

# Solaris Resource Management

Richard Mc Dougall, Adrian Cockcroft, Brian Wong,  
Enrique Vargas, Evert Hoogendoon,  
Tom Bialaski, Jeannie Johnstone

*The push for consolidation of workloads and servers onto fewer systems creates new demands on system and performance management. Resource management allows controlled allocation of resources to different applications as required to meet service levels and performance goals. This paper is a subset of a resource management guide being developed at Sun, and depicts how Solaris Resource Manager can be used to manage Solaris resources with various application types.*

## **1.0 INTRODUCTION**

If you are working in a large datacenter environment, the following questions are likely to be familiar to you.

- The datacenter is full of systems, many of those systems are lightly used, and there are always more applications to be brought on-line. There isn't room to add more systems, and the available systems are getting more powerful, so how can we add a new application to an existing system, without impacting the level of service provided to its users?
- There are so many small server systems that they are a nightmare to manage. Each has its own custom setup, and unlike desktop machines, its hard to automate a cloned installation process. So how can we combine lots of small servers into a few big ones?
- Very large systems have been installed in the datacenter, and lots of applications are sharing their resources. How can we measure and control these applications to meet a Service Level Agreement that we have agreed with our users?
- Solaris being installed as a replacement for mainframes running MVS, how can mainframe techniques be applied to a Unix system? What is the same and what is new?
- Sun provides a base level operating environment with many facilities and several unbundled products that extend its capability. How can we tell what combinations of products work together to solve our business problems?

Resource management is an established discipline in the mainframe operations arena. As Solaris systems

are deployed in the datacenter, existing staff are trying to figure out how to take their existing management practices and apply them to these unfamiliar systems.

Resource management for Unix systems is now an established discipline, although it is in its infancy compared to common practices under MVS. IBM has a resource management product, the IBM Workload Manager for MVS. We are both trying to solve the same set of problems, so over time, Solaris will be moving to provide comparable resource management and service level management features.

In December 1998, Sun introduced a flexible resource management package for Solaris, The Solaris Resource Manager (SRM). SRM provides the ability to allocate and control major system resources. It also implements administrative policies that govern which resources different users can access, and more specifically, what level of consumption of those resources each user is permitted.

In this paper we look at how resource management is about meeting performance and service levels. We then contrast the different resource management products available within the Solaris environment and position Solaris Resource Manager. We then study in further detail how Solaris Resource Manager can be used to manage several different application scenarios.

## **2.0 Service Levels**

Computer systems are used to provide a service to their end users. System managers are responsible for the quality of this service. System and application ven-

dors provide a range of components that can be used to construct a service. A service must be available when it is needed and must have acceptable performance characteristics.

Service Level Management involves interactions between end users, system managers, vendors and computer systems. A common way to capture these interactions is via a Service Level Agreement (SLA) between the system managers and the end users. In reality there are many additional interactions and assumptions that are not often captured formally.

The interactions are shown in Figure 1. Each interaction consists of a service definition combined with a workload definition. There are many kinds of service definition and many views of the workload. The processes involved in Service Level Management include creating service and workload definitions and translating from one definition to another.

The workload definition includes a *schedule* of what work is run at different times of the day. For example, daytime interactive use, overnight batch, backup and maintenance periods. For each period, the workload mix is defined in terms of applications, transactions, numbers of users, and work rates. The service level definition includes availability and performance for *service classes* that map onto key applications and transactions. Availability is specified as maximum downtime over a period of time. Performance may be specified as response time for interactive transactions, or throughput for batch.

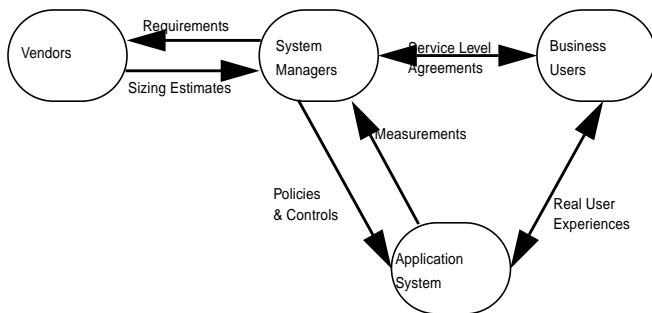


Figure 1. Service Level Management Interactions

## 2.1 Requirements

System managers establish a set of Service Level Requirements and a workload definition. The workload is defined by the managers, and is often the first description to be created. The requirements are communicated to vendors, who respond by proposing a system to meet these requirements.

## 2.2 Sizing Estimates

Vendors measure the performance of their system using generic benchmarks. They may also work with system managers to define a customer specific benchmark test. Vendors provide a sizing estimate, based on the service level requirements and workload definition. The basis of the sizing estimate may be published benchmark performance. In some cases, the measured performance on a customer-defined benchmark is used as the basis of a sizing estimate. Vendors provide reliability data for system components. They can also provide availability guarantees and performance guarantees for production systems with a defined workload, (at a price). Vendors cannot provide unqualified guarantees because there are too many application and environmental dependencies that are outside their control.

## 2.3 Service Level Agreements

System managers and end users setup a service level agreement that establishes a user-oriented view of the workload mix and the service levels required. This may take the form “95th percentile response time of under two seconds for the new-order transaction with up to 600 users on-line during the peak hour.” It is important to specify the workload (in this case the number of users at the peak period), both to provide bounds for what the system is expected to do, and to be precise about the measurement interval. Performance measures averaged over shorter intervals will have higher variance and higher peaks.

The agreed service levels could be either too demanding or too lax. The system may be quite usable and working well, but still failing its service level agreement. It could also be unusable, but still working within the agreed service levels. There needs to be a continuous process of updating and refining the SLA that is agreed between the parties involved.

## 2.4 Real User Experiences

The actual service levels experienced by users with a real workload is a subjective measure that is very hard to capture. It is quite common for problems to occur that affect parts of the system not covered explicitly by the service level agreement, or for the workload to vary from that defined in the service level agreement. One of the biggest challenges in performance management is to obtain measurements that have a good correlation with the real user experience.

### **3.0 Measurements**

The real service levels cannot always be captured directly, but you take measurements that you believe are representative of the real user experience. These measurements are then compared against the service level agreement to decide whether a problem exists. For example, downtime during the interactive shift is measured and reported. A problem could occur in the network between the users and the application system that causes poor service levels from the end user point of view but not from the system point of view. It is much easier to measure service levels inside a back-end server system than at the user interface, but it is important to be aware of the limitations of such measurements. This problem must be carefully considered when the service level agreement is made and when the service level measurements are being defined.

### **4.0 Policies and Controls**

System managers create policies that direct the resources of the computer system to maintain service levels according to the workload definition specified by the policy. This workload definition is closely related to the service level agreement workload definition, but may be modified to satisfy operational constraints. It is translated into terms that map onto system features. Example policies include "A maximum of 600 interactive users of the order entry application at any time," "order entry application has a 60 percent share of CPU, 30 percent share of network, and 40 percent share of memory," "if new-order response time is worse than its target, steal resources from other workloads that are over-achieving."

The policy works only in terms of the measurements it has available. If the wrong things are being measured, the policy will be ineffective. The policy can control resources directly or indirectly. For example, direct controls on CPU time and network bandwidth usage might be used to implement indirect controls on disk I/O rates by slowing or stopping a process.

#### **4.1 Capacity Planning and Exception Reporting**

The measured workload and service levels should be analyzed to extract trends. A capacity planning process can then be used to predict future scenarios and determine action plans to tune or upgrade systems, modify the service level agreement and proactively avoid service problems.

In cases where the measured workload from the users exceeds the agreed workload definition, or the mea-

sured service level falls short of the agreed level, an exception report is produced.

#### **4.2 Accounting and Chargeback**

The accrued usage of resources by each user or workload may be accumulated into an accounting system so that projects can be charged in proportion to the resources they consume.

### **5.0 Resource Management Products**

#### **5.1 Base Solaris**

The Solaris operating environment includes several features that provide control over certain types of resources. Some features, such as nice(1), and processor sets, are part of the basic Solaris system and allow a limited form of resource management.

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. This can be a useful feature (for example, when a user starts a low-priority batch job from an interactive login session), but it relies on the cooperation of the user. Solaris Resource Manager enforces administrative policies, even without the cooperation of the user.

#### **5.2 Processor Sets**

Processor sets were introduced in Solaris 2.6. This feature permits the administrator to divide multiprocessor systems into logical groups and permits users to launch processes into those groups. The advantage is that workloads running in one processor set are protected from CPU activity taking place in any other processor set. In some ways, this is similar to what Solaris Resource Manager does, but the two features operate on a completely different basis. Processor sets control only CPU activity. The control is at a relatively coarse-grained hardware level, because processors may belong to exactly one processor set at a time. Especially in the case of relatively small systems, the granularity may be quite high: on a 4-processor system, the minimum resource that can be assigned is 25 percent of the system.

Solaris Resource Manager has much finer-grained control; each user is allocated a share of the system. The shares can be distributed arbitrarily on a fine granularity, and the scheduler will allocate resources accordingly. For example, if 50 shares are granted, and one user has 40 of them, that user will get  $40 / 50 = 80$  percent of the resource. Similarly, if 67 total shares are

granted, a user with 57 shares will get 85 percent of the resource. In addition, Solaris Resource Manager can control resources other than CPU.

### 5.3 Dynamic System Domains

The Sun Enterprise 10000 has a feature called dynamic system domains, which permits the administrator to logically divide a single system rack into one or more independent systems, each running its own copy of Solaris. For example, a system with 32 CPUs on 8 system boards might be operated as 1 system with 16 CPUs, and 2 other systems with 8 CPUs each. In this configuration, three copies of Solaris would be running. The dynamic system domains feature also permits controlled movement of resources into and out of each of the Solaris images, thus creating a relatively coarse-grained facility for managing physical resources. (The minimum unit of inter-domain allocation is an entire system board.) Solaris Resource Manager is similar to dynamic system domains in that it provides the administrator with mechanisms to allocate resources, but it does so in very different ways. Solaris Resource Manager runs within a single instance of Solaris, and provides fine-grained administrative control to the resources in the system. Dynamic system domains divide a single piece of hardware into multiple instances of Solaris; it provides tools to manage the transfer of resources between instances of Solaris running in the same Sun Enterprise 10000 frame. Solaris Resource Manager can be run in each instance of Solaris within a Sun Enterprise 10000 system and used in conjunction with dynamic system domains.

### 5.4 Dynamic Reconfiguration

The dynamic reconfiguration feature of Sun Enterprise servers enables users to dynamically add and delete system boards, which contain hardware resources such as processors, memory, and I/O devices. The effect of a dynamic reconfiguration operation on memory has no impact on Solaris Resource Manager memory-limit checking.

### 5.5 Bandwidth Manager

Solaris Bandwidth Manager provides the means to manage your network resources to provide Quality of Service (QoS) to network users. It allows network traffic to be allocated to separate classes of service, so that 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. Service Level Agreements can be defined and translated into Bandwidth Manager controls and

policies. Tools and APIs provide the interface to monitoring, billing and accounting options.

Since almost all network links are shared by more than one user or application, the available bandwidth has to be shared. Bandwidth Management tools enable you to manage how the bandwidth is shared.

If a network link is continuously congested, the link needs to be upgraded to provide greater capacity. In many cases, however, the average load on a link is within the link capacity, and the link is only congested temporarily. Temporary congestion is sometimes predictable; for example, there are typically peaks in network use at particular time of the day or following a particular event. Other causes of temporary congestion, such as the transfer of a large file, are not possible to predict.

Solaris Bandwidth Manager enables you to manage the bandwidth used by IP traffic. It does this by:

Allocating traffic to a class based on the application type, source and destination address, URL group, or a combination, then assigning individual limits for each class. For example:

- Traffic to Engineering must have at least 50 percent of the link.
- HTTP traffic cannot exceed 10 percent of the link.

Prioritizing traffic. Some types of traffic, for example interactive traffic generated when using `telnet` or `rlogin`, need a quick response time. Solaris Bandwidth Manager lets you assign a higher priority to that traffic. Traffic that does not require a quick response time, such as a file transfer using FTP, can be assigned a lower priority.

By balancing the bandwidth allocated to different types of network traffic and the relative priorities, you can optimize your network performance. Solaris Bandwidth Manager also allows you to monitor the achieved performance of your network, and has interfaces for third party billing applications.

The current version of the product as of this writing is Bandwidth Manager 1.5, the previous version of the product was known as Bandwidth Allocator 1.0.

## **6.0 SOLARIS RESOURCE MANAGER**

Solaris resource manager is Sun's resource management extension for the Solaris operating environment and should be used when more advanced resource management and control is required.

For example, two workloads may 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. Although this would provide processor limits for each workload, resources could be potentially 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.

Solaris resource manager 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. Solaris Resource Manager Functions

	Policy	Control	Measurement	Accounting
CPU Usage	Per User Id	yes	Per User Id	yes
Virtual Memory	per-user per-process	per-user per-process	per-user per-process	yes
No. of processes	yes	yes	yes	yes
Max Logins	yes	yes	yes	yes
Connect Time	yes	yes	yes	yes

## 6.1 Solaris Resource Manager Policies

Solaris Resource Manager is built around a fundamental addition to the Solaris kernel called an Inode (limit node). Inodes correspond to UNIX UIDs, and may represent individual users, groups of users, applications, and special requirements. The Inodes are indexed by UID and are used to record resource allocations policies and accrued resource usage data by processes at the user, group of users, and/or application level.

## 6.2 Hierarchical Structure

The Solaris Resource Manager management model organizes Inodes into a hierarchical structure called the scheduling tree. The scheduling tree is organized by UID: each Inode references the UID of the Inode's parent in the tree. Each sub-tree of the scheduling tree is called a scheduling group, and the user at the root of a scheduling group is the group's header. The root user is the group header of the entire scheduling tree.

A group header can be delegated the ability to manage resource policies within the group. The Inodes are initially created by parsing the `/etc/passwd` file. An Inode administration command (`limadm(1MSRM)`) will

create additional Inodes after installation of Solaris Resource Manager and assign Inodes to parents.

The scheduling tree data is stored in a flat file database, which can be modified as required.

Though UIDs used by Inodes do not have to correspond to a system account, with an entry in the system password map, it is strongly recommended that a system account is created for the UID of every Inode. For non-leaf Inodes (those with subordinate Inodes below them in the hierarchy), it may be the case that the account associated with that Inode is purely administrative and no-one ever logs in to it. However, it is equally possible that it can be the Inode of a real user who does log in and run processes attached to this non-leaf Inode.

Note that Solaris Resource Manager scheduling groups and group headers have nothing to do with the system groups defined in the `/etc/group` database. Each node of the scheduling tree, including group headers, corresponds to a real system user with a unique UID.

## 6.3 Hierarchical Limits

If a hierarchical limit is assigned to a group header in an Inode tree (scheduling group), then it applies to the usage of that user plus the total usage of all members of the scheduling group. This allows limits to be placed on entire groups, as well as on individual members. Resources are allocated to the group header, who may allocate them to users or groups of users that belong to the same group.

## 6.4 Delegated Administration of Policies

The system administrator can set administrative privileges for any Inode, including assigning administrative privileges selectively to users. A user with hierarchical administrative privilege is called a sub-administrator. A sub-administrator may create, remove, and modify the Inodes of users within the sub-tree of which they are the group header.

Sub-administrators cannot normally alter their own limits or flags, and cannot circumvent their own flags or limits by altering flags or usages within their group.

The central administrator (or super-user) may alter the limits, usages and flags of any user, including itself. Ordinary users can be granted this privilege by setting a flag.

## 6.5 Policies by Executable Name

Resource policies that are not restricted to UIDs can be created by attaching workloads to Inodes, for example a policy based on application executable name can be created by creating Inodes for each application executable and then attaching the executable to the correct Inode.

A wrapper script may be required to implement this policy using the `srmuser` command.

## 6.6 Controls available with Solaris Resource Manager

Solaris Resource Manager provides control of the following system resources: CPU (rate of processor) usage, virtual memory, number of processes, number of logins, and terminal connect-time.

Solaris Resource Manager keeps track of each user's usage of each resource. For all resources except CPU usage, users may be assigned hard limits on their resource usages. A hard limit will cause resource consumption attempts to fail if the user allows the usage to reach the limit. Hard limits are directly enforced by either the kernel or whatever software is responsible for managing the respective resource. A limit value of zero indicates no limit. All limit attributes of the root Inode should be left set to zero.

Solaris Resource Manager progressively decays past usage so that only the most recent usage is significant. The system administrator sets a half-life parameter that controls the rate of decay. A long half-life favors even usage, typical of longer batch jobs, while a short half-life favors interactive users.

The CPU resource is controlled using the Solaris Resource Manager SHR scheduler. Users are dynamically allocated CPU time in proportion to the number of shares they possess (analogous to shares in a company), and in inverse proportion to their recent usage. The important feature of the SHR scheduler is that while it manages the scheduling of individual threads (technically, in Solaris, the scheduled entity is a light-weight process (LWP)), it also portions CPU resources between users.

Each user also has a set of flags, which are boolean-like variables used to enable or disable selective system privileges, for example, login. Flags may be set individually per user, or may be inherited from a parent Inode.

The usages, limits, and flags of a user can be read by any user, but can be altered only by users who have administrative powers.

## 6.7 Decay of Renewable Resources

Solaris Resource Manager employs a usage, limit, and decay model to control a user's rate of consumption of a renewable resource. Usage is defined as the total resource used, with a limit set on the ratio of usages in comparison to other fellow users. Decay refers to the period by which historical usage is discounted. The next resource quantum, for example, clock tick, will be allocated to the active Inode with the lowest decayed total usage value in relation to its allocated share. The decayed usage value is a measure of the total usage over time less some portion of historical usage determined by a half-life decay model.

## 6.8 CPU Resource Management

The primary advantage of the Solaris Resource Manager scheduler over the standard Solaris scheduler is that it schedules users or applications rather than individual processes. Every process associated with an Inode is subject to a set of limits. For the simple case of one user running a single active process, this is the same as subjecting each process to the limits listed in the corresponding Inode. When more than one process is attached to an Inode, as when members of a group each run multiple processes, all of the processes are collectively subject to the listed limits. This means that users or applications cannot consume CPU at a greater rate than their entitlements allow, regardless of how many concurrent processes they run. The method for assigning entitlements as a number of shares is simple and understandable, and the effect of changing a user's shares is predictable.

The allocation of the renewable CPU service is controlled using a fair share scheduler. Each Inode is assigned a number of CPU shares, analogous to shares in a company. The processes associated with each Inode are allocated CPU resources in proportion to the total number of outstanding active shares, where active means that the Inode has running processes attached. Only active Inodes are considered for an allocation of the resource, as only they have active processes running and need CPU time. As a process consumes CPU ticks, the CPU usage attribute of its Inode increases.

The scheduler regularly adjusts the priorities of all processes to force the relative ratios of CPU usages to converge on the relative ratios of CPU shares for all active Inodes at their respective levels. In this way, users can expect to receive at least their entitlements

of CPU service in the long run, regardless of the behavior of other users. The scheduler is hierarchical, because it also ensures that groups receive their group entitlement independently of the behavior of the members.

Solaris Resource Manager is a long-term scheduler; it ensures that all users and applications receive a fair share over the course of the “scheduler term.” This means that when a light user starts to request the CPU, that user will receive commensurately more resource than heavy users until their comparative usages are in line with their relative “fair” share allocation. The more you use over your entitlement now, the less you will receive in the future. Additionally, Solaris Resource Manager has a decay period, set by the system administrator, that forgets about past usage.

The decay model is one of half-life decay, where 50 percent of the resource has been decayed away within one half life. This ensures that steady, even users are not penalized by short-term, process-intensive users. The half-life decay period sets the responsiveness or “term” of the scheduler; the default value is 120 seconds. Shorter values tend to provide more even response across the system, at the expense of slightly less accuracy in computing and maintaining system-wide resource allocation. Regardless of administrative settings, the scheduler tries to prevent marooning (resource starvation) and ensure reasonable behavior, even in extreme situations.

Solaris Resource Manager will never waste CPU availability. No matter how low a user’s allocation, that user will always be given all the available CPU if there are no competing users. One of the consequences of this is that users may notice performance that is less smooth than they are used to. If a user with a very low effective share is running an interactive process without any competition, it will appear to run quickly. However, as soon as another user with a greater effective share demands some CPU time, it will be given to that user in preference to the first user, so the first user will notice a marked job slow-down. Nevertheless, Solaris Resource Manager goes to some lengths to ensure that legitimate users are not marooned and unable to do any work. All processes being scheduled by Solaris Resource Manager (except those with a maximum nice value) will be allocated CPU regularly by the scheduler. There is also logic to prevent a new user that has just logged on from being given an arithmetically “fair,” but excessively large proportion of the CPU to the detriment of existing users.

## 6.9 Virtual Memory (Per-User and Per-

## Process Limits)

Virtual memory is managed using a fixed resource model. The virtual memory limit applies to the sum of the memory sizes of all processes attached to the Inode. In addition, there is a per-process virtual memory limit that restricts the total size of the process’s virtual address space size, including all code, data, stack, file mappings, and shared libraries. Both limits are hierarchical. Limiting virtual memory is useful for avoiding virtual memory starvation. For example, Solaris Resource Manager will stop an application that is leaking memory from consuming unwarranted amounts of virtual memory to the detriment of all users. Instead, such a process only starves itself or, at worse, others in its resource group.

## 6.10 Number of Processes

The number of processes that users may run simultaneously is controlled using a fixed resource model with hierarchical limits.

## 6.11 Terminals and Login Connect-Time

The system administrator and group header can set terminal login privileges, number of logins, and connect-time limits, which are enforced hierarchically by Solaris Resource Manager. As a user approaches a connect-time limit, warning messages are sent to the user’s terminal. When the limit is reached, the user is notified, then forcibly logged out after a short grace period.

## 6.12 User Administration

The system administrator can set administrative privileges for any Inode, including assigning administrative privileges selectively to users. A user with hierarchical administrative privilege is called a sub-administrator. A sub-administrator may create, remove, and modify the Inodes of users within the sub-tree of which they are the group header.

Sub-administrators cannot normally alter their own limits or flags, and cannot circumvent their own flags or limits by altering flags or usages within their group.

The central administrator (or super-user) may alter the limits, usages and flags of any user, including itself. Ordinary users can be granted this privilege by setting a flag.

## 6.13 Processes

Every process is attached to an Inode. The `init` process is always attached to the root Inode. When processes are created by the `fork(2)` system call, they are attached to the same Inode as their parent. Processes may be re-attached to any Inode using a Solaris Resource Manager system call, given sufficient privilege. Privileges are set by root or by users with the correct administrative permissions enabled.

## 6.14 Measurement

Resource usage information is exposed to the administrators of the system, and provides two views of resource usage information: that based on each user, and a workload view of resource usage.

## 6.15 Per-user Resource Usage

A report of a individual users resource usage may be obtained by using the `liminfo` command, to find out their current usages, limits, privileges, and so on. The `liminfo` command is also of use to administrators who want to inquire on the attributes of other users.

There are a number of different report formats that can be requested, including options that make the output of `liminfo` suitable for processing by a pipeline of filters. Refer to the manual page for the `liminfo(1SRM)` command for details of the options and their meanings, and for a description of the fields that can be displayed.

## 6.16 Workload View of Resource Usage

The workload view of resource usage is based on the Inode resource hierarchy tree. Solaris resource manager does this by recording accrual of resource usage in the Inode tree.

The Inode's `cpu.accrue` attribute contains the accrued CPU usage for all Inodes within the group as well as that of the current Inode. When any of an Inode's `accrue` attributes are updated, the change is also applied to the Inode's parent (as with changes to the usage attribute) and so on up to the root Inode, so that the accrued usage at each level in the scheduling tree is the sum of the accrued usage for the Inode and the accrued usage of its children, if any.

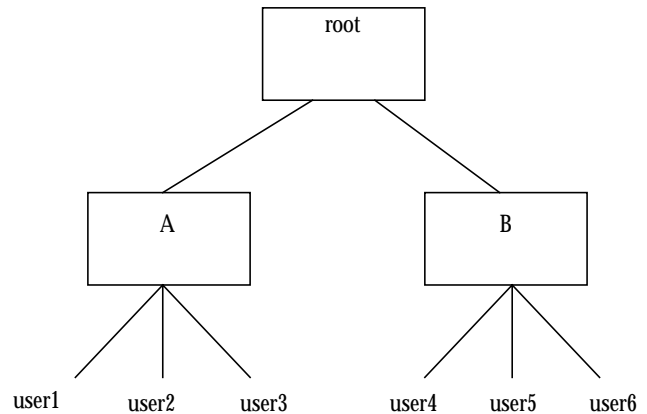


Figure 2. Accounting based on the Workload Hierarchy

For example, a system with two workloads A and B may have an Inode hierarchy with Inodes A and B reporting to the root node, and then individual users reporting into node A and B. The CPU usage of each user may be observed via the usage fields in the Inode of each user, but the sum total of CPU used for workload A is available by looking at the accrued CPU usage in the Inode for workload A.

## 6.17 System Resource Accounting Information

The Solaris Resource Manager system maintains information (primarily current and accrued resource usage) that may be used by administrators to conduct comprehensive system resource accounting. No accounting programs are supplied as part of Solaris Resource Manager but its utility programs provide a base for the development of a customized resource accounting system.

## 7.0 Workload Configuration

### 7.1 Mapping the Workload to the Inode Hierarchy

The key to effective resource management using Solaris Resource Manager is a well designed resource hierarchy. Solaris Resource Manager 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.

## 7.2 A Simple Flat Hierarchy

A simple hierarchy would be to 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 impact each other at different times of the day.

To resolve this, a single level hierarchy is constructed, and equal shares of CPU are allocated to each user.

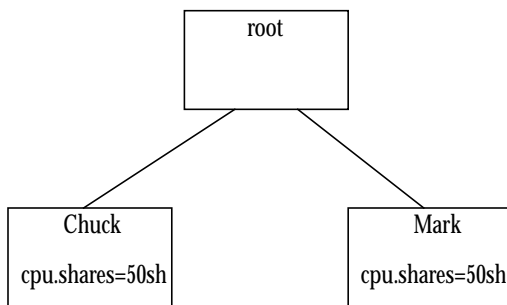


Figure 3. A simple Flat Solaris Resource Manager Hierarchy

This simple hierarchy is established using the `limadm` command to make Chuck and Mark children of root:

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

Now that both Chuck and Mark are children of the root share group, we may allocate resource shares against them. To allocate 50 per cent of the resources to each we give the same number of CPU shares to each. For simplicity's sake, we allocate 50 shares to each user. (There is no reason we could not allocate 1 share to each to achieve the same). The `limadm` command is used to allocate the shares:

```
# limadm set cpu.shares=50 chuck
# limadm set cpu.shares=50 mark
```

We 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 below. Refer to manual page on `liminfo` for more information on the fields described below. The first two lines of output from the `liminfo` command relate to aspects of the Inode UID and its position in the Inode tree.

## 7.3 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 on-line 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 this example, 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.

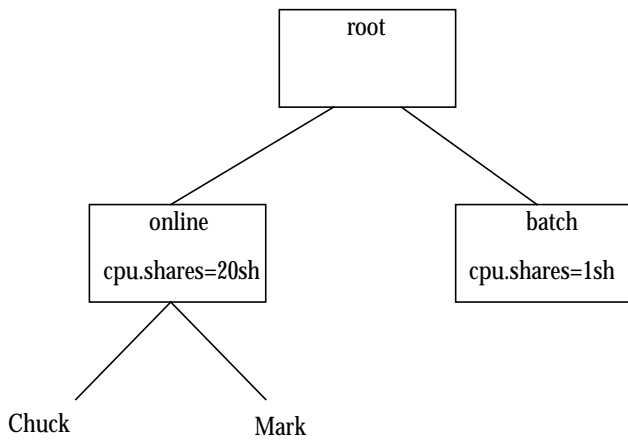


Figure 4. Creating On-line and Batch Shares with Solaris Resource Manager

By using the described hierarchy, you can ensure that the on-line 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 do, however, need to ensure that the batch processes run against the batch Inode.

You can do this by simply starting the batch jobs under the batch UID. There may however be situations where we 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 `srmsuser` command to start the batch job against the batch Inode:

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

For further information on batch management and batch management tools, refer to the Workload Management chapter.

### 7.4 Consolidation

Consolidation of workloads onto single system is discussed in detail in the Workload Management chapter. Since Solaris Resource Manager is a key component of batch management, we will discuss how Solaris Resource Manager would be used to implement portions of workload consolidation.

Solaris Resource Manager 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.

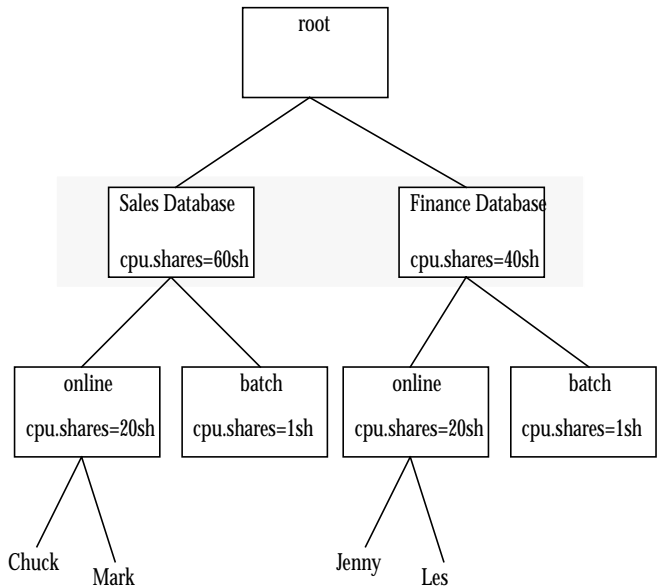


Figure 5. Consolidation of Two Databases with Solaris Resource Manager

### 7.5 Databases

Resource management of databases can be somewhat simplified by viewing the database as a “black box” and resource managing around it. This is a particularly useful strategy when doing server consolidation, since the main objective with consolidation is to partition each workload from each other.

It is rare to see a system with only a database server running on it, and more commonly we see several different packages interacting with the database to provide the overall application environment to the user. Resource management of the whole environment requires careful planning and a solid understanding of the interaction between these packages and the database.

If we look at a typical example where a single database instance is being used to provide database services for two different applications which require access to the same data we start to uncover some of the complexities of database resource management. In this example we show how Oracle is the backend database server, with a web application is used for customer order taking and SAS is used as a real time decision support tool.

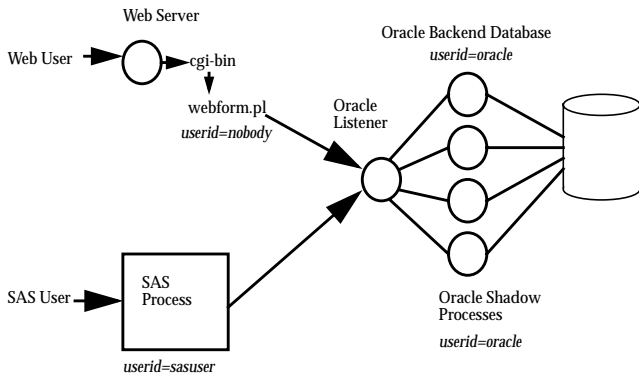


Figure 6. Practical Example with Oracle and SAS

The life cycle of each transaction crosses several management boundaries and no single control or policy can assign resources to ensure that adequate CPU is provided to meet response times. For example, if the SAS user is retrieving a large amount of transactions from the database, the web user may not be able to get adequate response from the database during that time. Ideally what we would like to do is control the allocation of resources as the transaction proceeds through the system. A time line of the transaction through the system is shown in Figure 7.

Each stage of the transaction uses different resources and different techniques may be applied to each stage. Bandwidth manager can be used to control the bandwidth allocation into the web server and the URL of the http content and cgi-bin can be control using bandwidth manager URL prioritizing.

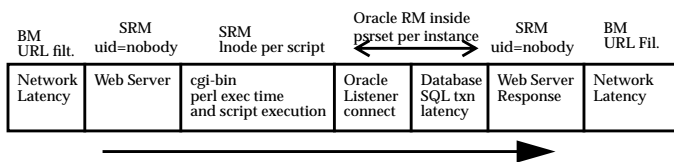


Figure 7. Transaction flow and Resource Management Product assignment

Once the web server has the transaction, SRM can be used to prioritize the CPU allocated to the transaction. SRM can assign resource to the user that the web server is running to ensure that adequate CPU is available. The whole web server can have its own resource allocation. There is rarely sufficient web load to cause contention between transactions on the same web server. The more important part of the web server load is the cgi-bin gateways used to interface with the data-

base. The cgi-bin script can be launched with a per-executable policy using SRM so that each cgi-bin script has its own set of resource guarantees.

The more difficult part using today's technologies is how to manage the priority of the transaction once it enters the database. We could look at approaches where we used fine grained resource management to control how much CPU is allocated to the users in the database, but some vendors will not allow external control of the database by resource management products.

## 7.6 Virtual Memory and Databases

Solaris Resource Manager and resource limits provide the capability to limit the amount of virtual memory used by processes, users and workloads. This capability does not manage physical memory, rather it effectively restricts the amount of global swap space that is consumed by each user.

When a user or workload reaches the virtual memory limit, the system returns a memory allocation error to the application, that is calls to `malloc()` fail. The the same error code is reported to the application as if the application had run out of swap space.

Only a handful of applications respond well to memory allocation errors, because their memory allocation routines have not been well tested under these situations. Because of this, it is a high risk to ever let a database server reach its virtual memory limit. In the event that the limit is reached, the database engine may crash, resulting in a corrupted database.

It is recommended that virtual memory limits be set high, so that these limits could never be reached under normal circumstances. However, the virtual memory limit can be used to place a ceiling over the entire database server to stop a failing database with a memory leak from affecting other databases or workloads on the system.

## 7.7 Database Consolidation

Database consolidation can be achieved by using a number of resource management techniques. The choice of technique should be made based on the supportability and functionality of the solution. Not all databases are supported under each of the resource management facilities and since most consolidated database servers are mission critical production servers then this should be the primary concern.

The objective for successful consolidation is to provide strong insulation between each database instance. To

do this requires careful resource management of CPU, Memory, I/O and Network traffic.

## 7.8 Insulating CPU Usage

Processor usage is a relatively easy resource to isolate, and there are several mechanisms by which this can be achieved. There is a trade-off between the strength of insulation and the flexibility of resource allocation. At one end of the scale is E10000 dynamic system domains which provides as the same level of insulation as separate machines, but requires manual intervention to shift resources between database instances. There is also a large granularity of resource allocation, which is 4 processors at a time.

At the other end of the spectrum is Solaris Resource Manager, which provides fine grained resource management with the ability to dynamically assign resources whenever they are needed, but there is much lower insulation between the applications.

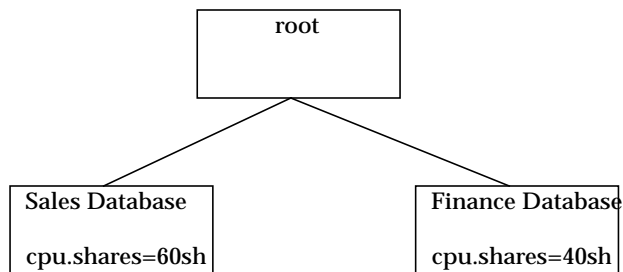


Figure 8. Consolidation of Two Databases with Solaris Resource Manager

The advantage with Solaris Resource manager is that resources are completely dynamic. If one database is not using the resources, then they are available to the other database. This prevents wasting of system resources because spare CPU cycles can be used by any other workload requiring them, and the more workloads consolidated the flatter the total resource usage becomes. By doing this, we can often size a system for the average resource requirements, rather than the peaks.

Processor sets provide a middle ground between the two. The granularity is 1 CPU, and the resources are easier to reallocate than with domains, but still relatively static in nature. Processor sets and domains do however provide the static processor assignment and granularity that is required by some databases, and in some cases this is the only supported way of dividing resources between databases. Unfortunately, resources are statically assigned, and although some movement of resources is possible it means that we need to size each database instance for most of its peak requirements.

## 7.9 Insulating Memory Usage

Databases have three types of memory usage which have very different characteristics:

- the large shared memory area
- the private memory area
- file system paging memory

The shared memory area is often the largest component of the databases memory requirements, and is the easiest to insulate between instances. This is because the memory is allocated as shared memory and all of the mainstream databases allocate this as intimate shared memory (ISM), which wires down the memory so that it can't be paged. Because this memory is wired down, the memory allocated to each instance stays allocated and one instance cannot steal memory from another. This does however provide a challenge for dynamic memory resource allocation, such as dynamic re-configuration since the wired down memory must be unallocated before it can be removed from the system, requiring quiescing of the database.

The private memory area requirements for each database varies, but are generally many times smaller than the global shared memory requirements. The amount of memory used is also configurable and simply capacity planning can be used to ensure that there is sufficient memory system wide to cater for all of the database private memory requirements. Take particular care when including DSS database instances, since these often have very large private memory requirements.

Often overlooked is the memory requirements of the file system. These can be avoided completely by installing each database on raw devices or by using the direct I/O feature of the UFS file system. Both of these options cause the file system to bypass the Solaris page cache. The Solaris page cache causes a very large memory demand which WILL place undue pressure on the database private memory and break down any insulation between the databases which had been established. The memory requirements of the Solaris file system is equal to the size of the data being accessed, i.e. if you are accessing 500GB of data within an instance then you will need 500GB of physical memory installed before the file system will stop putting pressure on other parts of the system. Obviously it is impractical to install this much memory, so we need to look at other alternatives. Priority paging is a new paging algorithm which is available at Solaris 7, and in Solaris 2.5.1 /2.6 with the latest kernel jumbo patches. Installing Priority Paging will put a hard fence between the file systems and applications, and the file system will only use free memory in the system for its cache. If

you are going to run a database on file systems, consider this a mandatory requirement.

In summary, to provide proper memory insulation between databases, it is important to ensure that the following are met:

- use ISM for the shared global areas to wire down memory
- use proper capacity planning and parameters to ensure there is sufficient RAM for private memory requirements
- Either use raw disk devices or ensure Priority Paging is installed to prevent file system memory pressure.

## 7.10 Insulating I/O

Until database resource managers implement I/O resource control the only way to insulate I/O within a database is by ensuring tables are placed on separate I/O devices. This does however still leave shared log devices which can be a source of I/O contention.

Consolidating database however can be achieved by simply ensuring each database is assigned its own I/O devices. By doing this we completely insulate one database from another. Care must be taken to ensure that all data paths from the server to the storage device are carefully isolated. For example, two databases may be placed on their own set of disks within an A3500 storage system, but because both set of disks use the same SCSI channel to the host the databases I/O is not properly insulated.

If a single storage controller must be used, then capacity planning should be used to ensure that sufficient bandwidth is available to combine both. For OLTP applications this is rarely an issue since the bandwidth requirements are so low. For example, an OLTP system generating 2000 8k IOPS only generates 16MB/s of I/O bandwidth. Decision support however is completely different, and a single decision support workload can generate several hundred megabytes a second of I/O. In addition to this, decision support often generates enough I/O to overwrite the cache continuously (cache wiping) which will hurt the OLTP random I/O which makes extensive use of the storage cache. A single storage controller should not be considered for consolidation of decision support workloads with OLTP workloads.

## 7.11 Managing Web Servers

Solaris Resource Manager can be used to manage resources on web servers by controlling the amount of

CPU and virtual memory. There are three basic topologies that are used on systems hosting web servers.

## 7.12 Resource Managing a Consolidated Web Server

A single web server can be managed by controlling the amount of resource 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 is the most basic form of resource management, and simply prevents other workloads from impacting the performance of the web server, and vice versa.

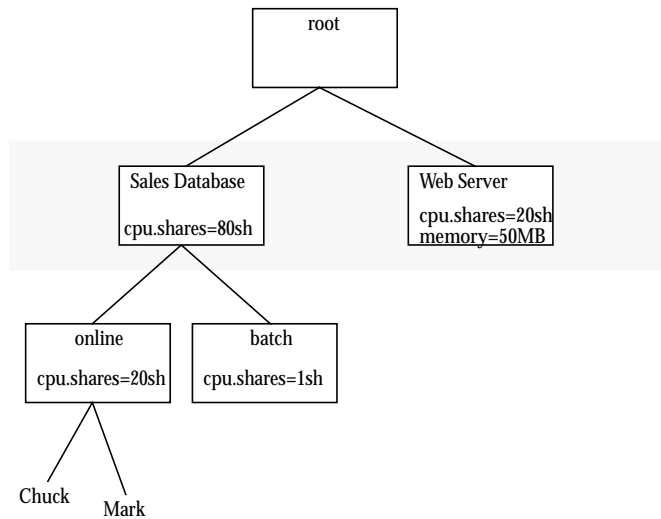


Figure 9. Resource Managing a Consolidated Web Server

In our example, the Web server is allocated 20 shares, which means that it is guaranteed at least 20 percent of the processor resources should 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, rather than the entire system running out of swap space, only the web server will be effected.

## 7.13 Finer 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 may be shared between many users, each with their own CGI-BIN programs.

An error in a single 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 pre-

vent this from happening, the per-process limits can be used.

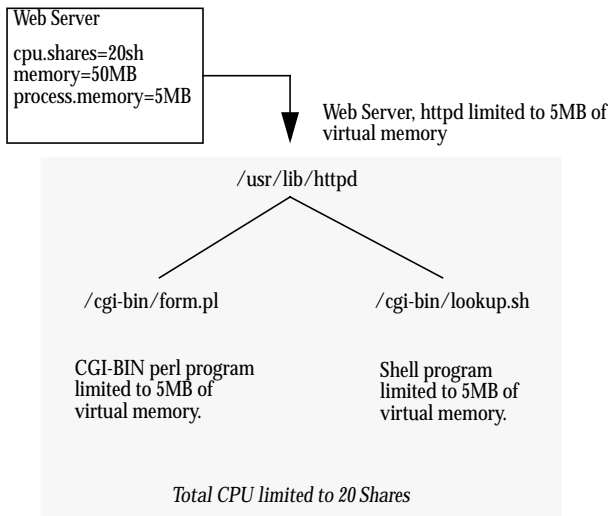


Figure 10. Finer Grained Resource Management of a Single Web Server

## 7.14 Resource Management of Multiple Virtual Web Servers

Single machines are often used to host multiple virtual web servers, in a consolidated fashion on a single machine. In this case, multiple instances of the httpd web server process exist, and far greater opportunity exists to exploit resource control via Solaris Resource Manager.

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 Inode in the Solaris Resource Manager 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 11. Solaris Web Server Parameter File

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

In this case, you can make use of most or all of the SRM resource controls and limits:

**Shares [cpu.shares]** The cpu shares may be used to proportionally allocate resources to the different web servers.

**Mem limit [memory.limit]** The memory limit may be used to limit the amount of virtual memory that the web server can use, which will prevent any one web server causing another to fail from memory allocation.

**Proc mem limit [memory.plimit]** The per-process memory limit can be used to limit the amount of virtual memory a single cgi-bin process can use, which 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.

## 7.15 Creating a Policy on Executable Name

In some cases it may be useful to construct a hierarchy which allocates resources based on the application which each user is executing. This can be achieved

using the `srmuser` command and wrappers around the executables.

A different point in the hierarchy is created for each application type and then authorization is given to those users to switch Inodes as they execute each application.

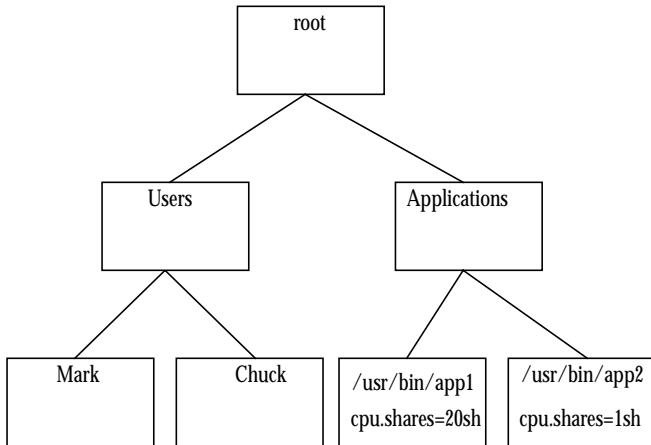


Figure 12. A Hierarchy which allows policies by executable name

This simple hierarchy is established using the `limadm` command to make Chuck and Mark children of the users group, and app1, app2 are made children of the applications group.:

```
# 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
```

Both Chuck and Mark are given permission to switch Inodes on their child processes which will allow us to attach `/usr/bin/app1` to the app1 Inode and `/usr/bin/app2` to the app2 Inode.

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

Shares may then be allocated to both applications in

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

the desired proportions:

A simple wrapper script may now be created which sets the appropriate Inodes upon execution of the applications using the `srmuser` command:

```
#!/bin/sh
#
# Execute /usr/bin/app1 and attach it to
# the app1 Inode
#

/usr/srm/bin/srmuser app1 /usr/bin/app1
```

## 7.16 System Resource Accounting Information

Solaris Resource Manager provides the ability to generate accounting data based on resource usage. This is made available by the accumulation of resource information in the Inode tree. Although Solaris resource manager does not provide resource accounting, it does provide the data and data extraction tools which can be used to develop a system to generate accounting (or billing) information.

## 7.17 Billing Issues

The administrator must decide which Inodes are to be billed for resource usage. For example, administrators may only be concerned with billing entire departments, so they may only wish to bill the group headers of the topmost groups, whose accrued usage will include all the accrued usage of the Inodes at lower levels within their departments.

For administrators to be able to implement a billing system, they must determine a costing function for each resource to be billed. This may be a simple linear relationship (where unit cost is the same, regardless of the amount used), or it may be a non-linear relationship, such as a step function, or a curve where unit cost varies as the amount of usage varies.

In deciding upon a costing function for each resource, the administrator should keep in mind that the costing function will not only control the assignment of cost to accrued resource usage, but can also have an impact

on the way in which a user uses the resource. For example, if the costing function for virtual memory usage causes the unit cost to increase as the amount of usage increases, there is a strong incentive for the users to keep virtual memory usage low. Therefore, it is possible for the administrator to control user behavior through the use of an appropriate costing strategy.

There is only one accrue attribute per resource. It contains the accrued usage for the resource based on the usage attribute for the resource. This means that there is no accrued usage corresponding to the myusage attribute. For group headers, there is no accrued usage for the user as an individual, since the accrue attribute holds the group's accrued usage. For Inodes with no children, leaf Inodes, this does not matter, since the myusage attribute and the usage attribute are identical. If a bill is required for the individual accrued usage for a group header, it must be calculated from the group total less the sum of the individual totals of each child in the group.

### 7.18 Extracting Accounting Data

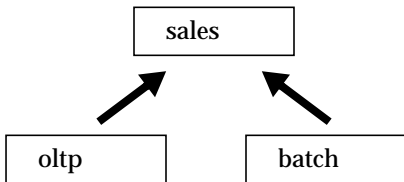
The limreport command allows an administrator to query any attributes, including the accrue attributes, of any user. The command provides a flexible method for the selection of information to be displayed from the chosen Inodes.

The limadm command provides a mechanism to reset the usage data of each user at an appropriate point, the timing of which will depend on the billing strategy.

1. Zero Usage Counters  

```
# limadm set cpu.accrue=0 sales
```

2. Usage is accrued up the tree



3. Usage may be reported with liminfo or limreport  

```
# liminfo sales
# limreport 'uid==5000' '%f' cpu.accrue
```

Figure 13. Life cycle of usage data

For example, if bills are to be produced at a group level, and then individual bills are to be produced for the group members, the accrue attributes of the group members would not be cleared until after both bills have been produced. However, if individual bills are not to be produced, the group members' accrue attributes may be cleared at the same time as the group header's, even though they may not have been individually used.

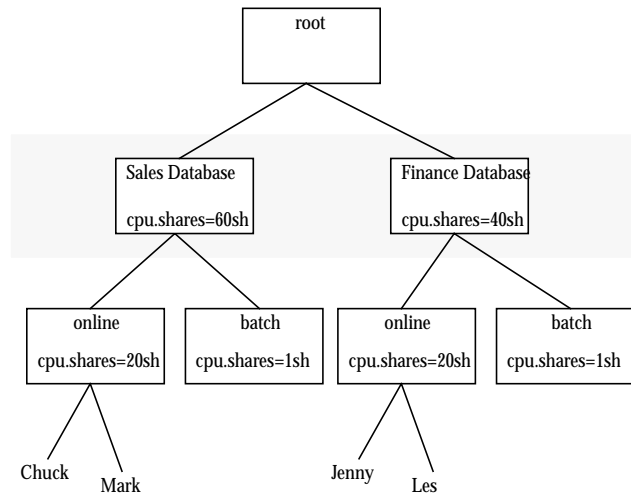


Figure 14. A sample hierarchy

If we take a look at one of our previous examples that uses a multi level hierarchy we can look at the points of the hierarchy from which we would want to take usage information.

### 7.19 The limreport and liminfo commands

A common requirement may be to generate billing information for all users in the "online" group. We can do this with the limreport command:

```
# limreport 'flag.real==set &&
sgroup=online' '%f' cpu.accrue
```

At a global level we may want to know the usage information for the two departments, sales and finance. This can be obtained by looking at the Inodes for finance and sales since the usage information is accrued up the hierarchy.

The liminfo command can be used to display the resource usage information for both Inodes and the limreport command can be used to print a list of Inodes matching a certain criteria.

```
# liminfo -c sales
Login name:      sales      Uid (Real,Eff):      5000 (-,-)
Sgroup (uid):   root (0)   Gid (Real,Eff):      200 (-,-)

Shares:         60         Myshares:             0
Share:         41 %       E-share:              0 %
Usage:         880425      Accrued usage:        3.28954e+09

Mem usage:      890.0859 / 484.2266 MB   Term usage:          2d9h8m41s
Mem limit:      0 B                    Term accrue:          2d9h8m41s
Proc mem limit: 0 B                    Term limit:           0s
Mem accrue:     3.247672902309 PB.s

Processes:      110 / 56   Current logins:      0
Process limit:  0

Last used: Tue Oct 4 15:04:20 1998
Directory: /
Name: Sales node in hierarchy
Shell: /bin/false

Flags:
```

Figure 15. Using liminfo to look at accrued CPU usage for all members under sales

Limreport can also be used to view the group level information with selective formatting.

```
# limreport 'flag.real==set && uid==5000' -
lname cpu.accrue

# limreport 'flag.real==set && uid==5000'
'%-20s %20.0f' lname cpu.accrue
```

The lim report command is a flexible report generator that can be used to produce reports using the fields from the Inodes. It does this by doing a linear scan through the passwd database and looking up the Inode for each entry. Notice that in our example the match criteria includes the 'flag.real==set', this is to prevent limreport from trying to get data from Inodes which don't yet exist, likely if the user has never logged in. The complete list of fields available to limreport are shown in TABLE 2..

TABLE 2. Identifiers to the limreport command

Identifier	Description
lname	string; login name
uid	integer; user UID
gid	integer; initial group GID
pword	string; encrypted password
dirpath	string; initial (home) directory
shellpath	string; initial shell
comment	string; the comment, or gecost field
gecos	string; a synonym for comment
sgroupname	string; the login name of the user's scheduling group parent
now	integer; the current time
level	integer; the depth of the Inode in the scheduling tree; root is 0. For orphan Inodes, this is the depth within the disjoint tree
orphan	integer; is non-zero if the Inode is an orphan

TABLE 2. Identifiers to the limreport command

Identifier	Description
preserve	string; a list of attribute value assignments using the syntax of limadm (1MSRM). Read-only attributes are omitted from the list. The command: limreport 'flag.real' -lname preserve will generate output which, if passed to limadm using the -f option, will completely reconstruct the state of all users' Inodes at the time of execution of the limreport command
myuid	integer; the UID of the Inode to which limreport is attached
mylname	string; the login name corresponding to myuid
clear	flag; a constant
set	flag; a constant
group	flag; a constant
inherit	flag; a constant
shellpath	string; initial shell

The full list of identifiers can be obtained with the limreport -? command.

The operators to the limreport command are summarized in TABLE 3.

TABLE 3. Operators to the limreport command

Op	Description	Op	Description
*/	Multiply/divide	%	Integer modulo
:+ -	Add/subtract	&   ^	Bitwise and/or/exclusive-or
~	Bitwise complement	:	Is a member of scheduling group
== !=	Equal/not equal to	> <	Greater/less than
>= <=	Greater/less than or equal to	&&	Logical and/or
{...}	A date and time		
!	Unary logical not	(expr)	A sub-expression

On systems with large password databases using name servers, the limreport command can take a very long time to execute. This can be resolved by setting the /etc/nsswitch.conf to files if absolutely necessary

## 8.0 CONCLUSION

The Solaris Resource Manager offers a comprehensive set of resource management features that allows us to:

- Control CPU Allocation
- Prevent Denial of service attacks
- Report of resource usage

Many scenarios may require more complex resource management which can be implemented by using a combination of resource management products from Sun.

## **9.0 FUTURE WORK**

As noted earlier, a complete study on how resource management should be implemented in the enterprise is being performed under the Sun BluePrints program. This information will be made available in July as a Resource Management guide and via the BluePrints web site at <http://www.sun.com/blueprints>.

## **10.0 REFERENCES**

Cockcroft, A., *Sun Performance and Tuning - Java and the Internet*, 2nd Edition, Sun Microsystems Press/Prentice Hall, 1998

Wong, B., *Capacity Planning on Sun Systems*, Sun Microsystems Press/Prentice Hall, 1996

*BluePrints Online<sup>TM</sup> Articles*, <http://www.sun.com/blueprints>

*The Resource Management Blueprint<sup>TM</sup>*, due July 1999.