CHAPTER 2

# Service Level Management

This chapter describes the overall methodology of service level management so that the resource management component can be put into a wider context. It also describes and compares several approaches to resource management.

## Service Level Definitions and Interactions

This section starts with a high level view of service level management and defines a terminology that is based on existing practices.

Computer systems are used to provide a service to end users. System and application vendors provide a range of components that can be used to construct a service. System managers are responsible for the quality of this service. A service must be available when it is needed and must have acceptable performance characteristics.

Service level management is the process by which information technology (IT) infrastructure is planned, designed, and implemented to provide the levels of functionality, performance, and availability required to meet business or organizational demands.

Service level management involves interactions between end users, system managers, vendors and computer systems. A common way to capture some of these interactions is with a service level agreement (SLA) between the system managers and the end users. Often, many additional interactions and assumptions are not captured formally.

Service level management interactions are shown in FIGURE 2-1. Each interaction consists of a service definition combined with a workload definition. There are many kinds of service definitions 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 the work that 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 to key applications and transactions. Availability is specified as uptime over a period of time and is often expressed as a percentage (for example 99.95 percent per month). Performance may be specified as response time for interactive transactions or throughput for batch transactions.

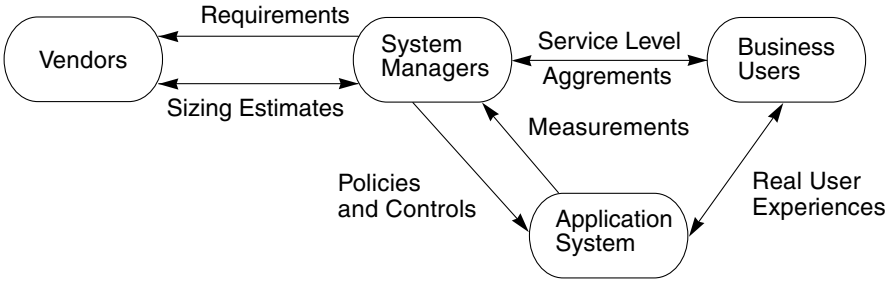


**FIGURE 2-1** Service Level Management Interactions

# System Level Requirements

System managers first establish a workload definition and the service level requirements. The requirements are communicated to vendors, who respond by proposing a system that meets these requirements.

# Sizing Estimates

Vendors measure the system performance 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 can be a 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 and performance guarantees for production systems with a defined workload (at a price). Vendors cannot provide unqualified guarantees because typically, many application and environmental dependencies are outside their control.

## Service Level Agreements

System managers and end users negotiate an SLA 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 online 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-upon service levels could be too demanding or too lax. The system may be quite usable and working well, but still failing an overly demanding service level agreement. It could also be too slow when performing an operation that is not specified in the SLA or whose agreed-to service level is too lax. The involved parties must agree to a continuous process of updating and refining the SLA.

## Real User Experiences

The actual service levels experienced by users with a real workload are subjective measures that are very hard to capture. Often problems occur that affect parts of the system not covered explicitly by the service level agreement, or the workload varies 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.

## Service Level Measurements

The real service levels cannot always be captured directly, but the measurements taken are believed to be representative of the real user experience. These measurements are then compared against the service level agreement to determine whether a problem exists. For example, suppose 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 backend server system than at the user interface on the client system, but it is important to be aware of the limitations of such measurements. A transaction may take place over a wide variety of systems. An order for goods will affect systems inside and outside the company and across application boundaries. This problem must be carefully considered when the service level agreement is made and when the service level measurements are being defined.

# 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 in those policies. A policy 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 oto 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 overachieving.

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

## 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-upon workload definition or where the measured service level falls short of the defined level, an exception report is produced.

## 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.

# Resource Management Control Loop

To manage a resource, you must be able to measure and control it. A control loop is set up that measures the performance of an application subsystem, applies policies and goals to decide what to do, then uses controls to change the resources being provided to that subsystem. This loop can be implemented manually on a time scale measured in days or weeks (by reconfiguring and upgrading entire systems), or it can be automated in software and run as often as every few seconds. The control loop is shown in FIGURE 2-2.

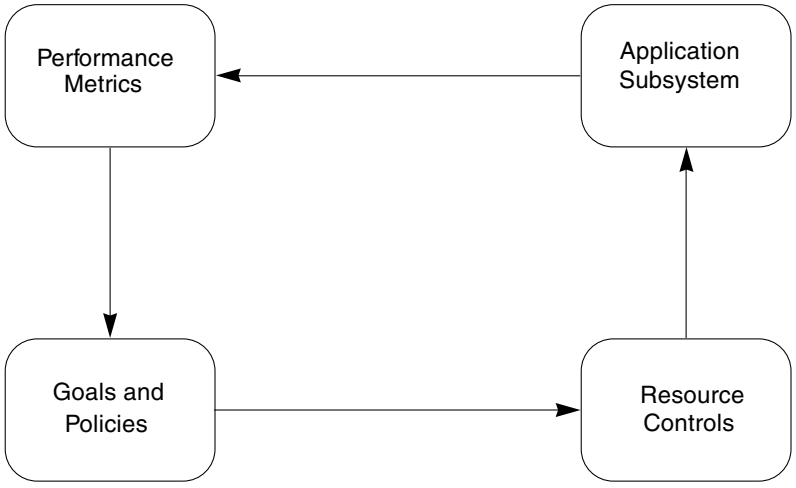


**FIGURE 2-2** Resource Management Control Loop

A complete system implements many control loops. A brief digression into basic control theory is provided at this point to help explain the behavior of such systems.

# A Simple Approach to Control Theory

We spend so much of our lives operating control loops that it is actually quite intuitive to most people. Designing a control loop is more complex and requires a more explicit understanding of the situation.

You start with an objective, such as to steer a car around a corner while staying in the lane. You apply a control input by turning the steering wheel to the point that will get you around the corner. After a delay. the car responds, You measure the response, compare it with what you wanted, and obtain the error difference. If the difference is zero you don't need to change the control input. If the turn is too tight, you need to reduce the input. If the turn is too wide, you need to increase the input. You have to decide how much extra correction is needed to compensate for being wrong the first time, and also decide whether the car has finished responding fully to the initial input. You may decide to over- or under-correct, and apply the correction gradually or quickly (for example, if you are heading straight for a lamp post!).

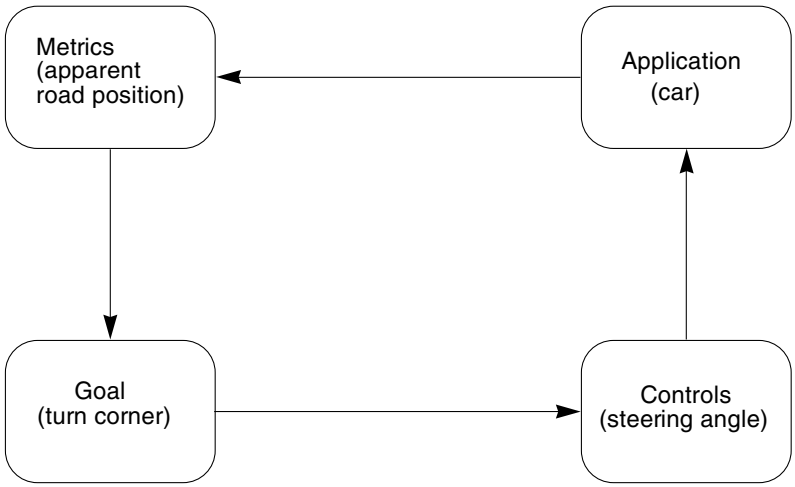


**FIGURE 2-3** Example Control Loop

The first time you tried a car driving game on a computer, you probably swung wildly from side to side. This wild swinging is caused by over-correcting too late because you don't have the same motion sensing inputs you have in a real car. When the car is oscillating, your corrections may end up being delayed to the point that you are turning the wheel the wrong way at the wrong time, and you might spin off or crash. Eventually, you learn to react based on what you see on the screen and make smaller corrections more quickly to keep the car on track and stable.

In control terms, you are applying *negative feedback* to the system. You take the error difference between what you wanted and what you got, and apply the inverse of the error to the system to reduce the error in the future. The rate at which you measure and apply corrections is called the *control interval*, and the rate at which the system responds to changes is called the *time constant* for the loop. The amount of the error that you feed back changes the characteristic behavior of the control loop. If you feed back a large proportion of the error with a short control interval, the system is *lightly damped* and will be very responsive to sudden changes but will probably oscillate back and forth. If you feed back a small proportion of the error over a longer control interval, the system is *heavily damped* and will tend to be sluggish and unresponsive with a large time constant.

When you apply these principles to computer system resource management you can see that it is important to average measurements over an appropriate time scale and get the damping factor right. The resource manager needs to respond quickly enough to cope with sudden changes in the workload such as many simultaneous user logins at the start of a shift, while maintaining a steady flow of resources to all the workloads on the system so that response times are consistent and predictable.

# Viewpoints of Resource Management

You can measure and control a computer system in many ways. This section examines various approaches to solving resource management problems. The methodology used depends upon the starting point of the developers, for example a network-centric methodology can be extended for use in other areas, so can a storage-centric or server-centric viewpoint.

## Diverse Methods

This diversity of approach occurs both because of the products that are available and because of the expectations of the end users who are purchasing solutions. In a large organization, several groups (such as system administrators, network managers, database administrators, and security managers) are responsible for different parts of operations management.

In some organizations, each group is free to use its own methods and obtain its own tools. This can cause demarcation problems because there is so much overlap in the scope of each method. Or a single methodology and tool could be imposed by the dominant group. The problem with such an aproach is that the methodology may be optimal for managing one aspect of the system only and could do a poor job in other areas.

In an ideal world, one all-encompassing mega-tool would implement an integrated methodology and solve all resource management problems. The closer you get to this ideal, the more expensive and complex the tool becomes. And you may not want all of its features. So it is harder to justify purchasing it.

A more pragmatic approach is to integrate simpler, more specialized tools that share information with each other, have a similar look and feel, and can be used as a standalone product as needed.

Today's diverse set of methodologies and tools have very little integration between them and several different user interfaces. The products produced at Sun Microsystems, Inc. are converging on a common user interface style and on common technologies for sharing information to provide better integrated resource management components. This book's primary role is to explain to data center operations managers how to use combinations of the current set of products and technologies. It also indicates the direction of the development and integration work that will produce the next generation of products.

## The System-Centric Viewpoint

The system-centric viewpoint focuses on what can be done on a single server using a mixture of hardware and operating system features. The operating system provides basic management capabilities for many components such as attached network and storage devices. But it specializes in managing CPU and memory resources for a single desktop or server system. Dynamic Reconfiguration (DR), processor sets, Solaris Resource Manager™, and Sun Enterprise™ 10000 (also known as Starfire™) Dynamic System Domains (DSDs) are all system-centric resource management technologies. The Sun hardware system-centric resource management tool is the Sun Enterprise SyMON™ 2.0 software. It is based on the Simple Network Management Protocol (SNMP), so it also has some network management capabilities. The Solaris Management Console™ software provides a more generic framework for gathering together operating system administration tools and interfacing to industry standard initiatives such as the web-based management initiative (WebM) and the Common Information Model (CIM). The Starfire system currently uses its own HostView interface running on a separate system service processor (SSP) to manage domains.

The system-centric viewpoint runs into problems when a lot of systems must be managed. Coordinating changes and allocating resources becomes complex quite rapidly. Tools like the Sun Enterprise SyMON 2.0 software can view many systems on one console, but cannot replicate and coordinate changes over multiple systems. One reason why the Sun Enterprise SyMON 2.0 software does not fully support management of the Starfire system is because each domain on the system runs a separate copy of the Solaris operating environment and sees a different subset of the

hardware configuration. The next release of this software will be extended to view an SSP and all the domains in a Starfire system as a special kind of cluster so it can be used in place of the HostView interface.

The operating system and devices provide a large number of measurements of utilization, throughput, and component-level response times. There are several good ways to control CPU resources, but at present, there is no way to control the usage of real memory by a workload. The only way to constrain a workload that is using too much memory is to slow down or stop its CPU usage so that it stops referencing its memory, which will then be stolen by other, more active processes.

Manual resource management policies can be implemented using the Sun Enterprise SyMON Health Monitor, which generates alerts when a component of the system becomes overloaded. The system administrator can then tune or reconfigure the system to avoid the problem. Automatic resource management policies are implemented by Solaris Resource Manager, which dynamically adjusts the priorities of processes to ensure that their recent CPU usage tracks the share of the system that has been given as a goal for that user.

## The Cluster-Centric Viewpoint

The cluster-centric viewpoint concentrates on coordinating resource management across the cluster. Systems are clustered together to provide higher availability and higher performance than that provided by a single system. A cluster is more complex to install and administer, so tools and methods attempt to automate cluster management to make it more like a single system. From a resource-management viewpoint, the primary issue is load balancing and the extra costs of accessing nonlocal information over the cluster interconnect. Sun has two kinds of clusters. The highly integrated SPARCcluster™ product range is focused on improved availability in commercial environments. Its management tools will eventually become an integrated extension to the SyMON software. For high performance computing, Sun HPC Servers use the platform computing load share facility (LSF) to perform load balancing on much larger and more loosely coupled clusters.

At a cluster level, multiple system level measurements are compared to measure load balance and look for spare resources. The cluster interconnect utilization and proportion of remote data access are important additional measures. The primary resource management control is the choice of where to run new work. It is not currently possible to migrate a running job from one node in a cluster to another, or to checkpoint a job to disk and restart it again later, either on the same node or on a different one.

When deciding on the resources that are available on a node, it is easy to decide if there is some spare CPU power, but very hard to decide if there is enough available real memory. The kernel maintains a free list of memory, but there is also some proportion of memory in use as a file system cache that could be reclaimed to run a new job if there was a way to measure it. This is a current issue for the LSF product.

## The Network-Centric Viewpoint

From a network point of view, there are a large number of devices to manage. Many of them are network components with limited monitoring capabilities, such as bridges and routers. The primary resource that is managed is network capacity. At the intersection of servers and networks, there are products that perform protocol based bandwidth management on a per-server basis (such as Solaris™ Bandwidth Manager software) or act as secure firewalls. In the telecommunications industry, network management encompasses all the equipment required to run a global system where the end-points are mostly telephones or mobile cellphones, and the traffic is a mixture of speech and data. In this environment, SNMP is too simple, so the more scalable OSI-based CMIP management protocol is often used. The Solstice™ Enterprise Manager product is a Telco-oriented CMIP and SNMP management system that is used to manage cellular networks. In theory it could be used to manage computer systems and local area networks, but it was not developed to do this. Computer-oriented local and wide area networks are normally managed using SNMP protocols, with the Solstice SunNet Manager™ or HP OpenView products collecting and displaying the data. Both products provide some visibility into what is happening in the computer systems on the network, but they are much more focused on network topology. At this level, resource management is done on a per-network basis, often by controlling the priority of data flows through intelligent routers and switches. The Sun Enterprise SyMON 2.0 software can act as a network management platform as well as a system hardware management platform, which may help to integrate these two viewpoints.

Protocol information along with the information found in packet headers and network addresses form the basis for network measurements. There is no user or process identifier in a packet, so it is hard to directly map network activity to system level activity unless an identifiable server process is dedicated to each protocol. Some protocols measure round trip times for their acknowledgments. This can provide an estimate of network latency between two systems.

Network controls are based on delaying and prioritizing packets based on the protocol and destination data in the packet headers. This can occur at the servers that make up the end points or in the routers that connect them.

## Storage-Centric Viewpoint

Storage has recently moved from being a simple attached computer system peripheral to a complex managed entity in its own right. Networked storage using fibre channel puts an interconnection layer in between multiple servers or clusters and multiple storage subsystems. This storage area network (SAN) can contain switches and routers just like local or wide area networks, but the protocol in common use is SCSI over fibre channel rather than IP over Ethernet. A SAN may also span multiple sites, for example, where remote mirroring is being used for disaster recovery. Storage is also now open for access in a heterogeneous multi-vendor environment, where multiple server and storage vendors can all be connected over the SAN. This is an emerging technology, and tools to manage a SAN are still being developed. Sun provides one approach with an industry-wide initiative called Jiro. It is based on a distributed pure Java™ technology platform that can run anywhere a JVM is available, scaling from embedded devices through open systems into mainframe and enterprise environments. Jiro enables management of any storage resource in a heterogeneous distributed environment, from storage hardware, like devices and switches, to storage software like backup solutions and volume managers. Jiro is being promoted as an open multi-vendor standard.

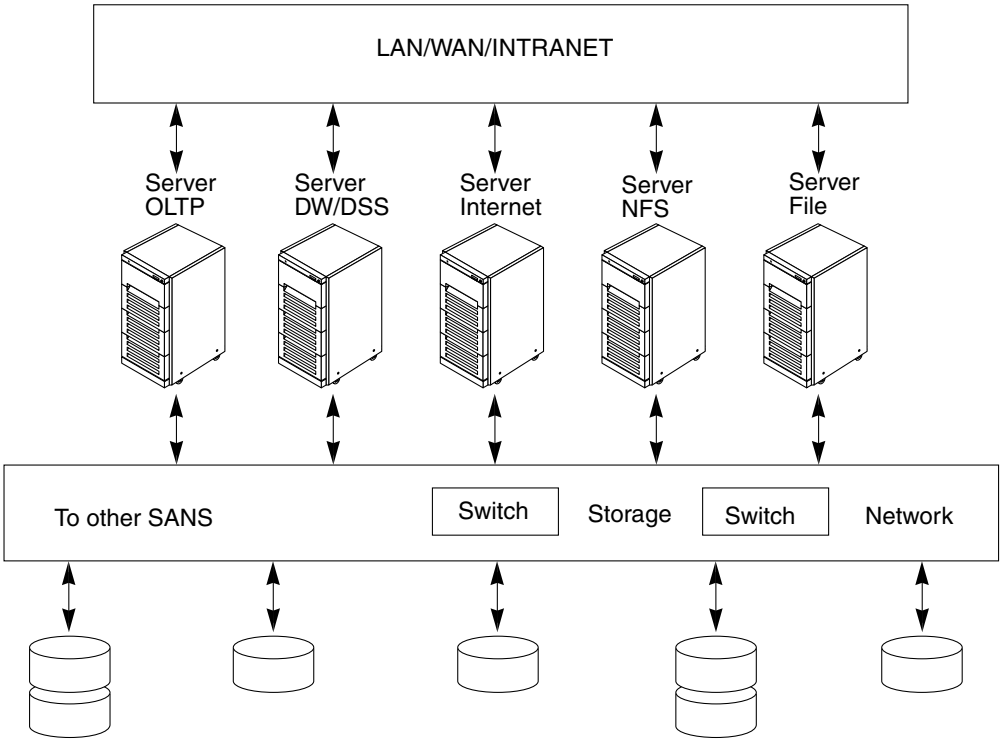


**FIGURE 2-4** Storage Area Network

Storage management has two constraints that make it interesting: one is that it must be distributed over many server systems and storage devices to be useful. The other is that these servers and devices come from many vendors and run different operating software. So Jiro cannot include a core dependency on the Solaris operating environment or other Sun products. Jiro must solve some of the generic problems of clustered and networked resource management. In particular, it manages the state of distributed devices in a persistent manner. Jiro must be capable of stand-alone operation, with component management interfaces to other products.

Jiro enables resource management of capacity, performance, and availability of data storage. Backup and archival policies can be used to automate migration of data to a tape library. Measurements of capacity and performance characteristics can be combined with availability policies, so the operator will be alerted of any problems. Ultimately storage subsystems will be reconfigured automatically.

At present, Jiro does not include a general purpose rule script-based policy engine. It does provide the infrastructure necessary to create polices, allowing the view of storage to be elevated to the Storage Service level.

Integration between the system viewpoint and the storage viewpoint has some of the same problems as the integration of system and network viewpoints. The traffic on the SAN does not contain any indication of which user or process generated the request. Within the Solaris software, it is possible to trace storage accesses on a per-process, per device basis. But the overhead of collecting and analyzing this data is quite high. There may be a need for a Jiro Solaris Storage Bandwidth Manager to bridge the two management viewpoints in a way that accounts for, prioritizes, and controls SAN bandwidth on a per process or per user basis.

## Database-Centric Viewpoint

A database management system has its own notion of users, manages its own memory and storage allocation, and can appear as a "black box" to the server system on which it runs. Some database vendors have implemented their own resource management capability on a per user or per transaction basis. This may take the form of priorities, shares, or limits on CPU usage per user or per transaction. The database also contains its own logic that implements policies and dynamically controls resources.

The database implementation can sometimes work well with server-based resource management. This is described in more detail in Chapter 4.

Integration is needed between the internal database resource management capabilities and the other resource management viewpoints.

## Application-Centric Viewpoint

Large and complex applications such as SAP R/3, Baan, and Oracle Financials contain their own resource management concepts and controls. For example, Oracle Financials implements its own batch queue system to decouple the generation of large reports from the interactive response time of the users. A report is sent to a subsystem called the concurrent manager, which is configured to have a number of parallel streams of work of various types according to the local policy. Concurrent manager activity can be scheduled to occur outside normal working hours, or it can be used to soak up spare cycles during the day.

SAP R/3 measures the response time of important transactions and breaks these down into application server time and backend database server time. As many users connect to an application server and many database servers connect to a single backend database, there is no concept of which user is doing work on the backend system. The application itself must have the instrumentation to keep track of what is going on. The application directly implements the policies and controls.

# Integrated Methods

In a large organization, all of the above viewpoints are useful, and some combination of methodologies is probably implemented already. FIGURE 2-5 indicates the relative breadth of coverage of each methodology. A darker box shows that better coverage is available, a white box indicates that little or no support is provided for a combination.

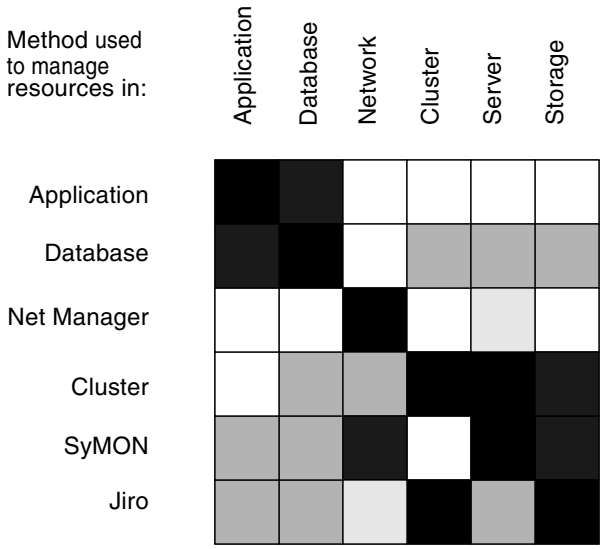


**FIGURE 2-5** Integrated Methodology by Viewpoint

The mature methods have remained specialized, but the emerging technologies of Sun Enterprise SyMON 2.0 and Jiro address a much broader scope of problems. The product overview section of this book discusses their capabilities in much greater detail.

So far the discussion has been quite abstract. The next chapter introduces several example workloads, showing the appropriate methodologies to manage resources for them.

# The Consolidation Process

The consolidation process starts when you identify candidate systems and applications. First measure the resource usage and service levels of those systems so you can see which application workloads will fit together best.

Next, migrate those systems to a common Solaris release and patch revision and do some testing so you can be sure that everything works correctly in the same environment. You also need to make sure that there are no conflicts in the name service configuration and network services files. For example, local password files may need to be merged, and any conflicting port numbers specified in the `/etc/services` file may need to be cleared. If you use a name service such as NIS for all your password and services information, then the systems should already be seeing the same name space and definitions. Using a common name service eases the consolidation process. If you prefer to make all the changes at one time, then you can upgrade as the application is consolidated, but allow for more testing time and a more incremental installation process on the consolidated system

For each group of consolidated applications, you must choose appropriate resource management controls. Once you have consolidated your applications to fewer systems, monitor and re-size the consolidated systems to allow for peak loads. You can either remove excess resources for use elsewhere or identify additional candidate applications to be consolidated onto these systems. Treat this as a rolling upgrade program rather than a one-time big change.

An obvious question that arises is how many systems should the new consolidation contain. Circumstances vary, but the basic principles remain the same. If you treat consolidation as a process, then the number of systems decreases over time and the size of systems increases.

Downtime impacts multiple applications on the consolidated systems. Therefore, when you increase the resources by adding an extra application, you want to do so without rebooting. Consolidated upgrades benefit from systems that can perform dynamic reconfiguration. The midrange Sun Ultra™ Enterprise™ E3000-E6500 servers can perform I/O board reconfiguration with the Solaris 2.6 release, but they require the Solaris 7 release for dynamic reconfiguration of CPU and memory, which causes some application availability issues. The number of system footprints may be too high with midrange servers, and it is hard to reduce the total number of servers effectively. With the high-end Starfire system, Dynamic System Domains (DSDs) solve these problems. DSDs are supported on the Solaris 2.5.1, 2.6, and 7 releases. The total number of DSDs can be reduced as applications are consolidated.

One approach is to use each DSD for a different Solaris revision. You may have a large DSD for the bulk of your Solaris 2.6 applications, a smaller one for applications that have not yet migrated from the Solaris 2.5.1 release, and a development and test

DSD for the Solaris 7 release. Over time, the Solaris 2.5.1 DSD will shrink away and its resources will migrate into the other DSDs. Applications will also migrate into the Solaris 7 DSD. The key benefit here is that this all happens under software control, using a single system footprint in the data center. DSDs are described in detail Chapter 8.

Use the Solaris Resource Manager or the Solaris Bandwidth Manager software or processor sets to control applications within a single copy of the Solaris operating environment.

A consolidated system runs a mixture of workloads. You have to choose relevant processes and aggregate to measure them. The remainder is overhead or unplanned activity. If it is significant, it should be investigated. Break down network workloads as well so that you know which applications are generating the network traffic.

There is a common set of measurements to collect per workload.

- Number of processes and number of users
- End user response times for a selection of operations
- User and System CPU usage
- Real and virtual memory usage and paging rates
- I/O rates to disk and network devices
- Microstate wait timers to see which resources are bottlenecks

To actually perform workload aggregation you have to match patterns in the data.

- match processes on user name
- match processes on command name and arguments
- match processes using processor set binding
- match system accounting data using user and command name
- match network packets on port number and protocol

You usually have to assign disks and file systems to workloads manually. Dealing with shared memory, libraries, and code makes RAM breakdown hard.

When you are accumulating measurements don't accumulate the `ps` command `CPU%`. It's a decayed average of recent CPU usage, not an accurate measure of actual CPU usage over an interval. You need to measure the actual process CPU time used in each interval by taking the difference of two measurements. There is more detail on the available measurements and what they mean in Chapter 5.