



Resource Management in the Solaris™ 9 Operating Environment

Stuart J. Lawson, Global Customer Benchmarking

Sun BluePrints™ OnLine—Sept 2002



<http://www.sun.com/blueprints>

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No.816-7753-10
Revision 03, 03/07/2005
Edition: September 2002

Copyright 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California, 95054 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Sun BluePrints, SunDocs, Sun Enterprise, Solaris Resource Manager, OpenWindows, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the US and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Netscape Navigator is a trademark or registered trademark of Netscape Communications Corporation in the United States and other countries.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

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.

Copyright 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, Californie 95054 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Sun BluePrints, SunDocs, Sun Enterprise, Solaris Resource Manager, OpenWindows, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. Netscape Navigator est une marque de Netscape Communications Corporation aux Etats-Unis et dans d'autres pays.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

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 REPOUDRE 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

Resource Management in the Solaris 9 Operating Environment

The UNIX™ operating system was designed to be a multi-user operating system. A single system, or resource, can be shared by multiple parties with differing computing needs. For IT managers, this meant that they could buy powerful systems for multiple users to access common resources.

The multi-user capability was implemented from the outset by using a timeshare model that runs multiple tasks on a common resource. The granularity of the timesharing is sufficiently frequent and regular enough that the processes used to complete the task appear responsive and are usable in an interactive session.

The timeshare model is still the default multi-tasking and multi-user implementation used on nearly all general purpose UNIX platforms today, including the Solaris™ Operating Environment (Solaris OE). Its rules and logic differ between implementations, but the timeshare principal remains the same.

UNIX has come a long way since the early days. Systems running the Solaris OE are now used as an interactive compute resource and a mission-critical application server. As businesses strive to reduce costs and to increase revenues, they demand more of their servers, often consolidating multiple applications onto a single multi-user, multi-tasking, platform to achieve lower capital investment, lower running costs, and greater utilization. In doing so, they require operating system functionality that enables them to guarantee the responsiveness and performance of their core applications.

The Solaris 9 OE has been enhanced to include resource management that enables IT managers to control the timesharing system, so they can achieve their business goals in a mixed workload environment. The Solaris OE provides a resource management framework for deploying applications within this controlled environment.

Solaris 9 OE Resource Management

The Solaris OE provides systems accounting that enables the use of reactive approaches to system resource management, such as billing and access controls. These tools are helpful in a purely interactive or batch-based environment; however, they are not ideal for today's IT business models.

Sun has made available a number of products in recent years to allow more structured approaches to resource management, which are better suited to the demands of online data processing and information generation. Dynamic system domains (DSDs) were first featured on the Cray 6400 platform, that Sun was rebranded by Sun as the Sun Enterprise™ 10000 server. DSDs enable very coarse-grained approaches to managing platform level resources. Processor sets also enable coarse-grained approaches to managing system-level processing resources. In the Solaris 9 OE, the Solaris™ Resource Manager (Solaris RM) enables the resources of a single instance of the operating environment to be *shared*, in an arbitrarily fine-grained manner, amongst consolidated and/or partitioned applications and/or system users. The Solaris RM can be used in such a way that a guaranteed level of service can be given where appropriate.

With such technologies and the implementation of a corresponding service level agreement (SLA), IT departments can lease fractions of a single system to different business units at the same time. There are many advantages to this approach:

- Lower capital investment
- Higher capital utilization
- Lower overhead
- Greater profit opportunity

In the Solaris 9 OE, sharing system resources has been integrated into the kernel. System resources can be combined into meaningful sets, and applications can be executed within a framework that enables straight forward management of the available resources.

This paper includes descriptions of the three core resource management approaches present in the Solaris 9 OE.

Projects and Resource Controls

This section contains a description of how projects and resource controls are used to manage the resources on a system.

Projects

The concept of *projects* is not new in the Solaris OE. However, they now form a key component in a system resource management scenario. Projects now enable a given user process, and all of its children, to belong to a common accounting entity (that is, a *project*). With the Solaris 9 OE, user processes also inherit resource access rights and access limitations granted to a project.

The project database is maintained on a system or network either through the `/etc/project` file or through a network information service, such as NIS or LDAP. The project database defines, amongst other things, the project names, the access list, and the attributes.

Any user process executing on a system will have an associated user identity (`uid`), group identity (`gid`), and project identity (`projid`). Process attributes and capabilities are inherited from each identity and form the execution context for a task, as shown in FIGURE 1.

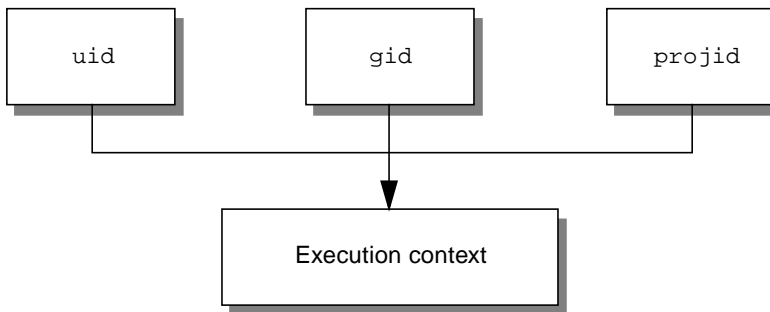


FIGURE 1 Execution Context for a Task

A Solaris 9 OE system can have a project, such as `data_mining`, to provide an exclusive accounting entity or attribute set for multiple users or processes related to the data mining service provided by that system.

Individual UNIX users can have access permission to multiple projects, but they can execute a given task (using `newtask`) only in one project. This maintains accounting data validity.

In contrast, the one-to-many relationship between users and projects is helpful on interactive systems, but it may not be relevant for a server running a well-defined application set. For this reason, it is possible to create a project that is the default for a given user, ensuring that all tasks (and processes) belonging to that user have a common accounting entity.

The example project database shown below defines two projects: database and appserver, where the database user defaults to the user.database project and the appserver user defaults to the user.appserver project. The admin user can be switched between either.

```
hostname# cat /etc/project
.
.
.
user.database:2001:Database backend:admin::
user.appserver:2002:Application Server frontend:admin::
.
.
.
```

Resource Controls

In the Solaris 9 OE, resource control attributes are set in the final field of the project database. Ignoring the use of resource pools and the fair-share scheduler for the time being, these attributes set resource consumption limits on tasks and processes.

With resource control attributes, a process or task can be given limits that it cannot exceed. These limits control the entity and protect other entities from antisocial behavior. For instance, the attributes can limit the consumption of operating environment services, such as open file descriptors or light-weight processes. In addition to controlling resource usage by an entity, the resource control framework also allows out-of-bound events to be logged in as a tool for system auditing and upstream billing of IT resources.

The following project database defines a per-process address space and a per-task, light-weight process limit on a project entity.

Note – In the following examples, the code line must be on a single line without linebreaks. Some examples show multiple lines for readability purposes only.

```
hostname# cat /etc/project
.  
.  
.  
development:2003:Developers:::task.max-lwps=(privileged,10,deny);  
process.max-addressspace=(privileged,209715200,deny)  
.  
.  
.
```

FIGURE 2 shows the relationship between the entities involved in a system using projects and resource controls.

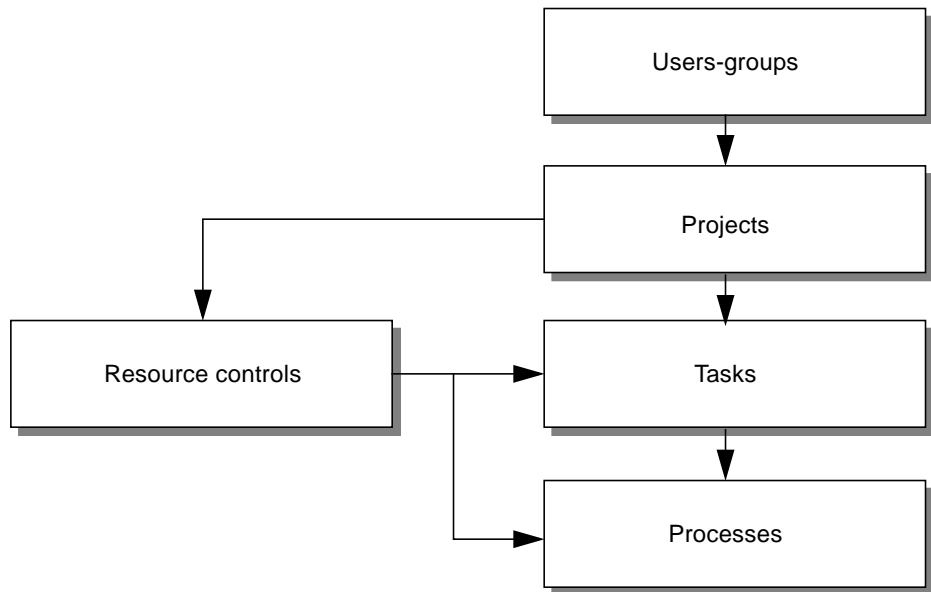


FIGURE 2 Entity Relationships Using Projects and Resource Controls

In a project database, the assignment of a project to a resource pool controls where and how tasks are allocated CPU resources.

Resource Pools

Resource pools are new to the Solaris 9 OE. They provide a framework for managing processor sets and thread scheduling classes (see “Process Scheduling and the Fair Share Scheduler” on page 8). Resource pools are flexible, dynamic reconfiguration (DR) friendly, and boot-persistent.

Processor sets group the CPUs on a system into a bounded entity, on which a process or processes can run exclusively. They cannot extend beyond the processor set, nor can other processes extend into the processor set. Although processor sets are a coarse-grained approach to resource management (currently, the best-case granularity is 1/106th of a system), they do enable you to group tasks of similar characteristics and to set a hard, upper boundary for CPU use if a throttle is desired.

Scheduling classes provide different CPU access characteristics to threads based on some algorithmic logic. You should not mix scheduling classes on any given CPU set because system performance may become erratic and unpredictable (see “Process Scheduling and the Fair Share Scheduler” on page 8). For this reason, you should use processor sets to segregate applications by their characteristics and the scheduling classes they require, perform, or cooperate best under.

The resource pool framework allows the definition of a soft processor set with a maximum and minimum CPU count requirement and a hard-defined scheduling class for that processor set (shown in FIGURE 3). This simplifies administration by automatically assigning CPUs arbitrarily to a pool (and reassigning them during a DR operation). It also avoids service unpredictability by allowing only one user-land scheduler to manage all light-wieght processes (LWPs) on that processor set.

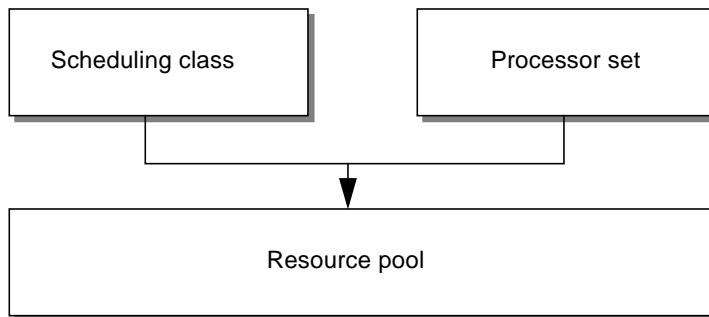


FIGURE 3 Resource Pool Framework

As an example, FIGURE 4 shows a system with two resource pools. The `batch_pool` resource pool performs batch processing that uses the default Solaris OE scheduling class (timeshare or interactive class). The `process_pool` resource pool performs manufacturing process control that uses the realtime scheduling class.

In the event of a rapid submission of multiple batch jobs, the ability of the system to respond to the process control requirements will not be affected. Additionally, in the event of a failed system component that needs to be replaced, a DR operation will not be affected.

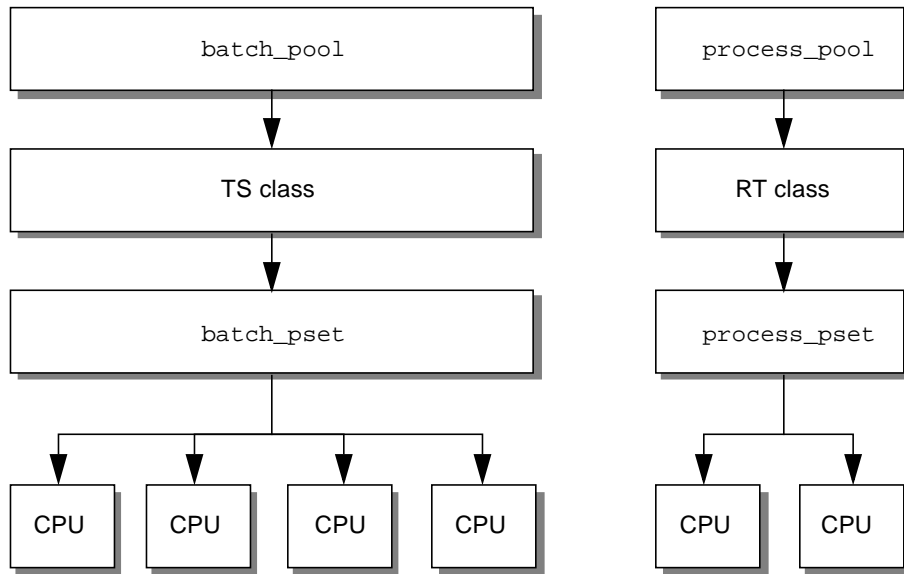


FIGURE 4 Dual Resource Pool Framework

A resource pool can be defined as an attribute for a project or multiple projects. In this way, user processes can automatically inherit a processor set from its execution context and a scheduling class. The following project database file shows how to set up project-based resource pool assignments.

```

hostname# cat /etc/project
.
.
.
batch:2001:Batch project::project.pool=batch_pool
process:2002:Process control::project.pool=process_pool
.
.
.
  
```

Process Scheduling and the Fair Share Scheduler

The Solaris OE kernel includes a simple mechanism for scheduling CPU time to threads. It decides what threads can run, when they can run, where they can run, and how long they can run. Its mechanism allocates resources based on global thread priority, which is continually re-evaluated. Consequently, higher priority threads get scheduling preference.

The thread priority range is ordinarily composed of three priority subsets that are used to schedule user threads, system threads, and interrupt threads. Any given scheduling class will manage one subset of threads. Each thread has a class structure defining this affinity. Scheduling classes are dynamically loadable kernel modules that assign thread priority based on an algorithm. Each scheduling class then maps its thread priorities onto a portion of the global scheduling priority range. The kernel then executes the threads based on the global-thread priorities. This hierarchical methodology allows multiple scheduling classes to co-exist and each class to use different scheduling rules.

Most system resource management exercises are concerned only with low priority threads (that is, user threads) because applications executing in user-space are usually under scrutiny. User threads occupy the bottom-end of the global priority range and are managed by a user-space scheduling class. System threads and interrupt threads are an essential part of the services the system provides and are usually best left alone. These threads are managed by scheduling classes that map thread priorities higher up in the global priority range. This makes them unsuitable choices for user-space applications.

Interactive and Timeshare Scheduling Classes

Prior to the release of the Solaris 9 OE, two main scheduling classes were available for user threads: interactive and timeshare. The choice of which to use was straightforward and performed by the system. The interactive class gave priority to processes that were initiated from an OpenWindows™ environment and was used to ensure GUI responsiveness. Timeshare threads did not give a priority to OpenWindow processes and was used when the system did not need a GUI process. Other than these differences, the classes are algorithmically identical and divide CPU time equally between all user threads that demand CPU time.

Fair Share Scheduler Class

In the Solaris 9 OE, an additional user-space scheduling class is available that divides CPU time equally between all of the threads in a given grouping, but not equally between groups of threads. This scheduling class is called the fair share scheduler (FSS). It assigns thread priority based on what CPU share a given group of threads has and is allowed to have relative to all of the other groups of threads.

Note – The priority assignment of the FSS is based on the individual properties of the thread *and* on the properties of all of the threads in the group to which it belongs.

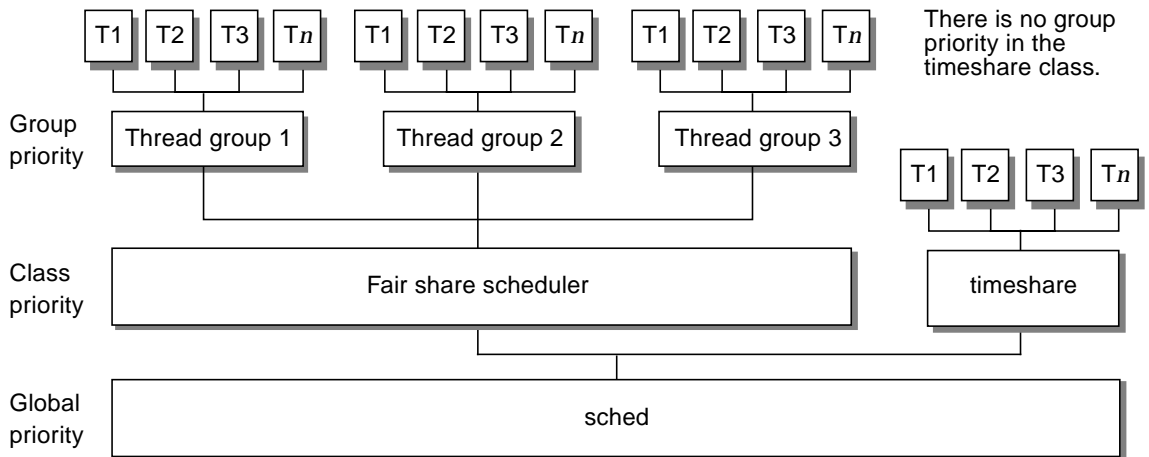


FIGURE 5 Fair Share Scheduler Priority Framework

As noted, the FSS divides CPU resources between threads based on what share of system resources are available to that thread. The concept of a CPU share is not the same as CPU percentage, shares provide equitable access to resources, *not* hard limits or bounds.

FIGURE 5 shows a simple system with three threads, each in a separate thread group (t_1 , t_2 , and t_3). Assuming that t_1 has 10 shares, t_2 has 20 shares, and t_3 has 30 shares, and that all of the threads are demanding 100 percent of the available CPU time, the FSS will ensure that t_1 receives one-sixth of the available CPU time, t_2 receives one-third of the available time, and t_3 receives one-half of the available CPU time (negating any threads running in other scheduling classes).

If t_2 stops demanding CPU time, the FSS allocates proportionately more CPU time to t_1 and t_3 so that the CPU time is fully used again. Thread groups are minimally guaranteed only their fair share of CPU time. This all happens in real-time, and CPU time is not stored up for thread groups over periods of inactivity.

The concept of shares should be familiar to users of the Solaris Resource Manager 1.x software. The concept is similar, but the implementation differs.

You can assign the FSS as the default user-space scheduling class; however, without share assignment, it would behave like the timeshare class because all of the threads would exist in one thread group. Shares can be assigned in an ad hoc fashion to running processes and can also be defined as a project attribute.

The following example project database shows how to set up two projects on a system using the FSS. One project has 20 shares, and the other has 10 shares.

```
hostname# cat /etc/project
.
.
.
user.database:2001:Database backend:admin::project.cpu-
shares(privileged,20,deny)
user.appserver:2002:Application Server frontend:admin::project.cpu-
shares(privileged,10,deny)
.
.
.
```

Resource Management Framework

With all of the elements of the Solaris 9 OE resource management framework in place, a simple and extensible framework can be used to implement system-wide resource management. Projects assign user processes resource limits through the resource control mechanism, CPU shares if the FSS is available, and a resource pool, under which the resulting processes can execute. The resource pool defines the scheduling class to be used and a group of processors, onto which the scheduling class can schedule the process threads.

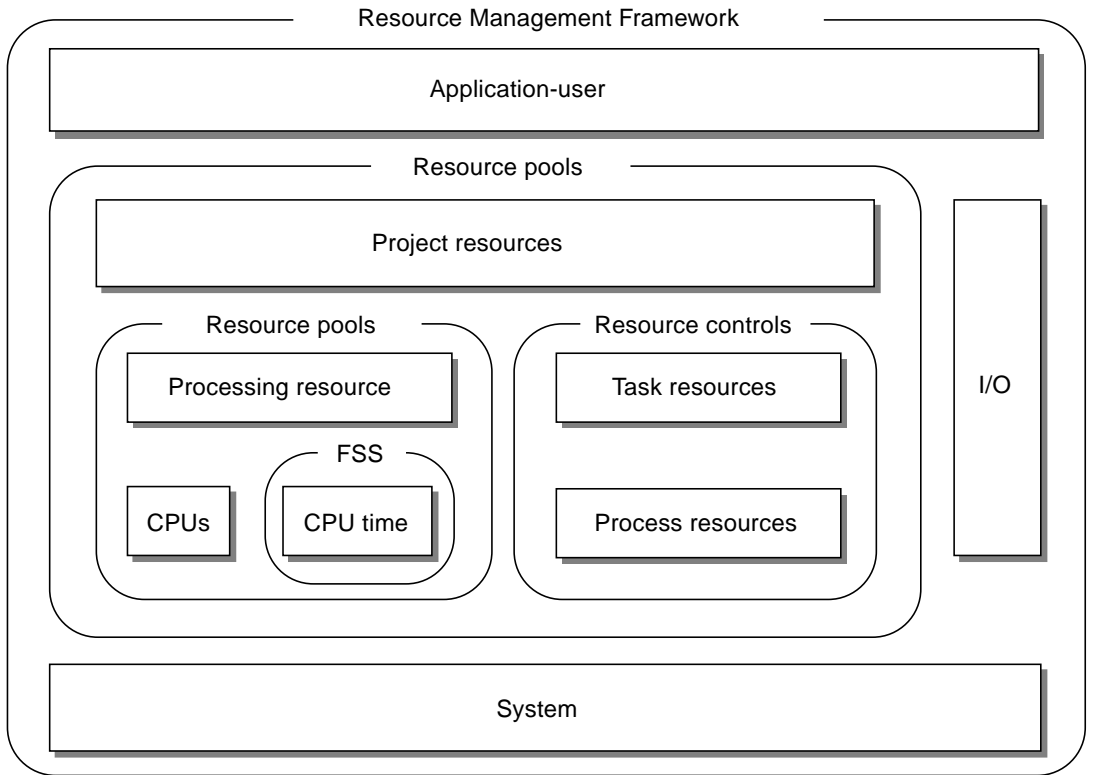


FIGURE 6 Resource Management Framework

The resource management framework (FIGURE 6) allows you to use a very fine-grained approach to system resource management, in a manner that can guarantee service levels of multiple applications running from a single system, even if the applications have differing resource utilization characteristics.

Putting It Into Action

Consider a hypothetical server consolidation project where five applications are being consolidated onto a single system. The applications have differing resource requirements, user populations, and architectures. Previously they existed on dedicated servers, each tailored to suit the requirements of the application. TABLE 1 shows the target applications and their characteristics.

TABLE 1 Target Applications and Characteristics

Name	Application	Characteristics
A	Application server	Exhibits negative scalability beyond two CPUs.
B	Database instance	Is heavily multithreaded
C	Test and development environment	Is Motif based, and hosts untested code execution
D	Transaction processing engine	Response time is paramount
E	Standalone database instance	Is heavily multithreaded, and serves multiple time zones

The following list contains the consolidation plan:

- Application A will be assigned a two-CPU processor set.
- Applications B and E will be consolidated on to a single processor set of at least four CPUs, with application E being guaranteed 75 percent of that resource.
- Application C requires the interactive scheduling class to ensure user interface responsiveness. Memory limitations will be imposed to keep the effects of antisocial processing to a minimum.
- Application D will be assigned a dedicated processor set of at least two CPUs to ensure response latency is kept at a minimum.

▼ To Set Up the Resource Management Framework

1. Edit the project database so that the following entries are in the file:

```
hostname# cat /etc/project
.
.
.
user.app_server:2001:Production Application Server::
```

These entries implement the resource controls and map users to resource pools.

2. Run the following script to configure the required resource pools:

```
hostname# cat pool.host
create system host
create pset default_pset (uint pset.min = 1)
create pset dev_pset (uint pset.max = 2)
create pset tp_pset (uint pset.min = 2)
create pset db_pset (uint pset.min = 4; uint pset.max = 6)
create pset app_pset (uint pset.min = 1; uint pset.max = 2)
create pool default_pool (string pool.scheduler="TS"; boolean pool.default =
true)
create pool dev_pool (string pool.scheduler="IA")
create pool appserver_pool (string pool.scheduler="TS")
create pool db_pool (string pool.scheduler="FSS")
create pool tp_pool (string pool.sceduler="TS")
associate pool default_pool (pset default_pset)
associate pool dev_pool (pset dev_pset)
associate pool appserver_pool (pset app_pset)
associate pool db_pool (pset db_pset)
associate pool tp_pool (pset tp_pset)
hostname# poolcfg -f pool.host
hostname# pooladm -c
```

The final command instructs the system to parse the resource pool configuration file that was created on the preceding line and to initialize the resource pools on the system.

The framework is now functional on the system and implements all of the criteria that was established. Applications do not need to be aware that a resource management framework has been implemented on the system. The development team will need to specifically execute tasks in the development project because the ACL for this project is based on a user group ID (`gid`), rather than on a user name for simplicity.

About the Author

Stuart Lawson specializes in high-end platform and Solaris OE level tuning and has over four years experience with Sun systems and the Solaris OE. Currently, he works in the Sun Microsystems Global Customer Benchmarking facility in Manchester, England, where he manages a data center composed of 350 online processors and 35 terabytes of online storage. The data center equipment is used to demonstrate commercial application performance and application tuning, using the latest Sun Microsystems platform stack.

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`