CHAPTER 3

Realizing the Vision

Chapter 2 discussed why there is a need to view the infrastructure on which services are deployed differently. This view leads to the definition of the N1 Grid vision, strategy, and architecture. This chapter discusses the desired properties, attributes, and potential routes to realizing N1 Grid systems. Although much of the text describes N1 Grid systems and the N1 Grid operating environment (N1 Grid OE), the definitions are neither implementations nor product descriptions, nor are they product names.

Properties of an N1 Grid System

One of the best ways of exploring the true nature of an N1 Grid system is to compare it with a more familiar class of systems (for example, a traditional symmetric multiprocessor or SMP computer system). A traditional SMP system is typically defined as the set of components that share a system bus, memory, and I/O, and that run under the control of an instance of an operating system (for example, the Solaris™ or Linux operating system).

An N1 Grid system is defined as a set of components (for example, servers, network switches, load balancers, firewalls, storage switches, storage controllers, and disks) under the control of a single instance of the N1 Grid OE. However, unlike a traditional operating system, the N1 Grid OE is distributed (that is, components of the operating environment run on various components within the N1 Grid system). An N1 Grid system is constrained to being within a single data center, although clusters of N1 Grid systems, both within a single data center or spread across a number of data centers, might act like and exhibit many, if not all, of the same attributes as a single N1 Grid system. Do not be distracted by the potential for geographically separated or clustered N1 Grid systems for now. Just like a traditional SMP system, an N1 Grid system is under the control of a single instance of an operating system.

An N1 Grid system consists of a set of platform resources, both hardware and software, within a single data center that is managed by a single instance of the N1 Grid OE. The N1 Grid OE itself is a part of the N1 Grid system, just as the Solaris™ Operating System (Solaris OS) is a part of a traditional server system. Platform resources include but are not limited to servers, service processors, network switches, load balancers, firewalls, storage switches, storage controllers, disks, hypervisors, and operating systems.

N1 Grid Operating Environment

In general, the N1 Grid OE is the software component of an N1 Grid system that manages the mapping of workload (that is, services) onto platform resources in line with a set of *policies*. *Workloads* include, but are not limited to:

- Multitier web services
- Traditional client-server applications
- Grid workloads

Policies reflect high-level business goals, related to cost and quality of service, such as:

- Priority (versus other services)
- Average transaction response time (for transactional workloads)
- Number of concurrent transactions or users
- Target completion time (for batch or non-interactive and non-transactional workloads)
- Acceptable outage (in hours per year or hours per month)
- Cost of service

Resources are system elements, such as:

- Compute elements (for example discrete servers—with or without installed operating systems or hypervisors—and physically partitionable servers, such as Sun servers with dynamic system domains)
- Network elements (such as switches, routers, load balancers, and firewalls)
- Storage elements (such as disks, arrays, array controllers, and SAN fabric elements)

The N1 Grid OE is unlike a traditional operating system in that it is inherently both distributed and hierarchical. Its components, as well as the resources it manages, are distributed across a network. The resources are not simply physical components. They can be aggregations of components, including software. Thus, a server with a traditional operating system (for example, the Solaris OS) or a cluster (for example, a Sun™ Cluster software environment) can be a resource, to which policy and the

management of specific workloads might be delegated. The N1 Grid OE can deploy the traditional operating system or clustering software to create what is sometimes referred to as a metaresource, which it then manages.

The full realization of an N1 Grid OE (for instance, a single, integrated operating environment) is in the future. However, some key components of the operating environment are available as products today, and N1 Grid systems can be implemented in a basic form using today's technologies, coupled with appropriate architectures and operational models.

Heterogeneity Within N1 Grid Systems

Heterogeneity is an obvious requirement for N1 Grid systems because data centers rarely have a single vendor strategy for their infrastructure. N1 Grid solutions are also service-centric, and typically, not all of the components or tiers of a service are deployed on a homogeneous environment. To manage services holistically, a distributed, heterogeneous set of resources has to be managed. Thus, the N1 Grid OE builds on existing, open network computing protocols and standards and on extant application models.

An analogy with a more traditional computer system should help clarify. Typically, a systems vendor architects a computer system. When that system is manufactured, some of the components of the system are designed and manufactured by the systems vendor. These are typically components that reflect the core competency, differentiation, and value-add of that vendor. Some other components might be designed by the vendor, but manufactured by another party to the vendor's designs. Again, this might be required to provide optimized components that ensure a better integrated and more valuable system. However, implementation of that componentry might not be a core competency of the vendor. Finally, some components are commodity parts. The only differentiation between one vendor's product and another is price or performance. Functionally, they are identical.

Just as this is true when building a traditional computer, it is also true in the case of an N1 Grid system. An N1 Grid system and its operating environment can include both software and hardware components from multiple vendors, which are driven by the same architecture with successful implementation being based on common functional and interface specifications. N1 Grid systems can include third-party network and SAN fabric elements and compute elements, together with their hosted operating systems, such as an IBM UNIX server, a Dell Linux server, or a Hewlett Packard server running the Microsoft Windows software.

Functional Breakdown

An N1 system consists of a collection of network distributed resources:

- Compute elements (such as traditional servers, blades, or hardware partitionable servers, such as the Sun Fire™ 15K server).
- Network elements (such as switches, routers, firewalls, and load balancers)
- Storage elements (such as disks, arrays, array controllers, and SAN switches)

These managed elements become components within the N1 Grid system, just like the individual processor in today's SMP, so that they are no longer managed directly. The traditional operating system manages processors on behalf of the user. The N1 Grid OE likewise manages these networked resources on behalf of the users of N1 Grid systems, mapping workload onto them in line with policies.

The workload that is managed is no longer the process or the thread. It becomes the service—a higher-level, more abstract entity that the business understands (for example, an online bookstore or an e-banking service). The policy that is used to manage that service is likewise more abstract.

Rather than specifying scheduling classes, priorities, or amounts of memory to be used by a service component, N1 Grid OE policies will eventually specify the business goals for the service, such as average transaction response time, number of concurrent customers, batch completion window, allowable outages per month, and desired cost. The N1 Grid OE will automatically translate between these higher-level abstractions and will automatically allocate the traditional resources to service components.

For the N1 Grid OE to be able to do this, it must mimic some of the activities that data center architects, managers, and administrators perform today. It must also capture the information they typically use. For example, the N1 Grid OE must have the following:

- Resource model

  The resource model provides a way of describing the set of resources available to the N1 Grid OE. An appropriate internal representation is required, along with the means of manually and automatically populating it.

- Service model

  The service model describes services so that the dependencies between various service components, between service components and resources, and among multiple services can be captured. The dependencies range from simple ones, such as those required to instantiate or run a service component (for example, the right version of the operating system) to performance, scaling, availability, and security. An appropriate internal representation is required, along with the means to manually and automatically populate it.

- Provisioning mechanisms

  The provisioning mechanisms enable the mapping of workloads to resources within the context of the network. Within an N1 Grid system, any given service component has the potential to run on different elements and can be easily moved from one element to another to improve flexibility, availability, and utilization.

- Policy and goals model

  The policy and goals model describes the policies and business goals that drive the system.

- Telemetry and controls interface

  Telemetry and control interfaces enable the state of the various system components, both the hosted service components and the compute, storage, and network elements on which they are deployed, to be monitored and controlled.

Resources

At the core of the N1 Grid vision is the concept that the system is now a network of resources and components, rather than a server connected to the network. This mandates a different perspective on the development and management of the resources. Specifically, resources are treated as components in a fabric. The management silos observed today need to be avoided, at least from the architectural perspective. Although N1 Grid systems can use existing componentry, it is ultimately intended to drive the development of some of these components, optimizing them for operation within an N1 Grid system so that they inherently increase the value of that system. In the long term, this implies the development of standards for the representation of the fabric of resources, together with mechanisms that enable the automatic discovery of fabric components, how they are connected together and how they can interact with one another.

Service-Centric Management

Central to the N1 Grid vision is the concept of moving from the management of low-level entities, such as servers, to the services or applications on which a business is built. By moving the focus of management to services and driving the delivery of those services using business goals, the intention is to remove the management headache associated with managing today's infrastructures.

For the N1 Grid OE to effectively manage the deployment of a service onto the underlying resources at its disposal, it must have an understanding of the nature of the service. FIGURE 3-1 shows a simplified service dependency graph, including the dependencies within a service, between different services within an enterprise, and

between services hosted within different enterprises. The dependency graph must capture instantiation, performance, scaling, availability, and security dependencies, along with others types of dependencies.

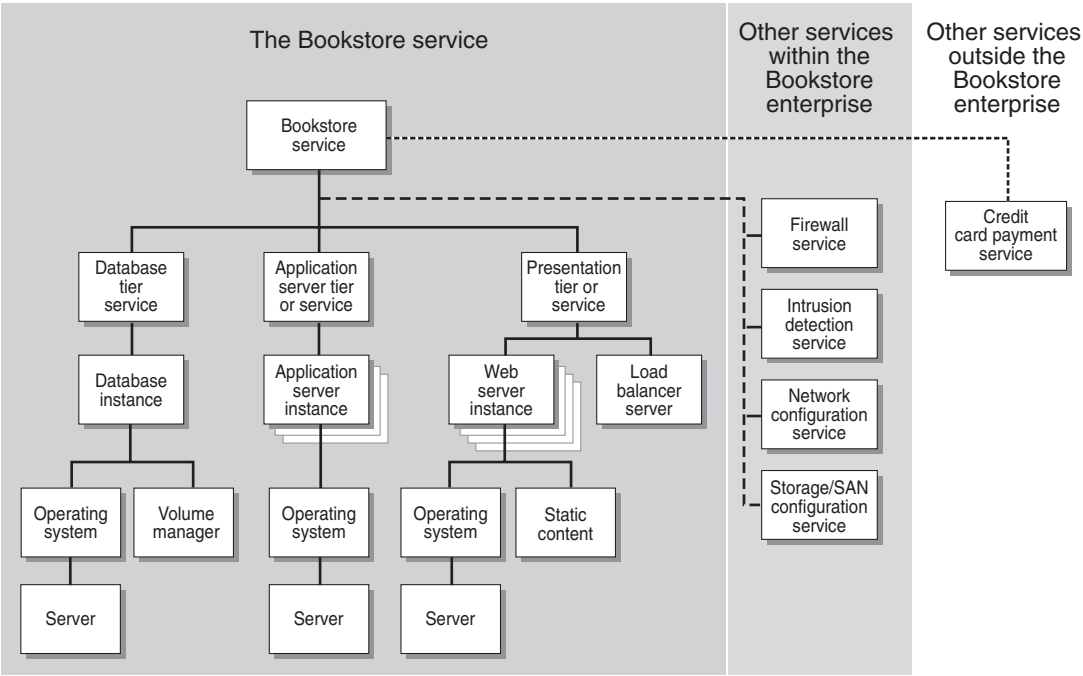

FIGURE 3-1 Simplified Service Dependency Graph

For example, if a Tier 3 service consists of database, application server, and web server components, the N1 Grid OE would need to represent and resolve the following dependencies:

- Sets of instantiation dependencies for the service and its service components, for example:
  - The bookstore depends directly on the database, application server, and web server tiers.
  - The web server tier is running Apache version *X*.
  - The version of Apache requires Linux version *Y* running on IA32-based servers.
  - The web tier uses a load-balancing appliance from a specific vendor.
  - The application server tier uses the Sun Java™ Enterprise System application server version 6.0, which requires the Solaris 9 OS, update *X*, with patches A, B, and C.

  - The database server is version *Z* and requires the Solaris 8 OS, update *X*, with patches E and F.
  - The bookstore has external dependencies on a separate intrusion detection service and firewall service.
- Performance and scaling attributes and dependencies of the service, for example:
  - The web tier scales though replication, and the load is distributed using the load balancer.
  - The application server tier scales on large SMPs, up to 16 processors per instance, as well as across multiple servers using its internal cluster and load balancing capabilities.
  - The database server tier scales as a single instance within an SMP system (perhaps up to 94 processors).
- Availability attributes of the service, such as:
  - The web tier is made highly available through replication and through a load balancer.
  - The application server tier is made highly available using its own clustering capabilities and replication.
  - The database is made highly available by ensuring that it runs only on compute elements with redundant components and within a Sun Cluster software environment.

These are simple examples. In reality, an N1 Grid OE service description would include information about where and how in the service to recover from certain types of component failures, how to ensure appropriate security, or information about the numbers of instances of service components required to cater to certain loads, derived from capacity-planning exercises.

In short, creating the description of the service is the formalization of what are today typically ad hoc processes within data centers. Processes typically vary, not only from one data center or enterprise to another, but often within the same data center. These processes are usually poorly documented and are often manually implemented by people working in disparate groups. Thus, their implementation might be both time consuming and error prone.

With N1 Grid systems and their focus on the description of the service, the emphasis shifts from the repetitive and often manual tasks associated with deployment and ongoing maintenance of components to actually defining and setting up the service properly so that ongoing maintenance can be optimized or automated. Representing a service as a dependency graph enables the manipulation of more abstract entities and provides finer-grained control.

The benefits of service-centric management include:

- It is more intuitive because it is focused on what the business cares about.
- It drives a new sustainable operational model that is not driven by the nature of the underlying componentry, but rather by a consistent view of the services.
- It essentially changes the TCO dynamics of the data center because the components that people manage today become hidden by new tools—and ultimately by the N1 Grid OE.

Workload-to-Resources Mapping

Having described the resources and workload managed by the N1 Grid OE, it is now appropriate to explore how the workload is mapped onto the resources. In a traditional system, the operating system scheduler, memory manager, and device driver components map workloads onto processors, memory, and I/O resources based on simple policies. The operating system virtualizes the underlying resources and provisions the workloads onto them. In an N1 Grid system, *virtualization* and *provisioning* are also fundamentally important mechanisms that enable the N1 Grid OE to map workload onto resource.

Today, both terms are used to describe work being undertaken to solve the data center management problems enumerated in Chapters 1 and 2. However, they are not usually described within the context of a single systemic view. Consequently, it is often unclear how these technologies should be combined or used together to deliver integrated infrastructure solutions. The N1 Grid vision, and specifically the N1 Grid OE, provides a single context for understanding these two important areas: how they are related, and how they can be combined to provide a single context for developing infrastructure solutions.

Virtualization

In general, virtualization is the abstraction of some entity. Typically, virtualization involves adding a layer of software onto some entity so that the new layer exhibits the interface properties of the original entity. However, this layer hides the true implementation of the virtualized object so that the original entity can be changed or replaced without fundamentally impacting how other entities, which have a dependency on it, interact with it. This provides flexibility.

For example, a storage controller might virtualize a raw disk or a collection of raw disks by presenting a logical unit number (LUN). The LUN has all of the interface properties of a raw disk, yet that LUN might be a part of one physical disk or it might be a whole disk or a collection of disks in a RAID stripe. The point is that

whoever or whatever uses the LUN does not need to care about its internal composition. They just use it and automatically take advantage of the properties of a specific underlying implementation (for example, greater performance or availability). Other examples include virtual local area networks (VLANs) and N1 Grid Containers.

Another example of virtualization involves adding a layer of software that changes the level of abstraction of interaction with an object, changing the attributes of the interfaces (for example, to make it more manageable). An example of this would be enabling the N1 Grid OE to manage an application (for example, a database service) based on quality-of-service goals (for example, the average transactional response time), rather than having to manually manage numbers of processors, amount of memory, and I/O allocation directly. Thus, software can be used to translate between the old entity that was managed and the new abstract entity. The administrator of an N1 Grid system manages services, while the N1 Grid OE translates this into the management of more traditional workloads and resources. This means that management scaling can be fundamentally changed and that efficiency, reliability, and agility can be improved through automation.

An operating system or operating environment virtualizes the sets of resources under its control, presenting them to the services that consume them in a consistent fashion (FIGURE 3-2).

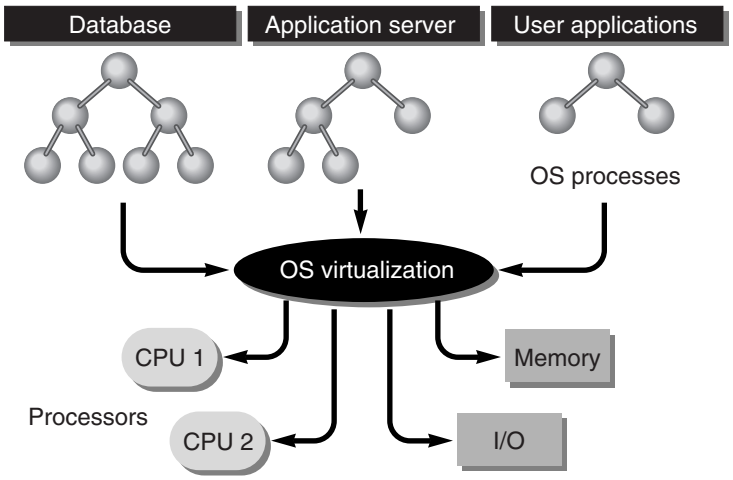

FIGURE 3-2 Traditional Operating System Virtualization

A traditional operating system like the Solaris OS virtualizes processors, memory, and I/O within a large SMP. The operating system maps a workload onto resources in line with policies. The workload resolves to a set of processes or threads that run

on some set of resources. The operating system virtualizes the resources so that they appear equivalent to the application. For example, an application that runs on four processors can run on any four processors within a 32-processor system.

The operating system and the system itself ensure that the identity of physical processors does not matter. Indeed, the actual processors used could change, from one instant to the next, as long as four of them are used. Thus, virtualization turns a collection of discrete, identical resources into a pool of shared resources. This enables greater flexibility and utilization of those resources because they can be efficiently shared. For example, two applications could run and use up all of the 32 processors. However, at any instant, application A could be using any four of the processors, and application B could be using 28. In another instant, application A could be using ten processors, and application B could be using 22. The system decides how many, and which specific ones, to use on behalf of the service, removing complexity and improving performance and utilization.

The N1 Grid system analogy of this (see FIGURE 3-3) is that the N1 Grid OE can turn 200 blade servers into a single pool of resources so that you no longer care which 12 are used for web server instances within a multitier bookstore service. The N1 Grid OE ensures that the right number, that is 12, are used to meet the business goals of the service.

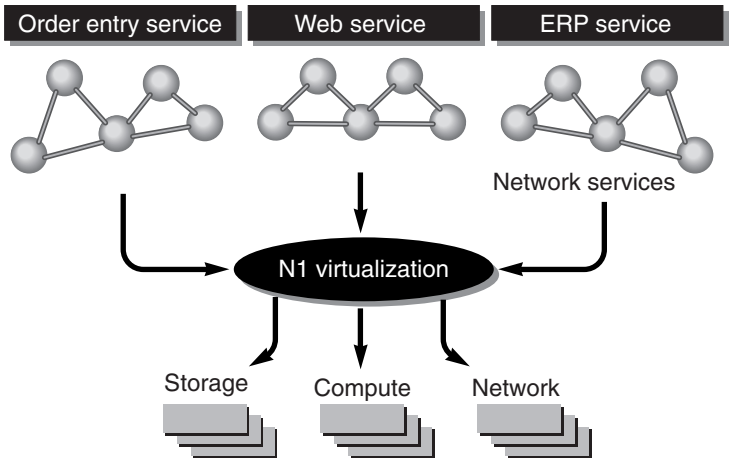

FIGURE 3-3 N1 Grid Operating Environment Virtualization

FIGURE 3-4 shows the most basic view of virtualization. If you think about both previously described definitions of virtualization, then you can view the data center as a series of layers of components, each depending on those below it and each virtualizing those below them, either through simply separating the logical properties from the physical entities or through changing the level of abstraction to present new properties to the layers above. Each layer of virtualization provides the opportunity to hide complexity and improve flexibility and efficiency.

Unable to access graphic file. File is apparently not in standard EPS format.

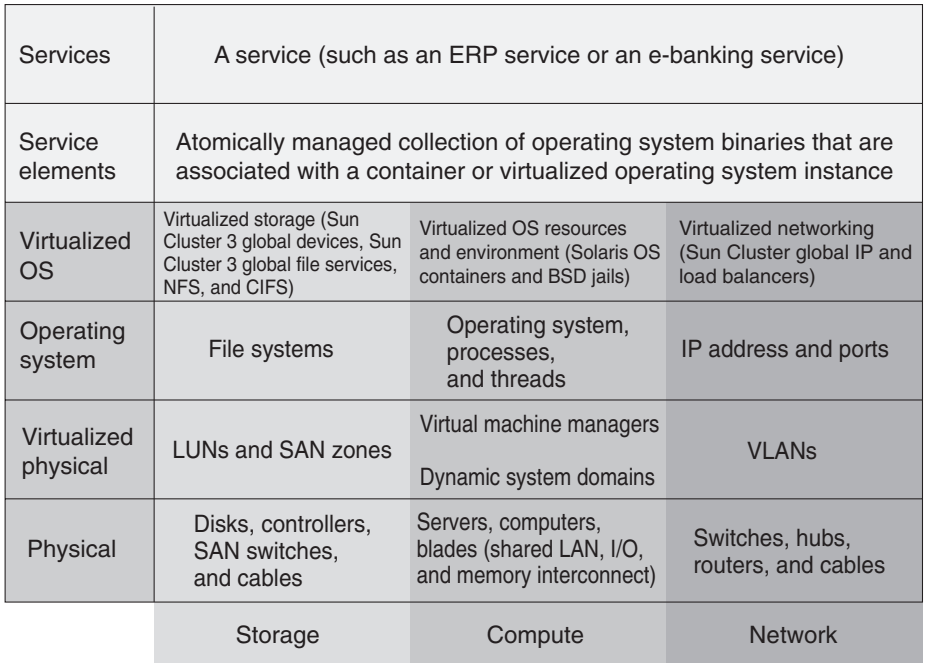

FIGURE 3-4 Layers of Virtualization

Applying this technique to the N1 Grid system means that several of the problems associated with managing data center infrastructures can be addressed. These problems include:

- Removing complexity

  This improves management scaling, reliability, availability, and security, and it reduces costs and risk.

- Increasing flexibility

  This enables faster repurposing to recover from failure, faster repurposing to cater to load or goal variation, and faster time to market in deploying new services through the pipeline.

Provisioning Services

The systemic approach, combined with virtualization techniques, also changes the perspective on provisioning services or applications. Provisioning is the act of taking a service, installing it, and ending up with a running service on a system or collection of systems.

Because most provisioning today has a considerable manual element to it and is typically *silo-ed* (for example, into network, storage, and server-related aspects), it is risk laden, time consuming, and expensive. This results in a reluctance to repurpose infrastructure components, which in turn leads to static environments with poor utilization and a lack of flexibility. The model for provisioning is server-centric, rather than system-centric. If you take a system-centric approach, these problems can be resolved.

Provisioning typically requires the following sequence of events:

1. Copying the bits somewhere (for example, from a compact disc to storage, associated with a specific operating system instance)
2. Binding them to the underlying platform instance (base hardware or hardware and operating system stack, if already configured)
3. Turning them into a service-specific instance
4. Instantiating them (that is, running them)

Typically, this sequence is done in one or two steps, whereas logically, it might be better to break it down into several separate steps. FIGURE 3-5 shows a comparison of traditional and N1 Grid OE-based steps of provisioning. The new sequence, and tools that automate it, enable more dynamic provisioning in response to load or policy changes.

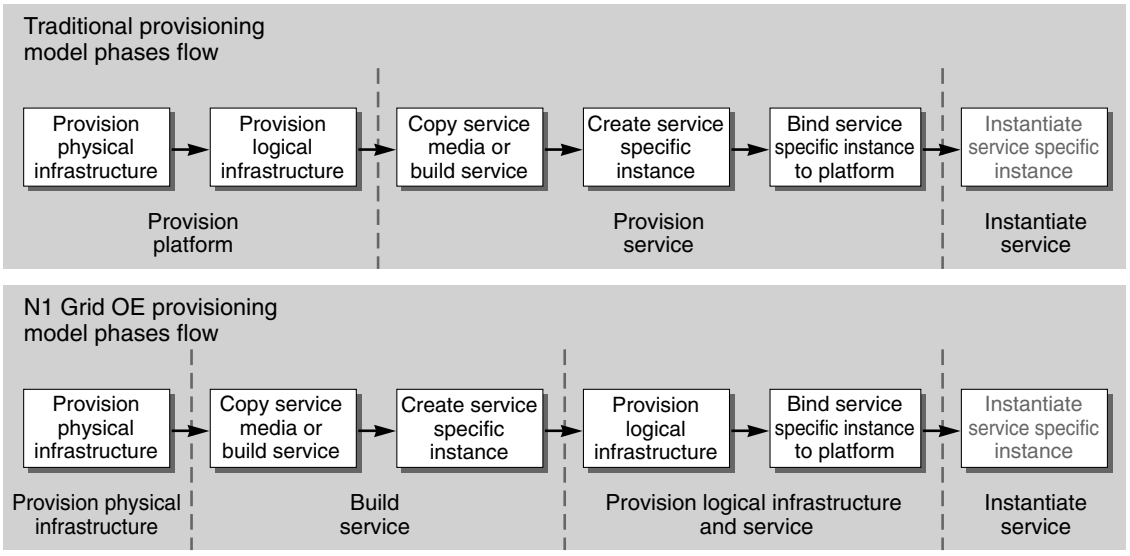

FIGURE 3-5 Flow Between Provisioning Model Phases

Provisioning today is viewed as server-centric. However, it is in fact system-centric, but in the sense of the old system—a server. For example, when installing a database that requires four processors on a 24-processor system, the software is installed to the system, not to four specific processors. Then, the system instantiates the database on the four processors of its choosing, based on policies and goals.

Extending this model to an N1 Grid system, the database is installed to the system, in this case the N1 Grid system. It is then instantiated on any four processors of any platform that meets the dependency requirements of the database. For example, it might require the Solaris 8 OS with patches X, Y, and Z.

The model is in essence the same, if you think in terms of a system. However, the sequence of events changes, or rather, it becomes important. In a traditional view, the ordering of binding to the platform instance and turning a generic component into a specific service instance (for example, turning a generic database installation into the database for a bookstore by creating the table spaces and loading them) is not important. Both have to be done after installing the bits and before running the instance.

The order matters when provisioning within the context of the N1 Grid system. By creating a service-specific instance that is not bound to a specific platform or operating system instance, you end up with a service component (in this case a database) that is able to run on a number of potential platform components. The target platform can be decided at runtime, based on available resources and a comparison of the goals and priorities of this service versus those of others hosted within the same N1 Grid system. Indeed, the target platform can be easily changed if the initial one no longer meets the resource requirements of the service component, due to load or policy changes.

In a traditional system, the database is installed to the system that is a server, and the system runs it on any of the processors it deems appropriate. In an N1 Grid system, the database is installed to the system that is a network. Thus, it is installed on shared storage so that it can be run on any four processors of any server that has the right underlying instruction set architecture and operating system version. Ultimately, it is virtualization that enables this dynamic binding and rebinding.

As an aside, it should be obvious that Java and the J2EE platform become the ultimate in server virtualization technologies. The target platform for a Java or Enterprise JavaBeans™ (EJB™) component then becomes any J2EE application server that supports the appropriate version of Java or the J2EE platform. Thus, the compute elements are effectively made homogeneous, and maximum flexibility and choice of deployment is achieved.

Virtualization and Provisioning Examples

The following are some simple examples that illustrate the potential of combining virtualization with provisioning, using functionality available today, but within the context of the N1 Grid solutions.

Provisioning a Service Component on an Existing Operating System

This section contains an example of provisioning a service component, such as a database, on an existing operating system instance.

A database can be installed, and table spaces created and loaded, on shared storage, such as network-attached storage (NAS) or SAN storage, so that it can be instantiated on any one of a set of servers, as long as each meets the database requirements in terms of operating system and patch level and each is able to access the storage.

The database can also be provided with its own IP address and host name, independent of those associated with the underlying server, so that they can *move* with the database service to wherever it is instantiated.

After it has been installed and configured on an initial host environment, the set of variables and configuration parameters that associate it with a specific operating system instance and underlying platform need to be noted. Then, the database can be stopped.

When the database is to be instantiated, the storage containing the database application and that containing the data can be associated with the chosen, preinstalled, and already running target operating system instance. Then, the appropriate operating system tuning can be done, together with the configuration of any other operating system instance-specific dependencies of the database (for example, an IP address).

If the database is to run in a shared environment (that is, one in which the hosting operating system is also hosting one or more other service components), this configuration can be applied to an operating system partition, such as an N1 Grid Container.

Virtualization of the storage environment and of the network, combined with the N1 Grid vision of provisioning, results in a database service that can be instantiated on any server running the correct operating system version and set of patches. This enables the database to be stopped and restarted on a different compute element in the event of changing resource requirements or in the event of underlying platform failures. Repurposing becomes automated and reliable enough to enable greater agility, potentially greater availability, and greater resource utilization.

Provisioning a Service Component as Part of a Complete Bootable Image

This section contains an example of provisioning a service component, such as an application server, as part of a complete bootable image.

Consider an application server instance that is part of a bootable image in a SAN environment. Again, this complete bootable image could be created on a reference server, and all instance-specific tunables and configuration parameters noted. The application server could be a part of a cluster of application servers that act as a front end by a load balancer of some description.

This bootable stack could then be booted on one or more compute elements. If the compute elements are identical, little modification to the boot image would be required during the boot and application server instantiation.

As with the first example, this method results in greater agility, especially if management of the load balancing or application server clustering functionality is automated and integrated with the application server instantiation.

Both of these examples illustrate that the principal behavioral aspects of an N1 Grid system can be realized today. The reasons they are often not are that too many tools must be used to implement this behavior, and because service component instance-specific configuration parameters and tuning information are often not properly understood or noted. In short, implementation is complex and typically a manual process. Thus, it is time consuming and error prone. These implementations become viable only after the configuration information can be formally captured and an integrated mechanism can be provided for associating and binding a service component with an operating system instance or for associating a stack with a compute element. Combining virtualization techniques with a different focus on provisioning results in the ability to provision and reprovision service components dynamically, resulting in better resource utilization, agility, and availability.

Separating the aspect of provisioning that creates a service-specific instance of an application (for example, the creation of the database for the bookstore) from the aspect that binds it to the operating system instance it is to run on (that is, instantiation) is of fundamental importance. Combining this with virtualization to provide flexible mapping between the service-specific instance and an appropriate underlying resource enables enormous flexibility.

Many of the mechanisms required to implement these aspects of an N1 Grid OE exist today. However, to actually deliver this functionality and consequent value, a consistent context is required, and many tools are required to be manipulated or managed, often manually. The lack of such a context and the complexity in terms of available tools result in risk. Thus, infrastructure architectures and implementations that leverage all of the available mechanisms are rarely realized. The goal of the N1 Grid strategy is to both automate this binding or provisioning and to automate the

decision-making process that determines what resources to allocate to the service components and how to change them to meet the high-level business goals for the service.

N1 Grid Systems Realized

Having explored some of the attributes and properties of N1 Grid systems and focused on the N1 Grid OE, this section explores what it means to realize an N1 Grid system. An N1 Grid system can be considered from two perspectives:

- As a system, which is defined by its *attributes*—that is, composed of a set of network-distributed hardware and software components that are virtualized and that behave as a single system

  This system is managed as a system, rather than as its discrete, individual components, and it is managed in a service-centric fashion.

- As a system, which is defined by its *implementation*—that is, adhering to the N1 Grid principles of operation

  This system is a realization of the N1 Grid architecture in which all of the architectural elements are realized as technology components that communicate with one another using standard IP protocols.

The second definition is more rigorous and describes a full implementation, yet there needs to be a means of evolving to this state with minimum disruption to existing data center infrastructure and practices.

Route to the N1 Grid Solutions

Infrastructure that realizes many of the basic attributes of a full N1 Grid system implementation can be constructed today by using both existing and custom components, and coordinated by using people and process. This is because most of the control and telemetry functionality required for the N1 Grid OE exists in a number of discrete products and tools.

Much of the value of an N1 Grid system is realized through its operational model (that is, the way in which it is managed). A user interface could be constructed that would be consistent with the full N1 Grid realization and that reflects the operational model. That interface could enable services and the service descriptions, together with the service-level objectives, to be defined and the underlying resource fabric to be described. All of this information could be stored in a traditional form, such as a database or spreadsheet.

The policies described could then be applied using people and process, where technology is unavailable. The process would be consistent with the N1 Grid principles of operation. When new technology is created that automates a new function, it could be integrated with the existing implementation.

This method means that a consistent management model can be instantiated. As new technology is rolled into the implementation, the management model need not change. All that would happen is that some tasks would disappear while the system managers would still interact through the user interface. Thus, there would be no need for them to acquire new skills every time a new implementation of an underlying mechanism is delivered. As technology is delivered, the overall abilities or attributes of the system improve as various tasks are automated (FIGURE 3-6). In fact, this could be considered virtualization of the data center management processes themselves or of the operational model implementation.

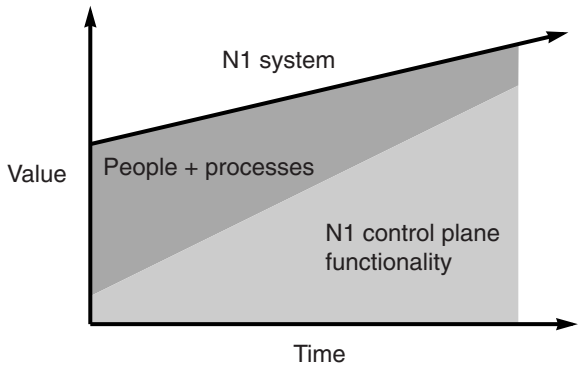

FIGURE 3-6 The Evolution of an N1 Grid System Implementation

Value of N1 Grid Systems

What value will fully implemented N1 Grid systems, as well as evolving N1 Grid implementations, actually deliver? The following lists the value:

- TCO improvements through high-level policy-based management

  This changes management scaling by hiding complexity so that rather than being based on numbers of servers, server vendors, number of operating system instances, versions and vendors, number of tools, and number of applications managed, it scales with the number and complexity of the services being offered.

- TCO improvements through the application of a consistent operational model

  The model reduces the need to continually invest in re-skilling operations staff in continually changing, upgraded product and tool implementations.

- Improved efficiency and reliability through the formalization of many aspects of data center management

  Currently, these aspects are ad hoc (for example, service life cycle management) and error prone. Ultimately, good practices and process become a part of the technology, in this case the N1 Grid OE.

- Higher efficiency or resource utilization

  Virtualization of individual servers, operating system instances, networked resources, and the automated mapping and remapping (provisioning) of workloads onto these resources, ensures higher utilization of resources.

- Greater agility and responsiveness, from the definition and deployment of new services to changing resource allocations rapidly in response to business goals

  Capacity planning becomes simpler because mistakes are more easily and dynamically corrected.

- Greater reliability and availability through reduction of exposed complexity and automation of routine component-centric tasks

Although N1 Grid systems are intended to solve the most pressing problems within the data center, they do so in a manner that changes the perception of IT and that changes how services and the infrastructure they run on are paid for. Specifically, N1 Grid systems ultimately deliver transparency in terms of value delivered versus cost.

N1 Grid systems are driven by business goals, expressed as policies that cause workloads to be mapped and remapped as appropriate onto resources. To achieve this, the N1 Grid OE has to be able to reconcile resource consumption and the cost of those resources with the services that consume them. This is required to manage the quality of service goals, but it also means that N1 Grid systems should exhibit a previously unheard-of transparency with regards to reconciling the value delivered by a service and its components and the costs of providing that value. This type of transparency enables new business choices to be made, or at the very least, it enables existing business choices to be made with the right data.

For example, N1 Grid systems enable *utility computing*. Although utility computing is often referred to as a technology paradigm, it is probably best to think about it as a business paradigm associated with the delivery of services. Specifically, IT infrastructures, and the services that run on them, should be paid for in a way that is proportional to their delivered value. Thus, a service provider delivering a payroll service could charge per transaction or per employee on the payroll. They in turn could pay for their resources, both traditional hardware and metaresources, such as a database or an application server, based on usage. Traditional platform usage could be based on usage of certain resources, such as processors, memory, and storage. An application server could then be paid for based on transaction throughput and complexity.

The point is that after the telemetry to reconcile resource consumption with the services that consume them is available, there is sufficient data to enable more interesting and more flexible business relationships (for instance, relationships between system vendors and their customers, between service providers and their customers within large enterprises, and between data center staff and the business).

The potential exists for a new relationship between information technology providers and consumers. It would be interesting if when a CEO approaches the CIO and says "I'm afraid we need to cut your budget by 10 percent this year," the CIO could confidently say "Fine, however, you will lose this level of performance or capacity for your main customer-facing services. Marketing tells us that would translate into a 15 percent reduction in our customer base due to dissatisfaction." Indeed, a more interesting conversation, one which probably rarely occurs today, would be when the CIO says "If you invest *x* amount of money into our infrastructure, I can improve responsiveness and reduce outages by *y*, which means that we would over take our chief competitor" or "If you invest *x*, then we can bring new services to market 50 percent faster than our competitors."

Ultimately, N1 Grid systems, and those like them, will turn arcane technology infrastructures into genuine business systems, where the work they undertake is easily understood by the business and where much of the day-to-day management of the services they host is undertaken by line-of-business people, rather than IT specialists. Over time, it becomes the role of the IT specialist to install and deploy new services on N1 Grid systems, to describe them to N1 Grid systems so that the N1 Grid OE and the line of business can do the day-to-day management of the service, and to maintain the N1 Grid system.

Summary

This chapter built on the fundamental basics of the N1 Grid solution. It explored the nature of an N1 Grid operating environment, especially some of the mechanisms that are already extant and are leveraged by the N1 Grid solution. The rest of the book focuses on how to build infrastructure solutions that are consistent with the N1 Grid principles of operation and architecture.