SUN BLUEPRINTS APPROVED

Solaris Resource Manager™ Decay and Scheduler Parameters

By Richard McDougall - Enterprise Engineering

Sun BluePrints™ OnLine - April 1999

Sun microsystems

ENTERPRISE
ENGINEERING

`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.: 806-3841-10
Revision 01, April 1999

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Please
Recycle

Adobe PostScript

Solaris Resource Manager™ Decay and Scheduler Parameters

This article examines how the different Solaris Resource Manager™ software scheduler can effect the allocation of resource to processes. In this article, we look at how we go about setting those parameters and what the effect this has on different workloads.

The default values for usage decay and process priority decay are suitable in most situations, since they include usage history over a relatively large window (about 1 minute) yet stop any one process from completely monopolizing the CPU for short periods. The default parameters particularly suit interactive workload environments, where different shares can be given to users without affecting keyboard response. Different behaviors can be achieved using different decay factors for other workloads, such as HPC or batch environments.

Decay and scheduler parameters are global, and may be set with the srmadm command, which must be run as root. The parameter changes take effect immediately and do not require a reboot or restarting of any srm processes.

```
# srmadm set usagedecay=240
```

The full list of parameters that can be set with the Solaris Resource Manager software command are shown in TABLE 1.

TABLE 1 Solaris Resource Manager™ Scheduler Parameters

Decay Parameter	Description
delta[=seconds]	The run interval for the Solaris Resource Manager™ CPU scheduler. This is the time interval that elapses between recalculations of the normalized usages of all active users. The normalized usage affects the priorities of a user's processes, so larger values of delta effectively reduce the short-term responsiveness of the scheduler.
maxusage[=float]	The upper bound for CPU usages used in the priority calculation. Users with usages larger than this will use this value for their priority calculation. This prevents users with high CPU usages from skewing the priorities of other users.
usagedecay[={seconds | hours{h}}]	The decay rate for users' usages, expressed as a half-life in seconds. The optional suffix character h may be used to specify the value in hours.
pridecay[={seconds | hours{h}}]	The decay rate for the priorities of processes with normal and maximum nice values respectively, expressed as half-lives. The rates for other nice values are interpolated between these two and extrapolated down to minimum nice. The second value must be greater than the first.
limshare[=y,n]	When this parameter is enabled, the Solaris Resource Manager™ CPU scheduler applies its priority ceiling feature to limit all users' effective shares to prevent extremely low-usage users from briefly acquiring almost 100 percent of CPU. The enabled state is recommended. The rate of CPU service for a user is roughly inversely proportional to the user's usage. If users have not been active for a very long time, then their usage decays to near-zero. When such a user logs in (or the lnode becomes active in any way), then, for the duration of the next run interval, the user's processes could have such high priority that they monopolize the CPU. Enabling the limshare scheduling flag causes the scheduler to estimate the effective share that an lnode will receive before the next run interval. If the result exceeds the user's assigned entitlement by a given factor (see maxushare), then the user's normalized usage is readjusted to prevent this.
maxushare[=float]	If the limshare scheduling mode is enabled, the maximum effective share an individual user can have is limited to float times their allocated share. Imaxushare must not be set less than 1.0, and the default is 2.0.

Changing the Default Process Priority Decay

The default configuration of decay parameters is a trade-off between process priority decay and usage decay to create an environment suitable for the short-term response required for interactive users. Sometimes the default parameters can provide incorrect results.

Process priority decay times have trade offs at each extreme. A long process priority decay causes strict CPU allocation between processes. But it can cause a process to be marooned or starved of CPU if one user has priority over another. A short decay implements a fairer scheme where processes are biased but not starved. The latter is represented by the Solaris Resource Manager™ software defaults, which means that when a user starts a large number of processes it can influence the amount of CPU the user is apportioned.

Consider an example where user 1 is given 99 shares, and user 2 is given 1 share. User 1 runs one CPU-bound process and user 2 runs 10 CPU bound processes. One would expect the Solaris Resource Manager software to allocate 99 shares to user 1 and one share to user 2. But in reality, the rapid process priority decay tries to prevent marooning and gives each process a small amount of CPU. Thus user 1 gets closer to 80 percent and user 2 gets 20 percent.

```
PID USERNAME THR PRI NICE  SIZE   RES STATE   TIME    CPU COMMAND
 1599 user1      1  59    0  896K  560K cpu/0   0:31 80.44% t1spin
  425 user2      1  15    4  896K  560K run     0:28  1.50% spin
  460 user2      1  31    4  896K  560K run     0:24  1.37% spin
  465 user2      1  26    4  896K  560K run     0:24  1.33% spin
  462 user2      1  26    4  896K  560K run     0:24  1.32% spin
  464 user2      1  34    4  896K  560K run     0:23  1.32% spin
  451 user2      1  34    4  896K  560K run     1:03  1.32% spin
  459 user2      1  34    4  896K  560K run     0:25  1.32% spin
  466 user2      1  34    4  896K  560K run     0:23  1.32% spin
  444 user2      1  34    4  896K  560K run     0:26  1.31% spin
  446 user2      1  34    4  896K  560K run     0:26  1.31% spin
```

The default process priority decay parameters decay processes over a short period (two seconds). This means each process will decay fast enough to get some CPU. By changing the scheduler to use a longer decay parameter you can cause the scheduler to enforce strict share allocation, which will maroon other processes.

```
# srmadm set pridecay=60,120

# srmadm show -V 3
Scheduling flags = -share, -limits, -adjgroups, -limshare, -
fileopen
Idle lnode = root
Lost lnode = NONEXISTENT

Usage decay rate half-life = 120.0 seconds,
(0.977159980000 - 4 second units, 0.999942239403 - 0 Hz units),

max. users                 = 12
active users               = 0
active groups              = 0
scheduler run rate         = 4 seconds
number of configured lnodes = 0

Process priority decay rate biased by "nice":-
   high priority (nice -20) 0.9825 (half-life  39.2 seconds)
average priority (nice   0) 0.9885 (half-life  60.0 seconds)
    low priority (nice  19) 0.9942 (half-life 120.0 seconds)
```

By making the priority decay time very long you cause the scheduler to allocate CPU according to the shares. User 1 gets 99 shares and user 2 gets one.

```
PID USERNAME THR PRI NICE  SIZE   RES STATE   TIME    CPU COMMAND
1599 user1      1  59    0  896K  560K cpu/0   0:31 99.10% t1spin
 460 user2      1  31    4  896K  560K run     0:24  0.07% spin
 465 user2      1  26    4  896K  560K run     0:24  0.03% spin
 462 user2      1  26    4  896K  560K run     0:24  0.02% spin
 464 user2      1  34    4  896K  560K run     0:23  0.02% spin
 451 user2      1  34    4  896K  560K run     1:03  0.02% spin
 459 user2      1  34    4  896K  560K run     0:25  0.02% spin
 466 user2      1  34    4  896K  560K run     0:23  0.02% spin
 444 user2      1  34    4  896K  560K run     0:26  0.01% spin
 446 user2      1  34    4  896K  560K run     0:26  0.01% spin
 425 user2      1  15    4  896K  560K run     0:28  0.00% spin
```

Changing the Default User Usage Decay

The default usage decay parameters are appropriate for consolidating typical commercial workloads such as OLTP, Batch, Web servers and so on. However they may be inappropriate for timeshare compute environments such as universities and HPC environments. Where the usage history is decayed over a period of minutes, it means that over the period of an hour or a day a user may overconsume their share of the machine.

A larger usage decay time will resolve such a problem and must be set on a global basis. The decay time can be set in hours or seconds.

```
# srmadm set usagedecay=4h

# srmadm show -V 3

Usage decay rate half-life = 4.0 hours,
(0.999807477651 - 4 second units, 0.999999518648 - 100 Hz units)
```

Note that there are side effects of setting a large usage decay. Users can over consume their share without knowing it. For example, a user can use 100 percent of the CPU for a sustained period when there are no other users on the system. Suppose that a user was granted 5 percent of the system but used 100 percent for two hours when no other users were logged in. Then, when other users log in the first user will almost stop until his usage is decayed.

Changing the Maximum User Share

Another interesting factor is the maximum user share clamp. This limits the maximum amount of share a user can get in a CPU constrained environment. The `maxushare` parameter, by default, limits users from exceeding 2.0 times their share. This parameter does not affect users in a case where each of two users is given 50 percent of the system, allowing either to swing to 100 percent. If you repeated the example with different shares, the results would be quite different. FIGURE 1 shows an example where user 1 is given 90 shares and user 2 is given 10 shares. Notice that even though user 2 has had no usage history and should be able to use 100 percent for a short duration, only 20 percent is realized.

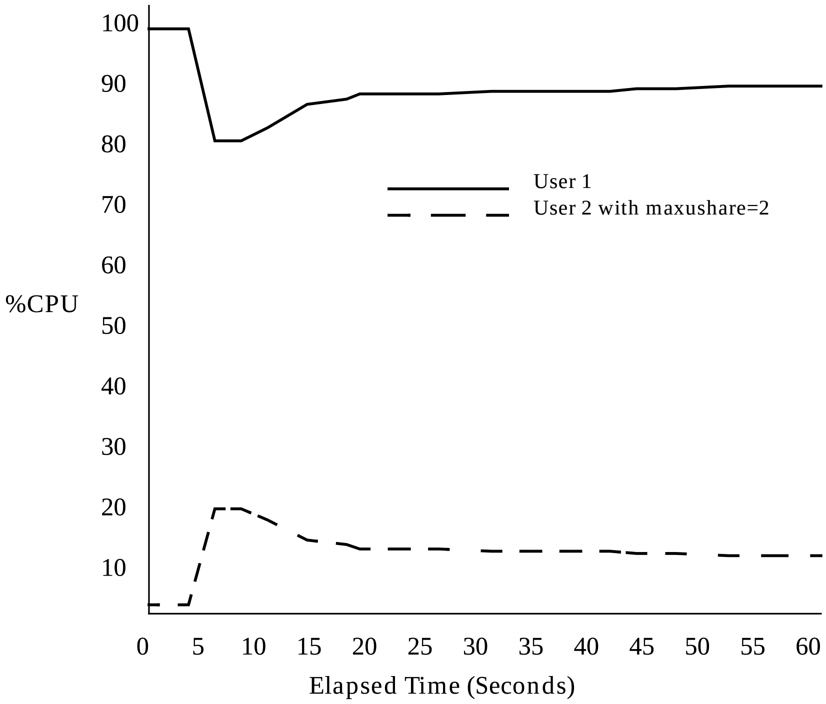

FIGURE 1 The Effect of Maximum User Share Clamping (default maxushare=2)

You can set the maximum share clamp in two ways: either adjust the maxushare parameter, or completely remove the clamp. Both these parameters are global.

```
# srmadm set maxushare=10

or

# srmadm show limshare
yes
# srmadm set limshare=n

# srmadm show limshare
no
```

If you change the maxushare parameter to 10, you now let user 2 use up to 10 times its share allocation or 100 percent. FIGURE 2 shows the effect of setting maxushare higher.

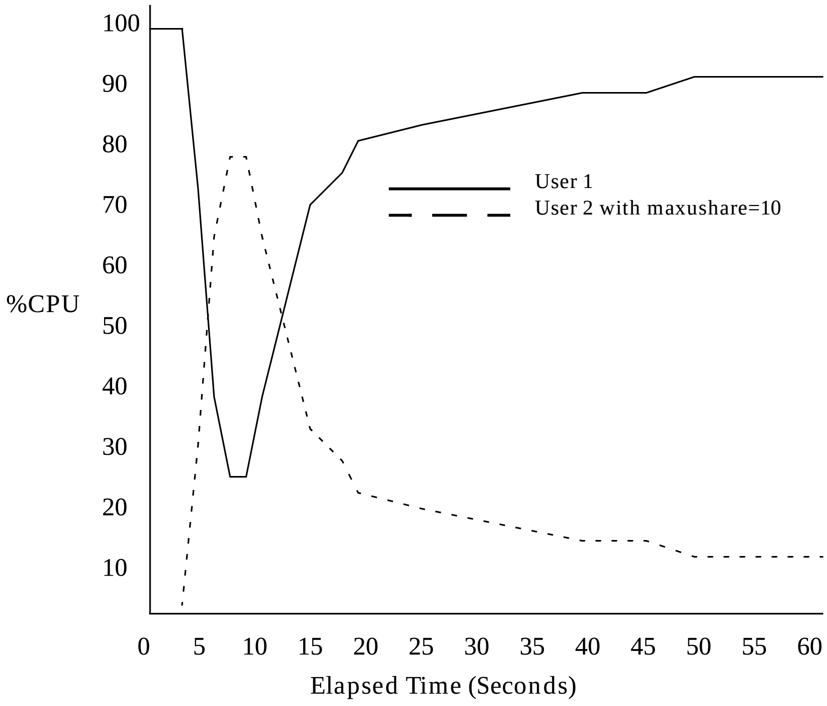

FIGURE 2 The Effect of Setting maxushare to 10

Another important factor is that the maxushare clamp works for groups when the group scheduler is enabled. This can sometimes provide unexpected results. For example, a hierarchy similar to the one shown in FIGURE 3 has shares allocated at two levels: the group level (batch and interactive) and the user level (user1, user2, batch1, batch2).

If you allocate one share each to batch and interactive at the group level, and then 99 shares to batch1, one share to batch2 you would expect that with nothing else running on the system, batch1 could use 100 percent of the CPU. This is true. However, you would expect that even if batch2 launched several jobs, batch 1 would still get 99 percent of the CPU since it has 99 shares at that level. This not true because of the interaction of the group scheduler and the maxushare clamp. At the group level, there are at least three top level groups (batch, interactive, and other srm default groups such as srmother). This means that the batch group has at most one in three shares, or 33 percent. And rather than 99 out of 100 shares at its level, batch1 really has 99 / 100 * 1 / 3 = 32%. The maxushare clamp by default allows a user to use 2.0 times it's share, or 64%. The batch2 user gets 1/100 * 1/3 * 2.0 = 0.6%, but because not all the CPU is allocated (64% + 0.6%) batch1 ends up getting about 80% and batch2 about 20%.

In situations like this, you can get the correct behavior by increasing maxushare or disabling the limshare option.

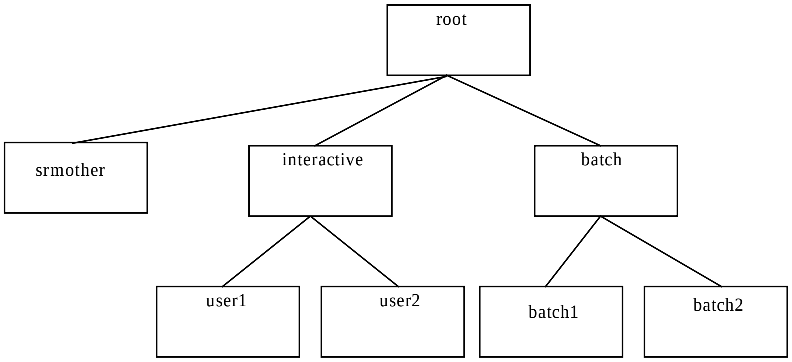

FIGURE 3 The Effect of maxushare with Group Scheduler Enabled

Changing the User Scheduler Run Rate

The scheduler run rate is the frequency at which the user scheduler is run to summize process usage per user. It is four seconds by default. The CPU overhead of doing this is minimal, and there is little reduction in overhead by slowing down the scheduler run rate. We recommend the default of four seconds remain unchanged.

Changing the Scheduler Quantum

The scheduler is configured by default for a time quantum of 11 ticks, which with the default system clock rate of 100Hz, is 110ms. The scheduler quantum can be changed, and ideally should not be a divisor of the system clock (100) to prevent possible beat effects with the per-second process scheduler and the per-four-second user scheduler. The scheduler quantum can be changed with a parameter in /etc/system or with the dispadmin command, as shown in the following examples:

```
# dispadmin -g -c SHR

(SHR) SRM Scheduler Configuration

Resolution=1000                # Resolution
Quantum=110                # Global time quantum for all processes

# cat >/tmp/shr
Resolution=1000
Quantum=60
# dispadmin -c SHR -s /tmp/shr
# dispadmin -g -c SHR

(SHR) SRM Scheduler Configuration


Resolution=1000                # Resolution
Quantum=60                 # Global time quantum for all processes
```

Setting the default share quantum via /etc/system:

```
* /etc/system
* Set SRM Scheduler Quantum
*
set shr_quantum = 59
```

Summary

The default configuration of Solaris Resource Manager software is best suited to commercial workloads. In some cases, this means that Solaris Resource Manager software behaves a little differently from expected. Most of the exceptions occur when extensive use of Solaris Resource Manager software is made to control concurrent batch style jobs with differing priorities.

Author's Bio: Richard Mc Dougall

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