have an impact on each other at different times of the day. To resolve this, construct a single level hierarchy and allocate equal shares of CPU to each user.
FIGURE 1  A Simple Flat Solaris Resource Manager Hierarchy

This simple hierarchy is established using the \texttt{limadm} command to make Chuck and Mark children of root:

\begin{verbatim}
# limadm set sgroup=root chuck
# limadm set sgroup=root mark
\end{verbatim}

Now that both Chuck and Mark are children of the root share group, you can allocate resource shares against them. For example, to allocate 50 percent of the resources to each, give the same number of CPU shares to each. (There is no reason you could not allocate one share to each user to achieve the same). Use the \texttt{limadm} command to allocate the shares:

\begin{verbatim}
# limadm set cpu.shares=50 chuck
# limadm set cpu.shares=50 mark
\end{verbatim}
You can observe the changes to the inode associated with Chuck with the `liminfo` command:

```
# liminfo -c chuck
Login name: chuck Uid (Real,Eff): 2001 (-,-)
Sgroup (uid): root (0) Gid (Real,Eff): 200 (-,-)
Shares: 50 Myshares: 1
Share: 41 % E-share: 0 %
Usage: 0 Accrued usage: 0
Mem usage: 0 B Term usage: 0s
Mem limit: 0 B Term accrue: 0s
Proc mem limit: 0 B Term limit: 0s
Mem accrue: 0 B.s
Processes: 0 Current logins: 0
Process limit: 0
Last used: Tue Oct 4 15:04:20 1998
Directory: /users/chuck
Name: Hungry user
Shell: /bin/csh
Flags:
```

The fields in from the `liminfo` command are explained in the `liminfo` manual page. 
<A HREF="http://docs.sun.com:80/ab2/coll.409.2/RSCMGNTREFMAN/@Ab2PageView/886?DwebQuery=srm"></A>

A Simple Form of Batch Management

You can create a simple hierarchy to control the environment in which batch jobs are run. To do this, add an extra layer in the hierarchy, to divide the computational requirements of online and batch using the `limadm` command:

```
# limadm set sgroup=root online
# limadm set sgroup=root batch
# limadm set cpu.shares=20 online
# limadm set cpu.shares=1 batch
# limadm set sgroup=online chuck
# limadm set sgroup=online mark
```
In the example above, the `limadm` command was used to create a new leaf in the hierarchy for online and batch, and put Chuck and Mark into the online group.

![Diagram of hierarchy](image)

By using the described hierarchy, you can ensure that the online users get their share of the processor resource. Without any further changes, both Chuck and Mark will have access to 20 times the processing resource than does batch, because their UID's map to inodes that are under the online inode. You must, however, ensure that the batch processes run against the batch inode. To do this simply start the batch jobs under the batch UID.

Note that there could be situations where you want to start batch jobs as a different UNIX ®UID but still have their resources controlled by the batch inode. To do this, use the `srmuser` command to start the batch job against the batch inode:

```
# srmuser batch /export/database/bin/batchjob
```

For further information on batch management and batch management tools, refer to the Workload Management chapter in the *Resource Management* BluePrint

### Consolidation

Since the Solaris Resource Manager software is a key component of batch management, we will discuss how Solaris Resource Manager would be used to implement portions of workload consolidation.
The Solaris Resource Manager software allows systems resources to be allocated at a system-wide level, sometimes in proportion with business arrangements of machine allocation between departments. An additional layer in the Solaris Resource Manager hierarchy is used to achieve this.

Managing Web Servers

The Solaris Resource Manager software can be used to manage resources on web servers by controlling the amount of CPU and virtual memory. Three basic topologies are used on systems hosting web servers.

Resource Managing a Consolidated Web Server

A single web server can be managed by controlling the amount of resources that the entire web server can use. This would be useful in an environment where a web server is being consolidated with other workloads. This most basic form of resource management simply prevents other workloads from affecting the performance of the web server, and vice versa.
In the above example, the Web server is allocated 20 shares. This means that it is guaranteed at least 20 percent of the processor resources if the database place excessive demands on the processor.

In addition, if a CGI-BIN process in the web server runs out of control with a memory leak, the entire system will not run out of swap space. Only the web server will be affected.

**Fine Grained Resource Management of a Single Web Server**

There are often requirements to use resource management to control the behavior within a single web server. For example, a single web server can be shared between many users, each with their own CGI-BIN programs.

An error in a CGI-BIN program could cause the entire web server to run slow, or in the case of a memory leak could even bring down the web server. To prevent this from happening, use the per-process limits.
Resource Management of Multiple Virtual Web Servers

Single machines are often used to host multiple virtual web servers. In such cases, there are multiple instances of the httpd web server. There is also greater opportunity to exploit resource control using the Solaris Resource Manager software.

It is possible to run each web server as a different UNIX UID by setting a parameter in the web server configuration file. This effectively attaches each web server to a different lnode in the Solaris Resource Manager software hierarchy.
For example, the Solaris web server has the following parameter in the configuration file, /etc/http/httpd.conf:

```
# Server parameters
server {
    server_root                   "/var/http/"
    server_user                   "webserver1"
    mime_file                     "/etc/http/mime.types"
    mime_default_type             "text/plain"
    acl_enable                    "yes"
    acl_file                      "/etc/http/access.acl"
    acl_delegate_depth            3
    cache_enable                  "yes"
    cache_small_file_cache_size   8 # megabytes
    cache_large_file_cache_size   256 # megabytes
    cache_max_file_size           1 # megabytes
    cache_verification_time       10 # seconds
    comment                       "Sun WebServer Default Configuration"
}

# The following are the server wide aliases
map /cgi-bin/                    "/var/http/cgi-bin/"      cgi
map /sws-icons/                  "/var/http/demo/sws-icons/"
map /admin/                      "/usr/http/admin/

# To enable viewing of server stats via command line,
# uncomment the following line
map /sws-stats dummy             stats
}
```

**FIGURE 6** Solaris Web Server Parameter File

By configuring each web server to run as a different UNIX UID you can set different limits on each web server. This is particularly useful for controlling and accounting for resource usage on a machine hosting many web servers.

You can make use of most or all of the Solaris Resource Manager resource controls and limits as follows:

- **Shares [cpu.shares]** Proportionally allocate resources to the different web servers.

- **Mem limit [memory.limit]** Limit the amount of virtual memory that the web server can use. This will prevent any single web server from causing another one to fail.
Proc mem limit [memory.plimit] Limit the amount of virtual memory a single cgi-bin process can use. This will stop any cgi-bin process from bringing down its respective web server.

Process limit [process.limit] The maximum total number of processes allowed to attach to a web server. This will effectively limit the number of concurrent cgi-bin processes.

Creating a Policy on Executable Name

In some cases it may be useful to construct a hierarchy that allocates resources based on the application each user is executing. Use the srmuser command and wrappers around the executables.

Create a different point in the hierarchy for each application type. Then authorize users to switch inodes as they execute each application.

![Hierarchy Diagram](image)

**FIGURE 7** A Hierarchy Allowing Policies by Executable Name
The simple hierarchy in the above example is established using the `limadm` command and makes Chuck and Mark children of the users group, and app1 and app2 are made children of the applications group:

```bash
# limadm set sgroup=root users
# limadm set sgroup=users chuck
# limadm set sgroup=users mark

# limadm set sgroup=root apps
# limadm set sgroup=apps app1
# limadm set sgroup=apps app2
```

Give both Chuck and Mark permission to switch lnodes on their child processes. This allows you to attach `/usr/bin/app1` to the app1 lnode and `/usr/bin/app2` to the app2 lnode:

```bash
# limadm set flag.admin=s chuck
# limadm set flag.admin=s mark
```

Allocate shares to both applications in the desired proportions:

```bash
# limadm set cpu.shares=20 app1
# limadm set cpu.shares=1 app2
```

Now use the `srmuser` command to create a simple wrapper script to set the appropriate lnodes upon execution of the applications:

```bash
#!/bin/sh
#
# Execute /usr/bin/app1 and attach it to the app1 lnode
#
/usr/srm/bin/srmuser app1 /usr/bin/app1
```

**Author’s Bio: Richard McDougall**

Richard has over 11 years of UNIX experience including application design, kernel development and performance analysis, and specializes in operating system tools and architecture.