UNIT III VIRTUALIZATION

Cloud deployment models: public, private, hybrid, community – Categories of cloud computing: Everything as a service: Infrastructure, platform, software - Pros and Cons of cloud computing – Implementation levels of virtualization – virtualization structure – virtualization of CPU, Memory and I/O devices – virtual clusters and Resource Management – Virtualization for data center automation.

PART A

1) What is private cloud?

The *private cloud* is built within the domain of an intranet owned by a single organization. Therefore, they are client owned and managed. Their access is limited to the owning clients and their partners. Their deployment was not meant to sell capacity over the Internet through publicly accessible interfaces. Private clouds give local users a flexible and agile private infrastructure to run service workloads within their administrative domains.

2) What is public cloud?

A *public cloud* is built over the Internet, which can be accessed by any user who has paid for the service. Public clouds are owned by service providers. They are accessed by subscription. Many companies have built public clouds, namely Google App Engine, Amazon AWS, Microsoft Azure, IBM Blue Cloud, and Salesforce Force.com. These are commercial providers that offer a publicly accessible remote interface for creating and managing VM instances within their proprietary infrastructure.

3) What is hybrid cloud?

A *hybrid cloud* is built with both public and private clouds, Private clouds can also support a *hybrid cloud* model by supplementing local infrastructure with computing capacity from an external public cloud. For example, the *research compute cloud* (RC2) is a private cloud built by IBM.

4) What is a Community Cloud ?

A **community cloud** in computing is a collaborative effort in which infrastructure is shared between several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.), whether managed internally or by a third-party and hosted internally or externally. This is controlled and used by a group of organizations that have shared interest. The costs are spread over fewer users than a public cloud (but more than a private cloud

5) What is cloud service management?

Cloud Service Management includes all of the service-related functions that are necessary for the management and operation of those services required by or proposed to cloud consumers.

6) What is on demand of cloud computing

On-demand (OD) computing is an increasingly popular enterprise model in which computing resources are made available to the user as needed. The resources may be maintained within the user's enterprise, or made available by a service provider.

7) What are the major actors in NIST cloud reference architecture?

The NIST cloud computing reference architecture defines five major actors: *cloud consumer, cloud provider, cloud carrier, cloud auditor* and *cloud broker*. Each actor is an entity (a person or an organization) that participates in a transaction or process and/or performs tasks in cloud computing.

8) What is the role of cloud consumer in cloud computing?

A person or organization that maintains a business relationship with, uses service from, *Cloud Providers.*A person, organization, or entity responsible for making a service available to interested parties.

9) What is the role of cloud auditor in cloud computing?

A party that can conduct independent assessment of cloud information system operations, performance and security of the implementation.

10) What is the role of cloud broker in cloud computing?

The cloud broker is an entity that manages the use, performance and delivery of cloud services, and negotiates relationships between *Cloud Providers* and *Cloud Consumers.*

11) What is the role of cloud carrier in cloud computing?

The cloud carrier is an intermediary that provides connectivity and transport of services from *Cloud Providers* to *Cloud Consumers.*

12) Define IaaS?

The IaaS layer offers storage and infrastructure resources that is needed to deliver the Cloud services. It only comprises of the infrastructure or physical resource. Top IaaS Cloud Computing Companies: Amazon (EC2), Rackspace, GoGrid, Microsoft, Terremark and Google.

13) Define PaaS?

PaaS provides the combination of both, infrastructure and application. Hence, organisations using PaaS don't have to worry for infrastructure nor for services. Top PaaS Cloud Computing Companies: Salesforce.com, Google, Concur Technologies, Ariba, Unisys and Cisco..

14) Define SaaS?

In the SaaS layer, the Cloud service provider hosts the software upon their servers. It can be defined as a in model in which applications and softwares are hosted upon the server and made available to customers over a network. Top SaaS Cloud Computing Companies: Amazon Web Services, AppScale, CA Technologies, Engine Yard, Salesforce and Windows Azure.

15) Differentiate SAN, LAN, WAN?

LAN - Local Area Network	**WAN - Wide Area Network**	**(SAN) - Storage Area Network**
A LAN connects network devices over a relatively short distance. A networked office building, school, or home usually contains a single LAN, though sometimes one building will contain a few small LANs (perhaps one per room), and occasionally a LAN will span a group of nearby buildings. In TCP/IP networking, a LAN is often but not always implemented as a single IP subnet.	WAN spans a large physical distance. The Internet is the largest WAN, spanning the Earth. A WAN is a geographically-dispersed collection of LANs. A network device called a router connects LANs to a WAN. In IP networking, the router maintains both a LAN address and a WAN address.	(**SAN**) is a dedicated network that provides access to consolidated, block level data storage. SANs are primarily used to enhance storage devices, such as disk arrays, tape libraries, and optical jukeboxes, accessible to servers so that the devices appear like locally attached devices to the operating system

16) What is meant by virtualization?

Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine. The idea of VMs can be dated back to the 1960s. The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility.

17) What are the implementation levels of virtualization?

The virtualization types are following

1. OS-level virtualization
2. ISA level virtualization
3. User-Application Level virtualization
4. hardware level virtualization
5. library level virtualization

18) What is OS-level virtualization?

OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hard-ware and software in data centers. The containers behave like real servers. OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users. It is also used, to a lesser extent, in consolidating server hardware by moving services on separate hosts into containers or VMs on one server.

19) What is ISA level virtualization?

At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. For example, MIPS binary code can run on an x86-based host machine with the help of ISA emulation. It is possible to run a large amount of legacy binary code writ-ten for various processors on any given new hardware host machine. Instruction set emulation leads to virtual ISAs created on any hardware machine

20) What is User-Application Level virtualization (or) Process level virtualization?

On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process-level virtualization. The most popular approach is to deploy high level language (HLL) VMs. the virtualization layer sits as an application program on top of the operating system, and the layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition. The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM.

21) What is hardware level virtualization?

Hardware-level virtualization is performed right on top of the bare hardware.this approach generates a virtual hardware environment for a VM. On the other hand, the process manages the underlying hardware through virtualization.

22) What is library level virtualization?

Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks. The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts. Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration

23) What are the types of VMM Design requirements?

There are three requirements for a VMM.

- First, a VMM should provide an environment for pro-grams which is essentially identical to the original machine.
- Second, programs run in this environment should show, at worst, only minor decreases in speed.
- Third, a VMM should be in complete control of the system resources.

24) What are the advantages of OS level Virtualization?

Compared to hardware-level virtualization, the benefits of OS extensions are twofold:

- VMs at the operating system level have minimal startup/shutdown costs, low resource requirements, and high scalability
- For an OS-level VM, it is possible for a VM and its host environment to synchro-nize state changes when necessary.

25) What are disadvantages of OS level Virtualization?

- The main disadvantage of OS extensions is that all the VMs at operating system level on a single container must have the same kind of guest operating system.
- That is, although different OS-level VMs may have different operating system distributions, they must pertain to the same operating system family.
- For example, a Windows distribution such as Windows XP cannot run on a Linux-based container. However, users of cloud computing have various preferences.
- Some prefer Windows and others prefer Linux or other operating systems. Therefore, there is a challenge for OS-level virtualization in such cases.

26) Write short notes on xen architecture?

- Xen is an open source hypervisor program developed by Cambridge University. It just provides a mechanism by which a guest OS can have direct access to the physical devices. As a result, the size of the Xen hypervisor is kept rather small.
- Xen provides a virtual environment located between the hardware and the OS.A number of vendors are in the process of developing commercial Xen hypervisors, among them are Citrix XenServer and Oracle VM

27) What are the types of hardware virtualizatiion?

Depending on implementation technologies, hardware virtualization can be classified into two categories: full virtualization and host-based virtualization.

28) Write short notes on KVM (Kernel-Based VM)

- This is a Linux para-virtualization system—a part of the Linux version 2.6.20 kernel.
- Memory management and scheduling activities are carried out by the existing Linux kernel.
- The KVM does the rest, which makes it simpler than the hypervisor that controls the entire machine. KVM is a hardware-assisted para-virtualization tool, which improves performance and supports unmodified guest OSes such as Windows, Linux, Solaris, and other UNIX variants

29) Write short notes about memory virtualization?

- Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory.
- All modern x86 CPUs include a memory management unit (MMU) and a translation look aside buffer (TLB) to optimize virtual memory performance.

30) Write short notes about CPU virtualization

- A CPU architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode.
- When the privileged instructions including control- and behavior-sensitive instructions of a VM are exe-cuted, they are trapped in the VMM.

31) What are the three categories of critical instructions?

The critical instructions are divided into three categories: privileged instructions, control-sensitive instructions, and behavior-sensitive instructions.

- **Privileged** instructions execute in a privileged mode and will be trapped if executed outside this mode.
- **Control-sensitive** instructions attempt to change the configuration of resources used.
- **Behavior-sensitive** instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.

32) What is Hardware-Assisted CPU Virtualization?

This technique attempts to simplify virtualization because full or paravirtualization is complicated. Intel and AMD add an additional mode called privilege mode level (some people call it Ring-1) to x86 processors. Therefore, operating systems can still run at Ring 0 and the hypervisor can run at Ring -1.

All the privileged and sensitive instructions are trapped in the hypervisor automatically. This technique removes the difficulty of implementing binary translation of full virtualization. It also lets the operating system run in VMs without modification

33) What is IO virtualization?

I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware.

34**) What are the ways to implement IO VIRTUALIZATION?**

There are three ways to implement I/O virtualization:

- Full device emulation,
- Para-virtualization,
- Direct I/O.

35) Write short notes on Full device emulation?

- Full device emulation is the first approach for I/O virtualization. Generally, this approach emulates well-known, real-world devices. All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated in software. This software is located in the VMM and acts as a virtual device.
- The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices. A single hardware device can be shared by multiple VMs that run concurrently.However, software emulation runs much slower than the hardware it emulates

36) Write short notes on para-virtualization?

- The para-virtualization method of I/O virtualization is typically used in Xen. It is also known as the split driver model consisting of a frontend driver and a backend driver. The frontend driver is running in Domain U and the backend driver is running in Domain 0. They interact with each other via a block of shared memory.
- The frontend driver manages the I/O requests of the guest OSes and the backend driver is responsible for managing the real I/O devices and multiplexing the I/O data of different VMs. Although para-I/O-virtualization achieves better device performance than full device emulation, it comes with a higher CPU overhead.

37) Write short notes on para-virtualization

Direct I/O virtualization lets the VM access devices directly. It can achieve close-to-native performance without high CPU costs. However, current direct I/O virtualization implementations focus on networking for mainframes.

38) What are virtual clusters?

Virtual clusters are built with VMs installed at distributed servers from one or more physical clusters. The VMs in a virtual cluster are interconnected logically by a virtual network across several physical networks.

39) What is network migration?

- A migrating VM should maintain all open network connections without relying on forwarding mechanisms on the original host or on support from mobility or redirection mechanisms.
- To enable remote systems to locate and communicate with a VM, each VM must be assigned a virtual IP address known to other entities.
- This address can be distinct from the IP address of the host machine where the VM is currently located. Each VM can also have its own distinct virtual MAC address.

40) What is data center?

In data centers, a large number of heterogeneous workloads can run on servers at various times.These heterogeneous workloads can be roughly divided into two categories: chatty workloads and noninter-active workloads. Chatty workloads may burst at some point and return to a silent state at some other point.

41) **Short Notes on levels of virtualization**?

Five virtualization techniques or levels:

Hardware Virtualization – This is the use of computing hardware in an environment separate from the actual existence of the hardware. There are different levels of hardware virtualization including full, partial and paravirtualization. Each has its own advantages and unique characteristics.

Virtual Machine – A virtual machine is a software clone of a real computer that executes programs as if it were the computer. Other types of software virtualization techniques include virtual appliance, application virtualization, cross-platform virtualization and OS virtualization.

Storage Virtualization – Just as the name implies, the focus is on separating physical storage from actual storage.
Desktop Virtualization – The ability to operate a computing environment remotely.
Network Virtualization – The creation of work space within a larger network or across networks using virtualization techniques

42) Define virtual machine monitor?

A Virtual Machine Monitor (VMM) is a software program that enables the creation, management and governance of virtual machines (VM) and manages the operation of a virtualized environment on top of a physical host machine. VMM is also known as Virtual Machine Manager and Hypervisor.

43) **What is Instruction Set Architecture level of virtualization?**

At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. Instruction set emulation leads to virtual ISAs created on any hardware machine. The basic emulation method is through code interpretation. An interpreter program interprets the source instructions to target instructions one by one. A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler

44) **What is hardware abstraction level of virtualization?**

Hardware-level virtualization is performed right on top of the bare hardware. On the one hand, this approach generates a virtual hardware environment for a VM. On the other hand, the process manages the underlying hardware through virtualization. The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices. The intention is to upgrade the hardware utilization rate by multiple users concurrently.

46) What is operating system level of virtualization?

This refers to an abstraction layer between traditional OS and user applications. OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hardware and software in data centers. The containers behave like real servers. OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users. It is also used, to a lesser extent, in consolidating server hardware by moving services on separate hosts into containers or VMs on one server.

47) What is library support level of virtualization?

Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks. The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts. Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.

48) What is user application level of virtualization?

Virtualization at the application level virtualizes an application as a VM. On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process-level virtualization. The most popular approach is to deploy high level language (HLL) VMs. Other forms of application-level virtualization are known as application isolation, application sandboxing, or application streaming. The process involves wrapping the application in a layer that is isolated from the host OS and other applications.

48) Write the relative merits of virtualization at various levels?

Table 3.1 Relative Merits of Virtualization at Various Levels (More "X"'s Means Higher Merit, with a Maximum of 5 X's)

Level of Implementation	Higher Performance	Application Flexibility	Implementation Complexity	Application Isolation
ISA	X	XXXXX	XXX	XXX
Hardware-level virtualization	XXXXX	XXX	XXXXX	XXXX
OS-level virtualization	XXXXX	XX	XXX	XX
Runtime library support	XXX	XX	XX	XX
User application level	XX	XX	XXXXX	XXXXX

49) List the requirements of VMM?

There are three requirements for a VMM.

First, a VMM should provide an environment for programs which is essentially identical to the original machine.

Second, programs run in this environment should show, at worst, only minor decreases in speed.

Third, a VMM should be in complete control of the system resources.

50) Differentiate full virtualization and para-virtualization?

Full Virtualization	**Para Virtualization**
Noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software. Both the hypervisor and VMM approaches are considered full virtualization. Running noncritical instructions on hardware not only can promote efficiency, but also can ensure system security	Para-virtualization needs to modify the guest operating systems. A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications. The guest operating systems are para-virtualized. They are assisted by an intelligent compiler to replace the non-virtualizable OS instructions by hypercalls.

51) Explain Host OS and Guest OS?

A comparison of the differences between a host system, a guest system, and a virtual machine within a virtual infrastructure.

A **host system (host operating system)** would be the primary & first installed operating system. If you are using a bare metal Virtualization platform like Hyper-V or ESX, there really isn't a host operating system besides the Hypervisor. If you are using a Type-2 Hypervisor like VMware Server or Virtual Server, the host operating system is whatever operating system those applications are installed into.

A **guest system (guest operating system)** is a virtual guest or virtual machine (VM) that is installed under the host operating system. The guests are the VMs that you run in your virtualization platform.

52) Define Hypervisor and Xen Server?

The hypervisor supports hardware-level virtualization on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physical hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor. The hypervisor provides hypercalls for the guest OSes and applications.

Xen is a microkernel hypervisor, which separates the policy from the mechanism. The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0. Xen does not include any device drivers natively . It just provides a mechanism by which a guest OS can have direct access to the physical devices

53) What is mean by host based virtualization?

Hardware virtualization can be classified into two categories: full virtualization and host-based virtualization.

Full virtualization does not need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive, nonvirtualizable instructions. The guest OSes and their applications consist of noncritical and critical instructions. In a **host-based system**, both a host OS and a guest OS are used. A virtualization software layer is built between the host OS and guest OS. These two classes of VM architecture are introduced next.

54) What are the advantages and disadvantages of OS extension?

Advantages	Disadvantages
• Flexible provisioning • Support of multiple images per system, including boot menu • Rapid Software (OS/Apps) Deployment • System are always 100% identical • Easy implementation of updates and hotfixes of the Operating System and Applications • Easy rollback scenarios • After restarting, the system is back to a clean state	• No work off-line capability • High-speed LAN recommended (>100Mb) • Not all operating systems are supported • Multiple PXE/BootP solutions in same network segment will cause issues • Imaging disadvantages apply to this technique

55) Define KVM?

KVM (Kernel-Based VM)

This is a Linux para-virtualization system—a part of the Linux version 2.6.20 kernel. Memory management and scheduling activities are carried out by the existing Linux kernel. The KVM does the rest, which makes it simpler than the hypervisor that controls the entire machine. KVM is a hardware-assisted para-virtualization tool, which improves performance and supports unmodified guest OSes such as Windows, Linux, Solaris, and other UNIX variants

56) Write the steps for live VM migration?

The five steps for live VM migration is

Stage 0: *Pre-Migration*

Active VM on Host A

Alternate physical host may be preselected for migration

Block devices mirrored and free resources maintained

Stage 1: *Reservation*

Initialize a container on the target host

Stage 2: *Iterative pre-copy*

Enable shadow paging

Copy dirty pages in successive rounds.

Stage 3: *Stop and copy*

Suspend VM on host A

Generate ARP to redirect traffic to Host B

Synchronize all remaining VM state to Host B

Stage 4: *Commitment*

VM state on Host A is released

Stage 5: *Activation*

VM starts on Host B

Connects to local devices

Resumes normal operation

57) **What is memory migration?**

Moving the memory instance of a VM from one physical host to another can be approached in any number of ways. Memory migration can be in a range of hundreds of megabytes to a few gigabytes in a typical system today, and it needs to be done in an efficient manner. The Internet Suspend-Resume (ISR) technique

exploits temporal locality as memory states are likely to have considerable overlap in the suspended and the resumed instances of a VM.

58) **Define file system migration?**

To support VM migration, a system must provide each VM with a consistent, location-independent view of the file system that is available on all hosts. A simple way to achieve this is to provide each VM with its own virtual disk which the file system is mapped to and transport the contents of this virtual disk along with the other states of the VM. However, due to the current trend of highcapacity disks, migration of the contents of an entire disk over a network is not a viable solution.

59) What is network migration?

A migrating VM should maintain all open network connections without relying on forwarding mechanisms on the original host or on support from mobility or redirection mechanisms. To enable remote systems to locate and communicate with a VM, each VM must be assigned a virtual IP address known to other entities. This address can be distinct from the IP address of the host machine where the VM is currently located

60) Explain the server consolidation?

In data centers, a large number of heterogeneous workloads can run on servers at various times. These heterogeneous workloads can be roughly divided into two categories: chatty workloads and non-interactive workloads. Chatty workloads may burst at some point and return to a silent state at some other point. on-interactive workloads do not require people's efforts to make progress after they are submitted.

61) Define the three resource managers?

- Instance Manager controls the execution, inspection, and terminating of VM instances on the host where it runs.
- Group Manager gathers information about and schedules VM execution on specific instance managers, as well as manages virtual instance network.
- Cloud Manager is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, makes scheduling decisions, and implements them by making requests to group managers

62) Define trust management in cloud computing?

A VMM can provide secure isolation and a VM accesses hardware resources through the control of the VMM, so the VMM is the base of the security of a virtual system. Normally, one VM is taken as a management VM to have some privileges such as creating, suspending, resuming, or deleting a VM

PART B

1) Explain with a neat diagram public, private, hybrid and community clouds?

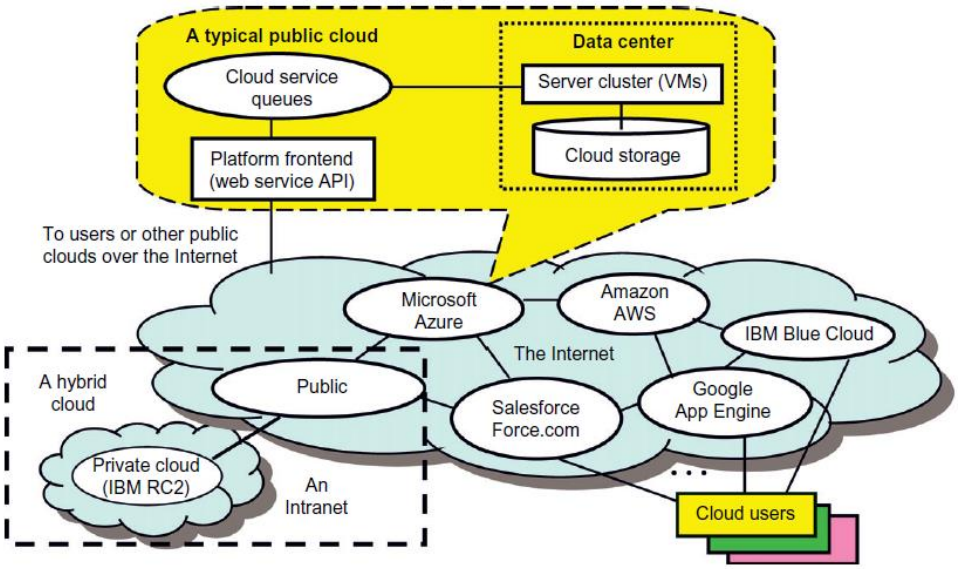


FIGURE 1 Public, private, and hybrid clouds illustrated by functional architecture and connectivity of representative clouds available.

As Figure 1 shows, both public clouds and private clouds are developed in the Internet. As many clouds are generated by commercial providers or by enterprises in a distributed manner, they will be interconnected over the Internet to achieve scalable and efficient computing services. Commercial cloud providers such as Amazon, Google, and Microsoft created their platforms to be distributed geographically.

i) Public Clouds

A public cloud is built over the Internet and can be accessed by any user who has paid for the service. Public clouds are owned by service providers and are accessible through a subscription. The callout box in top of Figure 1 shows the architecture of a typical public cloud. Many public clouds are available, including Google App Engine (GAE), Amazon Web Services (AWS), Microsoft Azure, IBM Blue Cloud, and Salesforce.com's Force.com. The providers of the aforementioned clouds are commercial providers that offer a publicly accessible remote interface for creating and managing VM instances within their proprietary infrastructure. A public cloud delivers a selected set of business processes. The application and infrastructure services are offered on a flexible price-per-use basis.

2) **Private Clouds**

A private cloud is built within the domain of an intranet owned by a single organization. Therefore, it is client owned and managed, and its access is limited to the owning clients and their partners. Its deployment was not meant to sell capacity over the Internet through publicly accessible interfaces. Private clouds give local users a flexible and agile private infrastructure to run service workloads within their administrative domains. A private cloud is supposed to deliver more efficient and convenient cloud services. It may impact the cloud tandardization, while retaining greater customization and organizational control.

3) **Hybrid Clouds**

A hybrid cloud is built with both public and private clouds, as shown at the lower-left corner of Figure 1. Private clouds can also support a hybrid cloud model by supplementing local infrastructure with computing capacity from an external public cloud. For example, the Research Compute Cloud (RC2) is a private cloud, built by IBM, that

interconnects the computing and IT resources at eight IBM Research Centers scattered throughout the United States, Europe, and Asia.

A hybrid cloud provides access to clients, the partner network, and third parties. In summary, public clouds promote standardization, preserve capital investment, and offer application flexibility. Private clouds attempt to achieve customization and offer higher efficiency, resiliency, security, and privacy. Hybrid clouds operate in the middle, with many compromises in terms of resource sharing.

4) Community Cloud: (**Write short notes on Community Cloud?)**

A community cloud falls between public and private clouds with respect to the target set of consumers. It is somewhat similar to a private cloud, but the infrastructure and computational resources are exclusive to two or more organizations that have common privacy, security, and regulatory considerations, rather than a single organization.

The community cloud aspires to combine distributed resource provision from grid computing, distributed control from digital ecosystems and sustainability from green computing, with the use cases of cloud computing, while making greater use of self-management advances from autonomic computing.

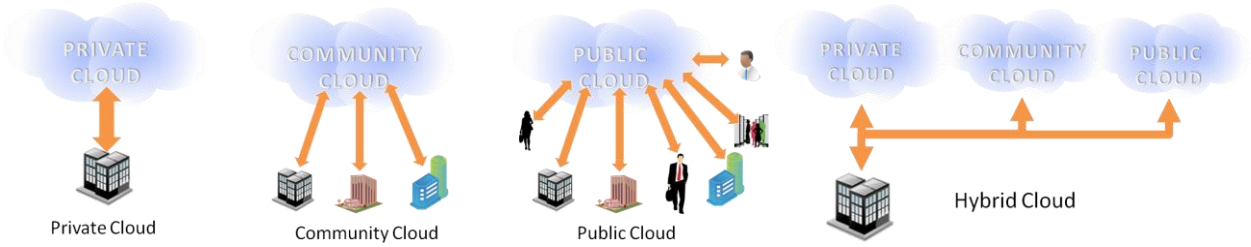
Replacing vendor clouds by shaping the underutilized resources of user machines to form a community cloud, with nodes potentially fulfilling all roles, consumer, producer, and most importantly coordinator.

The **advantages** of community cloud include:

- Cost of setting up a communal cloud versusϖ individual private cloud can be cheaper due to the division of costs among all participants.
- Management of the community cloud can be outsourced to a cloud provider. The advantage here is that the provider would be an impartial third party that is bound by contract and that has no preference to any of the clients involved other than what is contractually mandated.
- Tools residing in the community cloud can be used to leverage the information stored to serve consumers and the supply chain, such as return tracking and just-in-time production and distribution.

Drawbacks of community cloud:

- Costs higher than public cloud.
- Fixed amount of bandwidth and data storage is shared among all community members.



2) Explain in detail the Cloud Ecosystem and Enabling Technologies?
Why is cloud called as ecosystem? Justify (may/june 2014)

Cloud computing platforms differ from conventional computing platforms in many aspects. We will identify their differences in computing paradigms and cost models applied.

The traditional computing model is specified below by the process on the left, which involves buying the hardware, acquiring the necessary system software, installing the system, testing the configuration, and executing the application code and management of resources. What is even worse is that this cycle repeats itself in about every 18 months, meaning the machine we bought becomes obsolete every 18 months.

The cloud computing paradigm is shown on the right. This computing model follows a **pay-as-you-go model**. Therefore the cost is significantly reduced, because we simply rent computer resources without buying the computer in advance. All hardware and software resources are leased from the cloud provider without capital investment on the part of the users. Only the execution phase costs some money.

The experts at IBM have estimated that an 80 percent to 95 percent saving results from cloud computing compared with the conventional computing paradigm. This is very much desired, especially for small businesses, which requires limited computing power and thus avoid the purchase of expensive computers or servers repeatedly every few years.

Classical Computing	**Cloud Computing**
(Repeat the following cycle every 18 months)	(Pay as you go per each service provided)
Buy and own	**Subscribe**
Hardware, system software, applications to meet peak needs	- - - -
Install, configure, test, verify, evaluate, manage	**Use** (Save about 80-95% of the total cost)
- - - -	- - - -
Use	(Finally)
- - - -	**$ - Pay for what you use**
Pay $$$$$ (High cost)	based on the QoS

For example, IBM has estimated that the worldwide cloud service market may reach $126 billion by 2012, including components, infrastructure services, and business services. Internet clouds work as service factories built around multiple data centers. To formalize the above cloud computing model, we characterize the cloud cost model, the cloud ecosystems, and enabling technologies. These topics help our readers understand the motivations behind cloud computing. The intention is to remove the barriers of cloud computing

Cloud Design Objectives

Despite the controversy surrounding the replacement of desktop or deskside computing by centralized computing and storage services at data centers or big IT companies, the cloud computing community has reached some consensus on what has to be done to make cloud computing universally acceptable.
The following list highlights six design objectives for cloud computing:

• **Shifting computing from desktops to data centers** Computer processing, storage, and software delivery is shifted away from desktops and local servers and toward data centers over the Internet.

• **Service provisioning and cloud economics** Providers supply cloud services by signing SLAs with consumers and end users. The services must be efficient in terms of computing, storage, and power consumption. Pricing is based on a pay-as-you-go policy.

• **Scalability in performance** The cloud platforms and software and infrastructure services must be able to scale in performance as the number of users increases.

• **Data privacy protection** Can you trust data centers to handle your private data and records? This concern must be addressed to make clouds successful as trusted services.

• **High quality of cloud services** The QoS of cloud computing must be standardized to make clouds interoperable among multiple providers.

• **New standards and interfaces** This refers to solving the data lock-in problem associated with data centers or cloud providers. Universally accepted APIs and access protocols are needed to provide high portability and flexibility of virtualized applications.

Cost Model:

In traditional IT computing, users must acquire their own computer and peripheral equipment as capital expenses. In addition, they have to face operational expenditures in operating and maintaining the computer systems, including personnel and service costs.

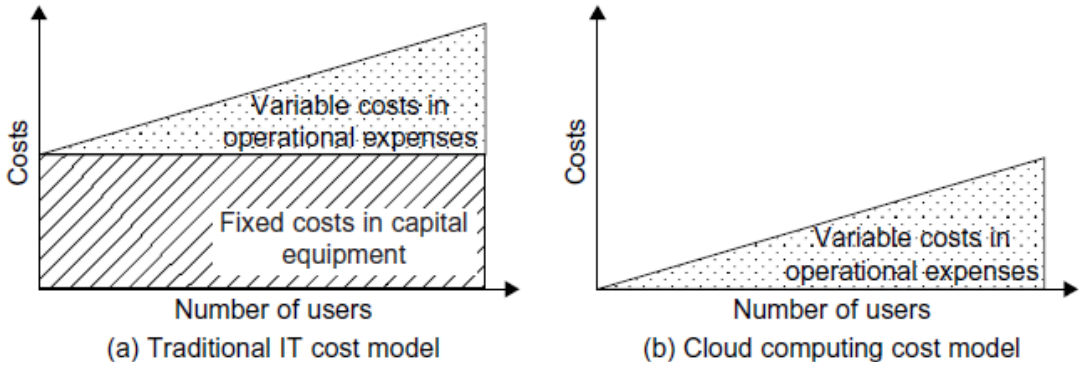


FIGURE: Computing economics between traditional IT users and cloud users

Figure **(a)** shows the addition of variable operational costs on top of fixed capital investments in traditional IT. Note that the fixed cost is the main cost, and that it could be reduced slightly as the number of users increases. However, the operational costs may increase sharply with a larger number of users. Therefore, the total cost escalates quickly with massive numbers of users. On the other hand, cloud computing applies a pay-per-use business model, in which user jobs are outsourced to data centers. To use the cloud, one has no up-front cost in hardware acquisitions.

Only variable costs are experienced by cloud users, as demonstrated in Figure 4.3**(b).** Overall, cloud computing will reduce computing costs significantly for both small users and large enterprises. Computing economics does show a big gap between traditional IT users and cloud users. The savings in acquiring expensive computers up front releases a lot of burden for startup companies. The fact that cloud users only pay for operational expenses and do not have to invest in permanent equipment is especially attractive to massive numbers of small users.

This is a major driving force for cloud computing to become appealing to most enterprises and heavy computer users. In fact, any IT users whose capital expenses are under more pressure than their operational expenses should consider sending their overflow work to utility computing or cloud service
Providers.

Cloud Ecosystems

An ecosystem was suggested by Sotomayor, et al. (Figure 4) for building private clouds.
They suggested four levels of ecosystem development in a private cloud. At the user end, consumers demand a flexible platform. At the cloud management level, the cloud manager provides virtualized resources over an IaaS platform. At the virtual infrastructure (VI) management level, the manager allocates VMs over multiple server clusters. Finally, at the VM management level, the VM managers handle VMs installed on individual host machines.

An ecosystem of cloud tools attempts to span both cloud management and VI management. Integrating these two layers is complicated by the lack of open and standard interfaces between them.

An increasing number of startup companies are now basing their IT strategies on cloud resources, spending little or no capital to manage their own IT infrastructures. We desire a flexible and open architecture that enables organizations to build private/hybrid clouds. VI management is aimed at this goal.

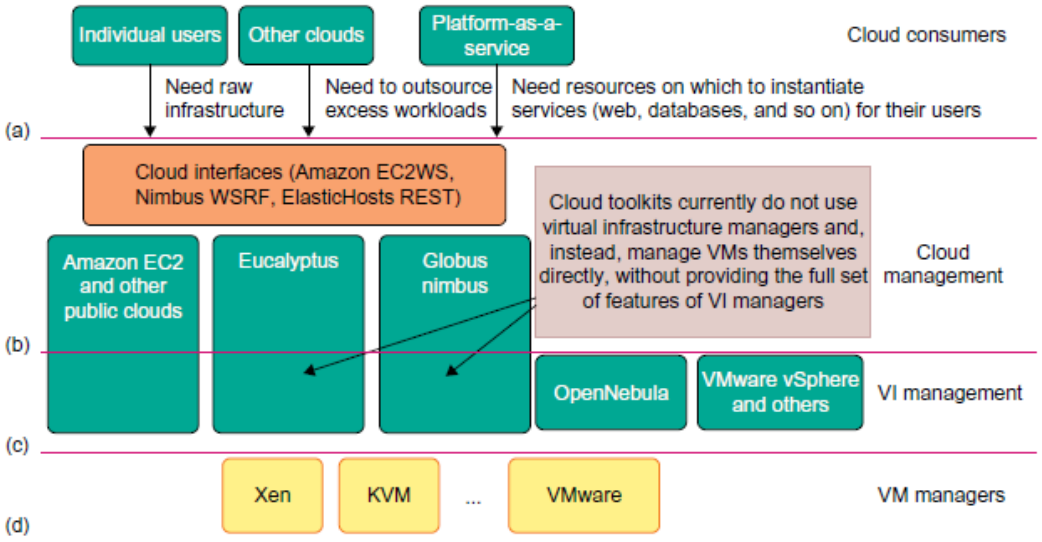


FIGURE 4 : Cloud ecosystem for building private clouds: (a) Consumers demand a flexible platform; (b) Cloud manager provides virtualized resources over an IaaS platform; (c) VI manager allocates VMs; (d) VM managers handle VMs installed on servers.

Example VI tools include oVirt, vSphere/4 from VMWare, and VM Orchestrator from Platfom Computing. These tools support dynamic placement and VM management on a pool of physical resources, automatic load balancing, server consolidation, and dynamic infrastructure resizing and partitioning. In addition to public clouds such as Amazon EC2, Eucalyptus and Globus Nimbus are open source tools for virtualization of cloud infrastructure. To access these cloud management tools, one can use the Amazon EC2WS, Nimbus WSRF, and ElasticHost REST cloud interfaces. For VI management, OpenNebula and VMware vSphere can be used to manage all VM generation including Xen, KVM, and VMware tools.

3) Enlist and explain the various services and deployment method of cloud computing environment ?(jan 2015) ? Explain the Categories of cloud computing?

Cloud computing delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. The services provided over the cloud can be generally categorized into three different service models: namely IaaS, Platform as a Service (PaaS), and Software as a Service (SaaS). These form the three pillars on top of which cloud computing solutions are delivered to end users. All three models allow users to access services over the Internet, relying entirely on the infrastructures of cloud service providers.

These models are offered based on various SLAs between providers and users. In a broad sense, the SLA for cloud computing is addressed in terms of service availability, performance, and data protection and security. Figure 4.5 illustrates three cloud models at different service levels of the cloud.

SaaS is applied at the application end using special interfaces by users or clients.

PaaS layer, the cloud platform must perform billing services and handle job queuing, launching, and monitoring services.

IaaS services, databases, compute instances, the file system, and storage must be provisioned to satisfy user demands.

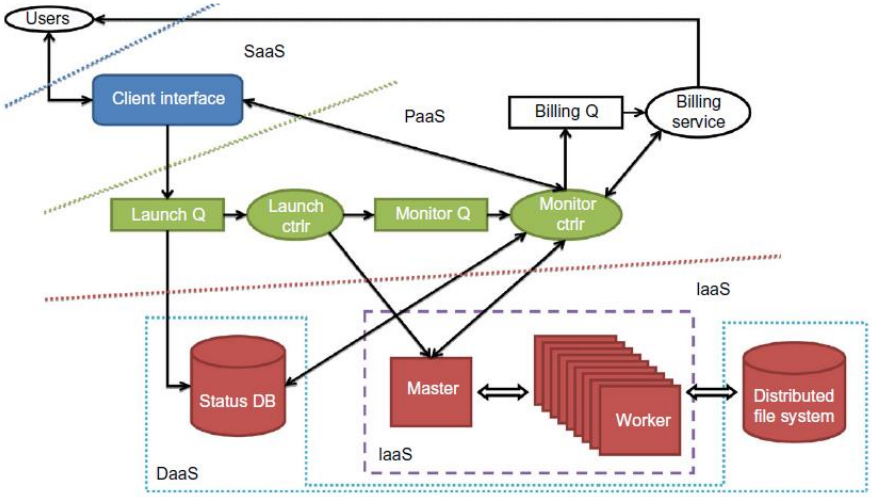


FIGURE 4.5 The IaaS, PaaS, and SaaS cloud service models at different service levels..

Infrastructure-as-a-Service (IaaS)

This model allows users to use virtualized IT resources for computing, storage, and networking. In short, the service is performed by rented cloud infrastructure. The user can deploy and run his applications over his chosen OS environment. The user does not manage or control the underlying cloud infrastructure, but has control over the OS, storage, deployed applications, and possibly select networking components. This IaaS model encompasses storage as a service, compute instances as a service, and communication as a service.

The Virtual Private Cloud (VPC) in Example 4.1 show to provide Amazon EC2 clusters and S3 storage to multiple users. Many startup cloud providers have appeared in recent years. GoGrid, FlexiScale, and Aneka are good examples. Table 4.1summarizes the IaaS offerings by five public cloud providers. Interested readers can visit the companies' web sites for updated information.

Example 4.1 Amazon VPC for Multiple Tenants

A user can use a private facility for basic computations. When he must meet a specific workload requirement, he can use the Amazon VPC to provide additional EC2 instances or more storage (S3) to handle urgent applications. Figure 4.6 shows VPC which is essentially a private cloud designed to address the privacy concerns of public clouds that hamper their application when sensitive data and software are involved. Amazon EC2 provides the following services: resources from multiple data centers globally distributed, CL1, web services (SOAP and Query), web-based console user interfaces, access to VM instances via

Table 4.1 Public Cloud Offerings of IaaS [10,18]

Cloud Name	VM Instance Capacity	API and Access Tools	Hypervisor, Guest OS
Amazon EC2	Each instance has 1–20 EC2 processors, 1.7–15 GB of memory, and 160–1.69 TB of storage.	CLI or web Service (WS) portal	Xen, Linux, Windows
GoGrid	Each instance has 1–6 CPUs, 0.5–8 GB of memory, and 30–480 GB of storage.	REST, Java, PHP, Python, Ruby	Xen, Linux, Windows
Rackspace Cloud	Each instance has a four-core CPU, 0.25–16 GB of memory, and 10–620 GB of storage.	REST, Python, PHP, Java, C#, .NET	Xen, Linux
FlexiScale in the UK	Each instance has 1–4 CPUs, 0.5–16 GB of memory, and 20–270 GB of storage.	web console	Xen, Linux, Windows
Joyent Cloud	Each instance has up to eight CPUs, 0.25–32 GB of memory, and 30–480 GB of storage.	No specific API, SSH, Virtual/Min	OS-level virtualization, OpenSolaris

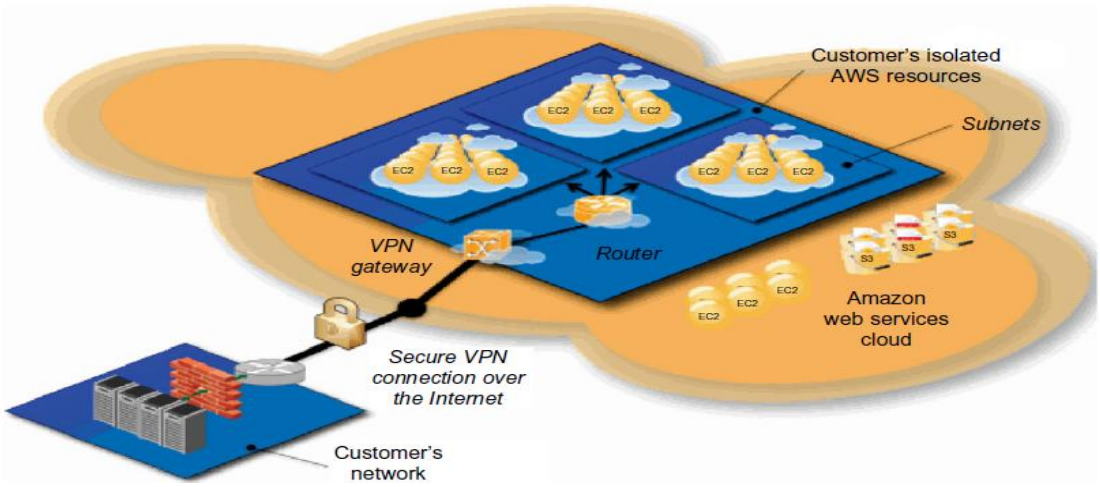


FIGURE 4.6: Amazon VPC (virtual private cloud).

SSH and Windows, 99.5 percent available agreements, per-hour pricing, Linux and Windows OSes, and automatic scaling and load balancing. VPC allows the user to isolate provisioned AWS processors, memory, and storage from interference by other users. Both auto-scaling and elastic load balancing services can support related demands. Auto-scaling enables users to automatically scale their VM instance capacity up or down. With auto-scaling, one can ensure that a sufficient number of Amazon EC2 instances are provisioned to meet desired Performance. Or one can scale down the VM instance capacity to reduce costs, when the workload is reduced.

Platform as a Service (PaaS)

To be able to develop, deploy, and manage the execution of applications using provisioned resources demands a cloud platform with the proper software environment. Such a platform includes operating system and runtime library support. This has triggered the creation of the PaaS model to enable users to develop and deploy their user applications. Table 4.2 highlights cloud platform services offered by five PaaS services.

Table 4.2 Five Public Cloud Offerings of PaaS [10,18]

Cloud Name	Languages and Developer Tools	Programming Models Supported by Provider	Target Applications and Storage Option
Google App Engine	Python, Java, and Eclipse-based IDE	MapReduce, web programming on demand	Web applications and BigTable storage
Salesforce.com's Force.com	Apex, Eclipse-based IDE, web-based Wizard	Workflow, Excel-like formula, Web programming on demand	Business applications such as CRM
Microsoft Azure	.NET, Azure tools for MS Visual Studio	Unrestricted model	Enterprise and web applications
Amazon Elastic MapReduce	Hive, Pig, Cascading, Java, Ruby, Perl, Python, PHP, R, C++	MapReduce	Data processing and e-commerce
Aneka	.NET, stand-alone SDK	Threads, task, MapReduce	.NET enterprise applications, HPC

The platform cloud is an integrated computer system consisting of both hardware and software infrastructure. The user application can be developed on this virtualized cloud platform using some programming languages and software tools supported by the provider (e.g., Java, Python, .NET). The user does not manage the underlying cloud infrastructure. The cloud provider supports user application development and testing on a well-defined service platform. This PaaS model enables a collaborated software development platform for users from different parts of the world. This model also encourages third parties to provide software management, integration, and service monitoring solutions.

Example 4.2 Google App Engine for PaaS Applications

As web applications are running on Google's server clusters, they share the same capability with many other users. The applications have features such as automatic scaling and load balancing which are very convenient while building web applications.

The distributed scheduler mechanism can also schedule tasks for triggering events at specified times and regular intervals. Figure 4.7 shows the operational model for GAE. To develop applications using GAE, a development environment must be provided. Google provides a fully featured local development environment that simulates GAE on the developer's computer. All the functions and application logic can be implemented locally which is quite similar to traditional software development.

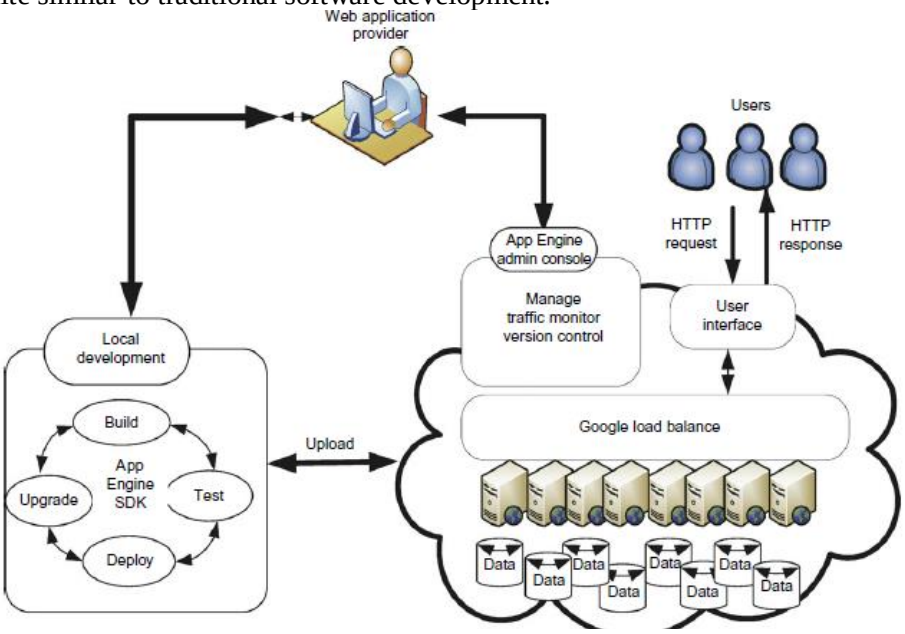


FIGURE 4.7 Google App Engine platform for PaaS operations

The coding and debugging stages can be performed locally as well. After these steps are finished, the SDK provided provides a tool for uploading the user's application to Google's infrastructure where the applications are actually deployed. Many additional third-party capabilities, including software management, integration, and service monitoring solutions, are also provided.

Software as a Service (SaaS)

The SaaS model provides software applications as a service. As a result, on the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are kept rather low, compared with conventional hosting of user applications. Customer data is stored in the cloud that is either vendor proprietary or publicly hosted to support PaaS and IaaS. The best examples of SaaS services include Google Gmail and docs, Microsoft SharePoint, and the CRM software from Salesforce.com. They are all very successful in promoting their own business or are used by thousands of small businesses in their day-to-day operations.

Providers such as Google and Microsoft offer integrated IaaS and PaaS services, where as others such as Amazon and Go Grid offer pure IaaS services and expect third-party PaaS providers such as Manjra soft to offer application development and deployment services on top of their infrastructure services.

To identify important cloud applications in enterprises, the success stories of three real-life cloud applications are presented in Example 4.3 for HTC, news media, and business transactions. The benefits of using cloud services are evident in these SaaS applications.

Example 4.3 Three Success Stories on SaaS Applications

1. To discover new drugs through DNA sequence analysis, Eli Lily Company has used Amazon's AWS platform with provisioned server and storage clusters to conduct high-performance biological sequence analysis without using an expensive supercomputer. The benefit of this IaaS application is reduced drug deployment time with much lower costs.
2. The New York Times has applied Amazon's EC2 and S3 services to retrieve useful pictorial information quickly from millions of archival articles and newspapers. The New York Times has significantly reduced the time and cost in getting the job done.
3. Pitney Bowes, an e-commerce company, offers clients the opportunity to perform B2B transactions using the Microsoft Azure platform, along with .NET and SQL services. These offerings have significantly increased the company's client base.

4). Discuss about the Pros and Cons of cloud computing? (May/June 2014)

The following list outlines the advantages and drawbacks of cloud computing:

Advantages of Cloud Computing

1) **Low Cost** -- Entering the cloud is a low-cost proposition. It does not require a large capital investment up front for hardware, equipment and infrastructure. You will need to have desktops, laptops or some type of device to access the Internet and utilize your data. But, the investment in implementing and maintaining your own in-house network is minimal. For many small businesses, this is important. The cost of maintaining an in-house network is enormous and seems to be increasing every day. With an in-house network, there are the costs of software and hardware upgrades as well as maintaining and training of an IT staff. Because these responsibilities are with the cloud services provider, those costs are theirs. The best way to think of the cost of cloud computing is to think in terms of renting vs. owning. The cloud provider owns and maintains all of the resources and the business client begins to use it for a monthly or annual fee. This is similar to what is commonly referred to as the Software-as-a-Service (or SaaS) model. You pay as you go. This type of model gives the business owner more predictability in budgeting for these costs.

2) **Flexibility** -- With cloud computing, because you only pay for what you use, you have the flexibility to only use what you need. This means if you are a small business in a start-up mode, you can start small. As the business grows, you simply use more computing resources as needed. It can be very fast to set up, implement, and bring into operation. Some cloud providers are set up to automatically scale for your resource demands. The

infrastructure can usually be customized to your needs. It can be a private network, public, or a combination of both. In addition, the cloud supports multi-platform development environments.

3) **Simple, Fast, Easy** -- The beauty of cloud computing is that it's easy. Using cloud computing can streamline many parts of your business. Your business can run more efficiently when you tap into web-based applications that are available in the cloud. Everything from prospect management applications to customer billing and invoicing can be moved out of your shop and into the cloud. This gives you the ability to focus on what you do best in your business and excel at your strengths while someone else handles the administrative functions.

4) **Accessibility** -- Regardless of where you are in the world, you can access your cloud based applications. Gone are the complicated remote login procedures required for your in-house network. The only thing required is a device that can access the web and an Internet connection. This means your staff can have access anywhere and at anytime, from home, office or on the road at a client's office.

5) **Sustainability** -- Should a natural disaster strike your business, the good news is that your computing capability resides somewhere else. Obviously, this is a disadvantage if the disaster hits your cloud provider

Potential Disadvantages of Cloud Computing

1) **Security of Your Data** -- This is one of the primary concerns related to cloud computing. In a very basic sense, the data that used to reside within the four walls of your facility now resides elsewhere. The security of that data must be addressed, particularly if the data contains trade secrets, proprietary lists, customer files, etc. Adding to the concern are the results from a recent survey: 69% of cloud providers said that data security was the responsibility of the end-user. By contrast, only 35% of the end-users agreed that they should be responsible. It is clear that there is a disparity between cloud providers and cloud users about who is responsible for data security.

2) **Redundancy** -- This term refers to the reliability of your web-based applications that run in the cloud. A practical example would be if the server that your website is running on crashes, another server picks up where the other left off and your business keeps going. This is redundancy. In the cloud many think that because their applications are running "out there" on the Internet that there is built-in redundancy. This is a misconception and not true. The typical cloud provider will have resources running on one server. Some will have multiple servers and this is good; however, the truth is that those multiple servers typically reside in one data center. If there is a problem at the data center that takes down multiple servers, or the entire data center, then your business can come down with it. This has happened. There are several recent examples of major entities being down for anywhere from an hour to several days over the past two years. These have included Amazon, Microsoft SideKick, Google, Hotmail, Intuit and others.

3) **Costs When Under Attack** -- Even though cloud computing offers a great low cost option for small businesses, it can actually cost more money if a company's website comes under a distributed denial of service (DDoS) attack. The reason is that the typical goal of a DDoS attack is to gobble up resources in order to render the server incapacitated. If the cloud computing provider has no protections in place against DDoS attacks, when the site is attacked, the provider will simply begin to increase the resources that the site requires (due to the attack) and bill for the resources provided. It is important to ask the provider what kind of provisions they have to protect against this type of attack.

4) **Performance Can Vary** -- In a cloud environment, your applications are running on servers that simultaneously provide resources to other businesses. As the requirements for the other users go up and down, the performance of your share of the resources will vary. Often, a cloud provider may claim that the resources available to you are unlimited. This may be theoretically true, but from a practical point of view the hardware scalability is probably limited. You may not know exactly how scalable it is until you reach your usage limitation on their system

5) Explain the characteristics and types of virtualization in cloud computing ?(may/june 2014)
Explain with a neat diagram the five abstraction levels of virtualization?

Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine. The idea of VMs can be dated back to the 1960s. The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility. Hardware resources (CPU, memory, I/O devices, etc.) or software resources (operating system and software libraries) can be virtualized in various functional layers.

Levels of Virtualization Implementation

A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in Figure 3.1(a). After virtualization, different user applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS. This is often done by adding additional software, called a virtualization layer as shown in Figure 3.1(b). This virtualization layer is known as hypervisor or virtual machine monitor (VMM). The VMs are shown in the upper boxes, where applications run with their own guest OS over the virtualized CPU, memory, and I/O resources.

The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs, exclusively. This can be implemented at various operational levels, as we will discuss shortly. The virtualization software creates the abstraction of VMs by interposing a virtualization layer at various levels of a computer system.

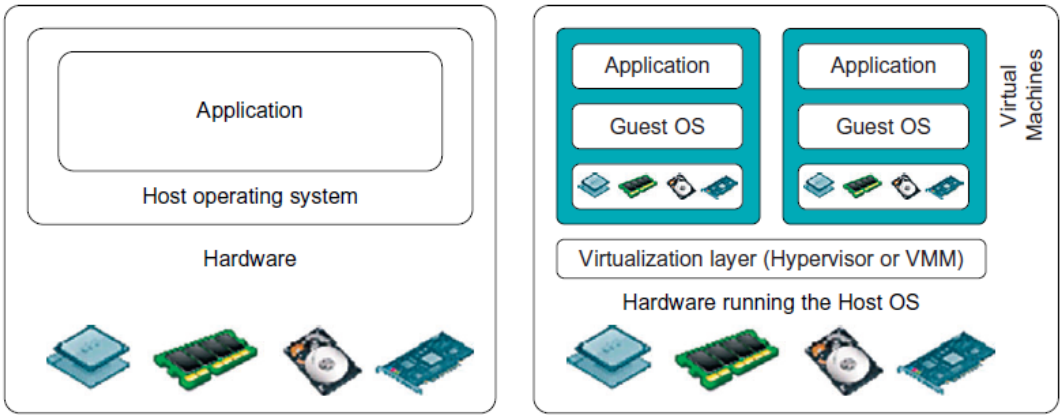


FIGURE 3.1 The architecture of a computer system before and after virtualization, where VMM stands for virtual machine monitor.

Common virtualization layers include

- Instruction set architecture (ISA) level,
- Hardware abstraction level,
- Operating system level,
- Library support level
- Application level

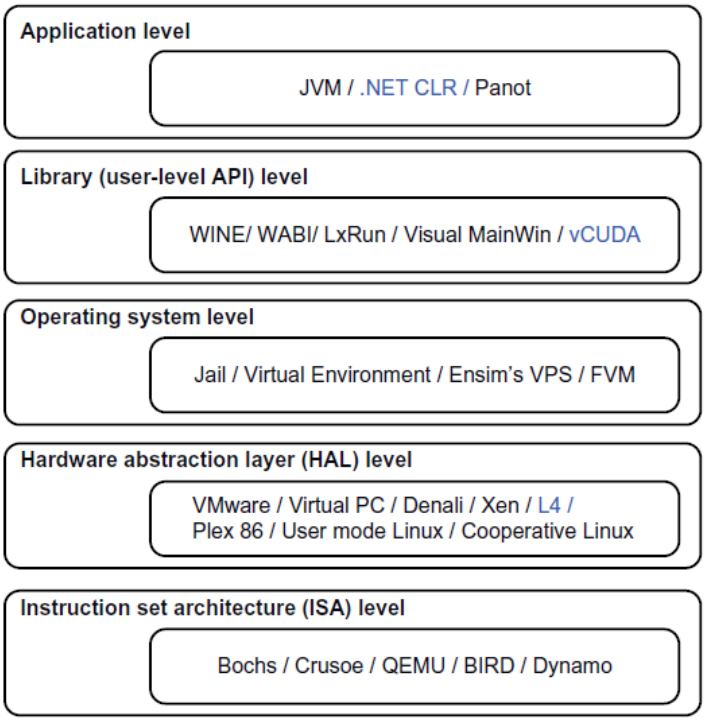


FIGURE 3.2 Virtualization ranging from hardware to applications in five abstraction levels

Instruction Set Architecture Level (ISA):

At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. For example, MIPS binary code can run on an x86-based host machine with the help of ISA emulation. With this approach, it is possible to run a large amount of legacy binary code written for various processors on any given new hardware host machine. Instruction set emulation leads to virtual ISAs created on any hardware machine.

The basic emulation method is through code interpretation. An interpreter program interprets the source instructions to target instructions one by one. One source instruction may require tens or hundreds of native target instructions to perform its function. Obviously, this process is relatively slow. For better performance, dynamic binary translation is desired.

This approach translates basic blocks of dynamic source instructions to target instructions. The basic blocks can also be extended to program traces or super blocks to increase translation efficiency. Instruction set emulation requires binary translation and optimization. A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.

Hardware Abstraction Level

Hardware-level virtualization is performed right on top of the bare hardware. On the one hand, this approach generates a virtual hardware environment for a VM. On the other hand, the process manages the underlying hardware through virtualization. The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices. The intention is to upgrade the hardware utilization rate by multiple users concurrently. The idea was implemented in the IBM VM/370 in the 1960s. More recently, the Xen hypervisor has been applied to virtualize x86-based machines to run Linux or other guest OS applications.

Operating System Level

OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hardware and software in data centers. The containers behave like real servers. OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users. It is also used, to a lesser extent, in consolidating server hardware by moving services on separate hosts into containers or VMs on one server.

Library Support Level

Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS. Since most systems provide well-documented APIs, such an interface becomes another candidate for virtualization. Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks. The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts. Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.

User-Application Level

Virtualization at the application level virtualizes an application as a VM. On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process-level virtualization. The most popular approach is to deploy ***high level language (HLL)*** VMs. The virtualization layer sits as an application program on top of the operating system, and the layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition. Any program written in the HLL and compiled for this VM will be able to run on it.

The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM. Other forms of application-level virtualization are known as application isolation, application sandboxing, or application streaming. The process involves wrapping the application in a layer that is isolated from the host OS and other applications. The result is an application that is much easier to distribute and remove from user workstations.

6) Explain in detail the virtualization and Structures /tools and Mechanisms?

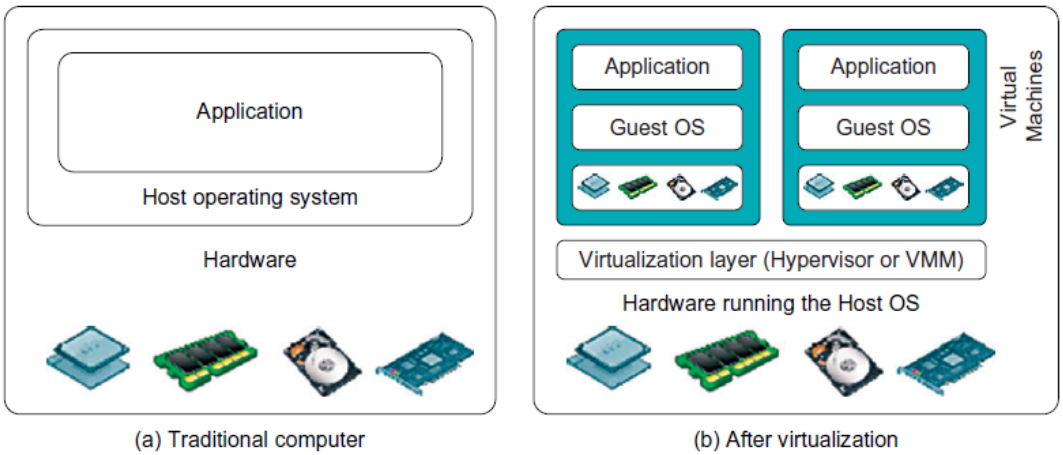


FIGURE 3.1 The architecture of a computer system before and after virtualization, where VMM stands for virtual machine monitor.

In general, there are three typical classes of VM architecture. Figure 3.1 showed the architectures of a machine before and after virtualization. Before virtualization, the operating system manages the hardware. After virtualization, a virtualization layer is inserted between the hardware and the operating system. In such a case, the virtualization layer is responsible for converting portions of the real hardware into virtual hardware. Therefore, different operating systems such as Linux and Windows can run on the same physical machine, simultaneously. Depending on the position of the virtualization layer, there are several classes of VM architectures, namely the hypervisor architecture, paravirtualization and host-based virtualization. The hypervisor is also known as the VMM (Virtual Machine Monitor). They both perform the same virtualization operations.

Hypervisor and Xen Architecture: (Write short notes on Hypervisor and Xen Architecture?)

The hypervisor supports hardware-level virtualization (see Figure 3.1(b)) on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physical hardware and its

OS. This virtualization layer is referred to as either the VMM or the hypervisor. The hypervisor provides hypercalls for the guest OSes and applications. Depending on the functionality, a hypervisor can assume micro-kernel architecture like the Microsoft Hyper-V. Or it can assume monolithic hypervisor architecture like the VMware ESX for server virtualization.

A ***micro-kernel hypervisor*** includes only the basic and unchanging functions (such as physical memory management and processor scheduling). The device drivers and other changeable components are outside the hypervisor. A monolithic hypervisor implements all the aforementioned functions, including those of the device drivers. Therefore, the size of the hypervisor code of a micro-kernel hypervisor is smaller than that of a monolithic hypervisor. Essentially, a hypervisor must be able to convert physical devices into virtual resources dedicated for the deployed VM to use.

1. The Xen Architecture

Xen is an open source hypervisor program developed by Cambridge University. Xen is a microkernel hypervisor, which separates the policy from the mechanism. The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0, as shown in Figure 3.5. Xen does not include any device drivers natively . It just provides a mechanism by which a guest OS can have direct access to the physical devices. As a result, the size of the Xen hypervisor is kept rather small. Xen provides a virtual environment located between the hardware and the OS.

A number of vendors are in the process of developing commercial Xen hypervisors, among them are Citrix XenServer and Oracle VM .The core components of a Xen system are the hypervisor, kernel, and applications. The organization of the three components is important. Like other virtualization systems, many guest OSes can run on top of the hypervisor. However, not all guest OSes are created equal, and one in particular controls the others.

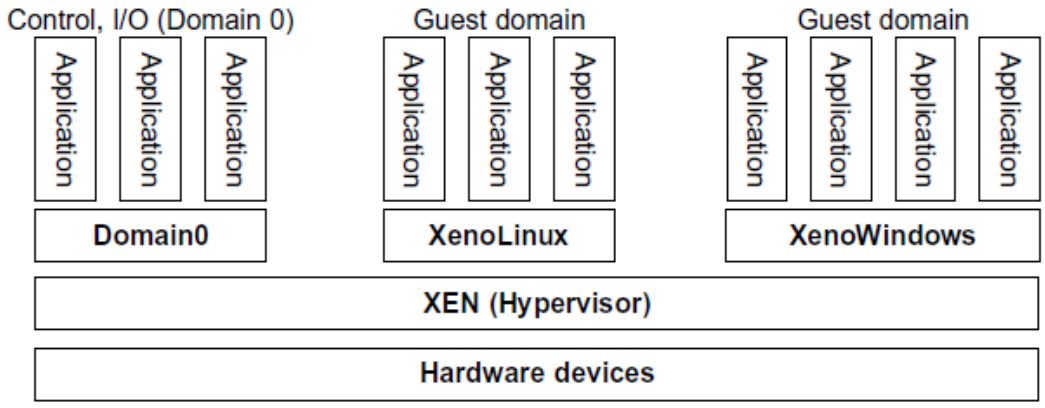


FIGURE 3.5 The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications.

The guest OS, which has control ability, is called Domain 0, and the others are called Domain U. Domain 0 is a privileged guest OS of Xen. It is first loaded when Xen boots without any file system drivers being available. Domain 0 is designed to access hardware directly and manage devices. Therefore, one of the responsibilities of Domain 0 is to allocate and map hardware resources for the guest domains (the Domain U domains).

For example, Xen is based on Linux and its security level is C2. Its management VM is named Domain 0, which has the privilege to manage other VMs implemented on the same host. If Domain 0 is compromised, the hacker can control the entire system. So, in the VM system, security policies are needed to improve the security of Domain 0. Domain 0, behaving as a VMM, allows users to create, copy, save, read, modify, share, migrate, and roll back VMs as easily as manipulating a file, which flexibly provides tremendous benefits for users. Unfortunately, it also brings a series of security problems during the software life cycle and data lifetime.

Traditionally, a machine's lifetime can be envisioned as a straight line where the current state of the machine is a point that progresses monotonically as the software executes. During this time, configuration changes are made, software is installed, and patches are applied.

In such an environment, the VM state is akin to a tree: At any point, execution can go into N different branches where multiple instances of a VM can exist at any point in this tree at any given time. VMs are allowed to roll back to previous states in their execution (e.g., to fix configuration errors) or rerun from the same point many times (e.g., as a means of distributing dynamic content or circulating a "live" system image).

2. **Binary Translation with Full Virtualization**
   (Write short notes on Binary Translation with Full Virtualization?)

Depending on implementation technologies, hardware virtualization can be classified into two categories:

- Full virtualization
- Host-based virtualization.

Full virtualization: does not need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive, nonvirtualizable instructions. The guest OSes and their applications consist of noncritical and critical instructions.

host-based system: both a host OS and a guest OS are used. A virtualization software layer is built between the host OS and guest OS. These two classes of VM architecture are introduced next.

***Full Virtualization*:**

With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software. Both the hypervisor and VMM approaches are considered full virtualization. Why are only critical instructions trapped into the VMM? This is because binary translation can incur a large performance overhead. Noncritical instructions do not control hardware or threaten the security of the system, but critical instructions do. Therefore, running noncritical instructions on hardware not only can promote efficiency, but also can ensure system security.

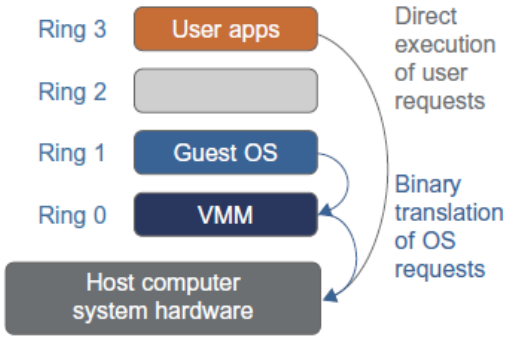


FIGURE 3.6

Indirect execution of complex instructions via binary translation of guest OS requests using the VMM plus direct execution of simple instructions on the same host.

Binary Translation of Guest OS Requests Using a VMM

This approach was implemented by VMware and many other software companies. As shown in Figure 3.6, VMware puts the VMM at Ring 0 and the guest OS at Ring 1. The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions. When these instructions are identified,

they are trapped into the VMM, which emulates the behavior of these instructions. The method used in this emulation is called binary translation. Therefore, full virtualization combines binary translation and direct execution. The guest OS is completely decoupled from the underlying hardware. Consequently, the guest OS is unaware that it is being virtualized. The performance of full virtualization may not be ideal, because it involves binary translation

Which is rather time-consuming . In particular, the full virtualization of I/O-intensive applications is a really a big challenge. Binary translation employs a code cache to store translated hot instructions to improve performance, but it increases the cost of memory usage. At the time of this writing, the performance of full virtualization on the x86 architecture is typically 80 percent to 97 percent that of the host machine.

3. **Host-Based Virtualization (Write short notes on Host-Based Virtualization?)**

An alternative VM architecture is to install a virtualization layer on top of the host OS. This host OS is still responsible for managing the hardware. The guest OSes are installed and run on top of the virtualization layer. Dedicated applications may run on the VMs. Certainly, some other applications can also run with the host OS directly.

This host-based architecture has some distinct advantages, as enumerated next.

First, the user can install this VM architecture without modifying the host OS. The virtualizing software can rely on the host OS to provide device drivers and other low-level services. This will simplify the VM design and ease its deployment.

Second, the host-based approach appeals to many host machine configurations. Compared to the hypervisor/VMM architecture, the performance of the host-based architecture may also be low.

When an application requests hardware access, it involves four layers of mapping which downgrades performance significantly. When the ISA of a guest OS is different from the ISA of the underlying hardware, binary translation must be adopted. Although the host-based architecture has flexibility, the performance is too low to be useful in practice.

4. Para-Virtualization with Compiler Support: (Write short notes on Para-Virtualization?)

A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications. Performance degradation is a critical issue of a virtualized system. No one wants to use a VM if it is much slower than using a physical machine. The virtualization layer can be inserted at different positions in a machine software stack. However, para-virtualization attempts to reduce the virtualization overhead, and thus improve performance by modifying only the guest OS kernel.

Figure 3.7 illustrates the concept of a para-virtualized VM architecture. The guest operating systems are para-virtualized. They are assisted by an intelligent compiler to replace the nonvirtualizable OS instructions by hypercalls as illustrated in Figure 3.8. The traditional x86 processor offers four instruction execution rings: Rings 0, 1, 2, and 3. The lower the ring number, the higher the privilege of instruction being executed. The OS is responsible for managing the hardware and the privileged instructions to execute at Ring 0, while user-level applications run at Ring 3. The best example of para-virtualization is the KVM to be described below.

4.1 Para-Virtualization Architecture

When the x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS. According to the x86 ring definition, the virtualization layer should also be installed at Ring 0. Different instructions at Ring 0 may cause some problems. In Figure 3.8, we show that para-virtualization replaces nonvirtualizable instructions with hypercalls that communicate directly with the hypervisor or VMM. However, when the guest OS kernel is modified for virtualization, it can no longer run on the hardware directly.

Although para-virtualization reduces the overhead, it has incurred other problems. ***First***, its compatibility and portability may be in doubt, because it must support the unmodified OS as well. ***Second***, the cost of maintaining para-virtualized OSes is high, because they may require deep OS kernel modifications. ***Finally***, the performance advantage of para-virtualization varies greatly due to workload variations

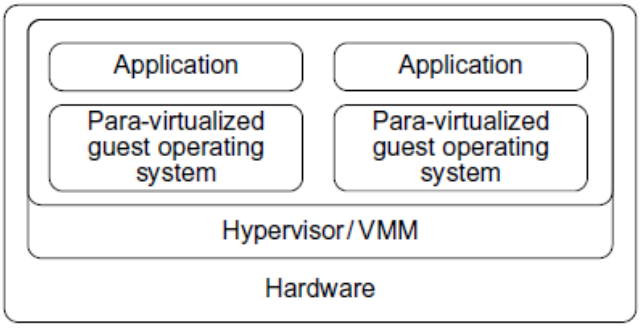


FIGURE 3.7

Para-virtualized VM architecture, which involves modifying the guest OS kernel to replace nonvirtualizable instructions with hypercalls for the hypervisor or the VMM to carry out the virtualization process (See Figure 3.8 for more details.)

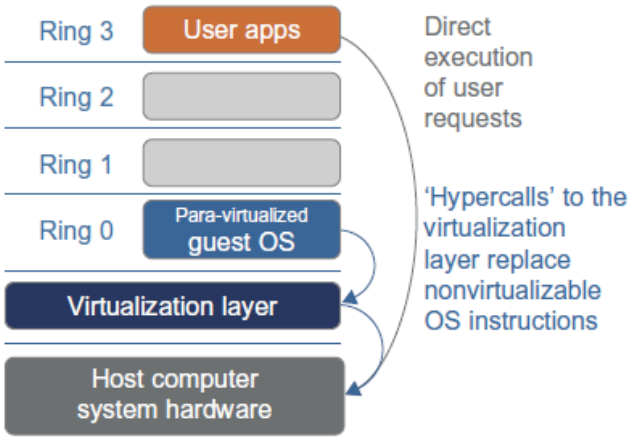


FIGURE 3.8

The use of a para-virtualized guest OS assisted by an intelligent compiler to replace nonvirtualizable OS instructions by hypercalls.

(*Courtesy of VMWare [71]*)

Compared with full virtualization, para-virtualization is relatively easy and more practical. The main problem in full virtualization is its low performance in binary translation. To speed up binary translation is difficult. Therefore, many virtualization products employ the para-virtualization architecture. The popular Xen, KVM, and VMware ESX are good examples.

4.2 KVM (Kernel-Based VM) (Write short notes on Kernel-Based VM with example?)

This is a Linux para-virtualization system—a part of the Linux version 2.6.20 kernel. Memory management and scheduling activities are carried out by the existing Linux kernel. The KVM does the rest, which makes it simpler than the hypervisor that controls the entire machine. KVM is a hardware-assisted para-virtualization tool, which improves performance and supports unmodified guest OSes such as Windows, Linux, Solaris, and other UNIX variants.

Example 3.3 VMware ESX Server for Para-Virtualization

VMware pioneered the software market for virtualization. The company has developed virtualization tools for desktop systems and servers as well as virtual infrastructure for large data centers.

ESX is a VMM or a hypervisor for bare-metal x86 symmetric multiprocessing (SMP) servers. It accesses hardware resources such as I/O directly and has complete resource management control.

An ESX-enabled server consists of four components:

1. Virtualization layer,
2. Resource manager,
3. Hardware interface components,
4. Service console as shown in Figure 3.9.

To improve performance, the ESX server employs a para-virtualization architecture in which the VM kernel interacts directly with the hardware without involving the host OS.

The VMM layer virtualizes the physical hardware resources such as CPU, memory, network and disk controllers, and human interface devices. Every VM has its own set of virtual hardware resources. The resource manager allocates CPU, memory disk, and network bandwidth and maps them to the virtual hardware resource set of each VM created. Hardware interface components are the device drivers and the

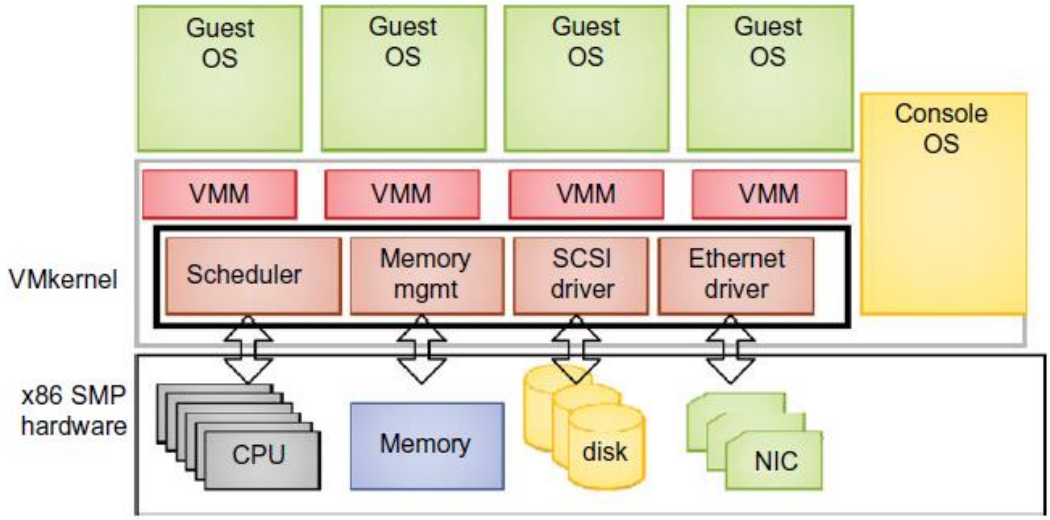


FIGURE 3.9 The VMware ESX server architecture using para-virtualization.

VMware ESX Server File System. The service console is responsible for booting the system, initiating the execution of the VMM and resource manager, and relinquishing control to those layers. It also facilitates the process for system administrators.

7. Explain in detail Virtualization of CPU, MEMORY, AND I/O DEVICES?

To support virtualization, processors such as the x86 employ a special running mode and instructions, known as hardware-assisted virtualization. In this way, the VMM and guest OS run in different modes and all sensitive instructions of the guest OS and its applications are trapped in the VMM. To save processor states, mode switching is completed by hardware. For the x86 architecture, Intel and AMD have proprietary technologies for hardware-assisted virtualization.

1. Hardware Support for Virtualization

Modern operating systems and processors permit multiple processes to run simultaneously. If there is no protection mechanism in a processor, all instructions from different processes will access the hardware directly and cause a system crash. Therefore, all processors have at least two modes, ***user mode*** and ***supervisor mode***, to ensure controlled access of critical hardware.

Instructions running in supervisor mode are called privileged instructions. Other instructions are unprivileged instructions. In a virtualized environment, it is more difficult to make OSes and applications run correctly because there are more layers in the machine stack.

The VMware Workstation is a VM software suite for x86 and x86-64 computers. This software suite allows users to set up multiple x86 and x86-64 virtual computers and to use one or more of these VMs simultaneously with the host operating system. The VMware Workstation assumes the host-based virtualization.

Xen is a hypervisor for use in IA-32, x86-64, Itanium, and PowerPC 970 hosts. Actually, Xen modifies Linux as the lowest and most privileged layer, or a hypervisor. One or more guest OS can run on top of the hypervisor.

KVM (Kernel-based Virtual Machine) is a Linux kernel virtualization infrastructure. KVM can support hardware-assisted virtualization and paravirtualization by using the Intel VT-x or AMD-v and VirtIO framework, respectively. The VirtIO framework includes a paravirtual Ethernet card, a disk I/O controller, a balloon device for adjusting guest memory usage, and a VGA graphics interface using VMware drivers

Example 3.4 Hardware Support for Virtualization in the Intel x86 Processor

Since software-based virtualization techniques are complicated and incur performance overhead, Intel provides a hardware-assist technique to make virtualization easy and improve performance. Figure 3.10 provides an overview of Intel's full virtualization techniques. For processor virtualization, Intel offers the VT-x or VT-i technique. VT-x adds a privileged mode (VMX Root Mode) and some instructions to processors. This enhancement traps all sensitive instructions in the VMM automatically. For memory virtualization, Intel offers the EPT, which translates the virtual address to the machine's physical addresses to improve performance. For I/O virtualization, Intel implements VT-d and VT-c to support this

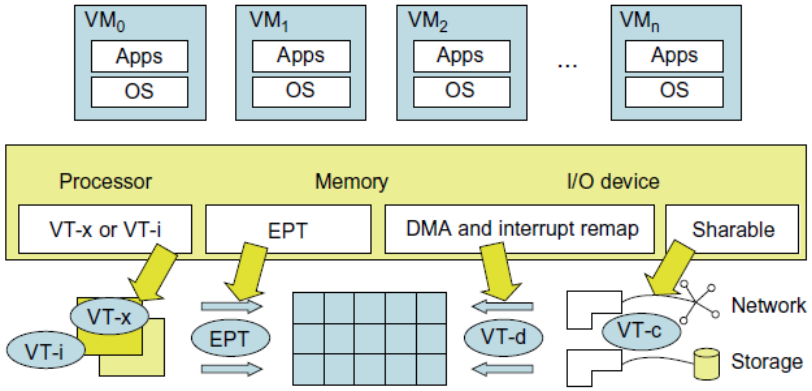


FIGURE 3.10 Intel hardware support for virtualization of processor, memory, and I/O devices

CPU Virtualization:

A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in native mode. Thus, unprivileged instructions of VMs run directly on the host machine for higher efficiency. Other critical instructions should be handled carefully for correctness and stability.

The critical instructions are divided into three categories:

1. **Privileged instructions**
   Privileged instructions execute in a privileged mode and will be trapped if executed outside this mode.
2. **Controlsensitive instructions**
   Control-sensitive instructions attempt to change the configuration of resources used
3. **Behavior-sensitive instructions.**
   Behavior-sensitive instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.

A CPU architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode. When the privileged instructions including control- and behavior-sensitive instructions of a VM are executed, they are trapped in the VMM.

The VMM acts as a unified mediator for hardware access from different VMs to guarantee the correctness and stability of the whole system. However, not all CPU architectures are virtualizable. RISC CPU architectures can be naturally virtualized because all control- and behavior-sensitive instructions are privileged instructions.

On the contrary, x86 CPU architectures are not primarily designed to support virtualization. This is because about 10 sensitive instructions, such as SGDT and SMSW, are not privileged instructions. When these instructions execute in virtualization, they cannot be trapped in the VMM.

On a native UNIX-like system, a system call triggers the 80h interrupt and passes control to the OS kernel. The interrupt handler in the kernel is then invoked to process the system call. On a paravirtualization

system such as Xen, a system call in the guest OS first triggers the 80h interrupt normally.Almost at the same time, the 82h interrupt in the hypervisor is triggered. Incidentally, control is passed on to the hypervisor as well.

When the hypervisor completes its task for the guest OS system call, it passes control back to the guest OS kernel. Certainly, the guest OS kernel may also invoke the hypercall while it's running. Although paravirtualization of a CPU lets unmodified applications run in the VM, it causes a small performance penalty.

1. Hardware-Assisted CPU Virtualization

This technique attempts to simplify virtualization because full or paravirtualization is complicated. Intel and AMD add an additional mode called privilege mode level (some people call it Ring-1) to x86 processors. Therefore, operating systems can still run at Ring 0 and the hypervisor can run at Ring -1. All the privileged and sensitive instructions are trapped in the hypervisor automatically. This technique removes the difficulty of implementing binary translation of full virtualization. It also lets the operating system run in VMs without modification.

Example 3.5 Intel Hardware-Assisted CPU Virtualization

Although x86 processors are not virtualizable primarily, great effort is taken to virtualize them. They are used widely in comparing RISC processors that the bulk of x86-based legacy systems cannot discard easily. Virtualization of x86 processors is detailed in the following sections. Intel's VT-x technology is an example of hardware-assisted virtualization, as shown in Figure 3.11. Intel calls the privilege level of x86 processors the VMX Root Mode. In order to control the start and stop of a VM and allocate a memory page to maintain the

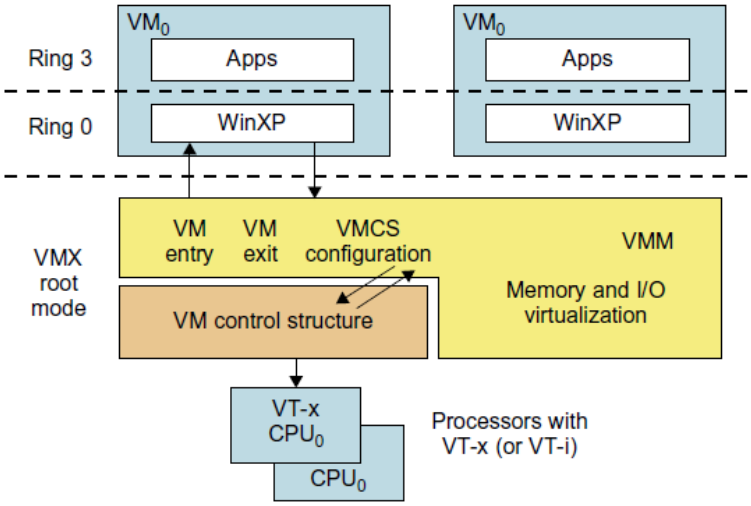


FIGURE 3.11 Intel hardware-assisted CPU virtualization

CPU state for VMs, a set of additional instructions is added. At the time of this writing, Xen, VMware, and the Microsoft Virtual PC all implement their hypervisors by using the VT-x technology. Generally, hardware-assisted virtualization should have high efficiency. However, since the transition from the hypervisor to the guest OS incurs high overhead switches between processor modes, it sometimes cannot outperform binary translation.

Hence, virtualization systems such as VMware now use a hybrid approach, in which a few tasks are offloaded to the hardware but the rest is still done in software. In addition, para-virtualization and hardware-assisted virtualization can be combined to improve the performance further.

Memory Virtualization

Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory.

All modern x86 CPUs include a ***memory management unit (MMU)*** and ***a translation look aside buffer (TLB)*** to optimize virtual memory performance. However, in a virtual execution environment, virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs. That means a two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory.

Furthermore, MMU virtualization should be supported, which is transparent to the guest OS. The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs. But the guest OS cannot directly access the actual machine memory. The VMM is responsible for mapping the guest physical memory to the actual machine memory. Figure 3.12 shows the two-level memory mapping procedure.

Since each page table of the guest OSes has a separate page table in the VMM corresponding to it, the VMM page table is called the *shadow page table*. Nested page tables add another layer of indirection to virtual memory. The MMU already handles virtual-to-physical translations as defined by the OS. Then the physical memory addresses are translated to machine addresses using another set of page tables defined by the hypervisor. Since modern operating systems maintain a set of page tables for every process, the shadow page tables will get flooded. Consequently, the performance overhead and cost of memory will be very high

VMware uses shadow page tables to perform virtual-memory-to-machine-memory address translation. Processors use TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access. When the guest OS changes the virtual memory to a physical memory mapping, the VMM updates the shadow page tables to enable a direct lookup

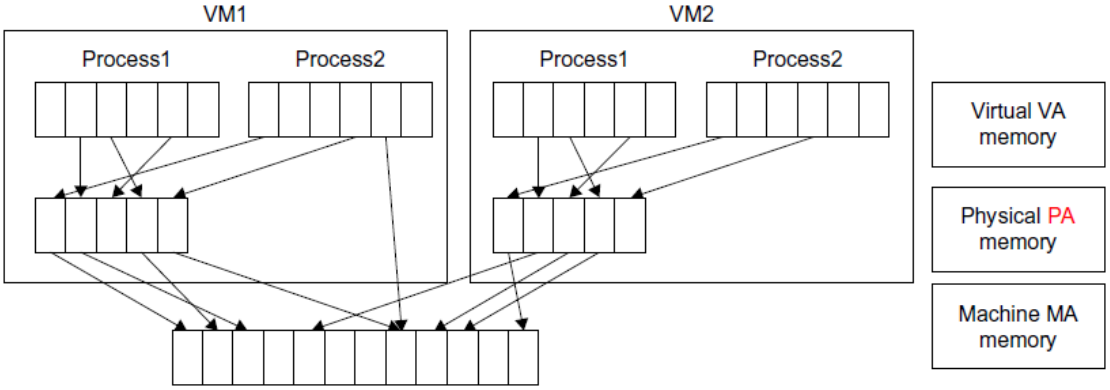


FIGURE 3.12 Two-level memory mapping procedure

The AMD Barcelona processor has featured hardware-assisted memory virtualization since 2007. It provides hardware assistance to the two-stage address translation in a virtual execution environment by using a technology called nested paging.

I/O Virtualization

8. What is I/O virtualization? What changes should be incorporated to map the virtual devices to real devices? and also show how packet transmission happens though the virtual NIC card? (jan 2015)

I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware. At the time of this writing, there are three ways to implement I/O virtualization:

1. Full device emulation
2. Para-virtualization
3. Direct I/O.

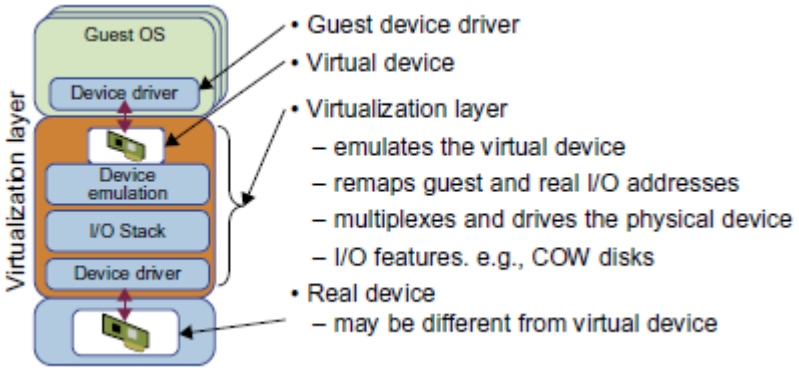


FIGURE 3.14: Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices into the virtual devices for the guest device driver to use.

1. ***Full device*** emulation is the first approach for I/O virtualization. Generally, this approach emulates well-known, real-world devices. All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated in software. This software is located in the VMM and acts as a virtual device. The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices. The full device emulation approach is shown in Figure 3.14. A single hardware device can be shared by multiple VMs that run concurrently. However, software emulation runs much slower than the hardware it emulates.
2. The ***para-virtualization method*** of I/O virtualization is typically used in Xen. It is also known as the split driver model consisting of a frontend driver and a backend driver. The frontend driver is running in Domain U and the backend driver is running in Domain 0. They interact with each other via a block of shared memory. The frontend driver manages the I/O requests of the guest OSes and the backend driver is responsible for managing the real I/O devices and multiplexing the I/O data of different VMs. Although para-I/O-virtualization achieves better device performance than full device emulation, it comes with a higher CPU overhead.
3. ***Direct I/O virtualization*** lets the VM access devices directly. It can achieve close-to-native performance without high CPU costs. However, current direct I/O virtualization implementations focus on networking for mainframes. There are a lot of challenges for commodity hardware devices.

For example, when a physical device is reclaimed (required by workload migration) for later reassignment, it may have been set to an arbitrary state (e.g., DMA to some arbitrary memory locations) that can function incorrectly or even crash the whole system. Since software-based I/O virtualization requires a very high overhead of device emulation, hardware-assisted I/O virtualization is critical. Intel VT-d supports the remapping of I/O DMA transfers and device-generated interrupts.

The architecture of VT-d provides the flexibility to support multiple usage models that may run unmodified, special-purpose, or "virtualization-aware" guest OSes. Another way to help I/O virtualization is via self-virtualized I/O (SV-IO). The key idea of SV-IO is to harness the rich resources of a multicore processor. All tasks associated with virtualizing an I/O device are encapsulated in SV-IO. It provides virtual devices and an associated access API to VMs and a management API to the VMM. SV-IO defines one virtual interface (VIF) for every kind of virtualized I/O device, such as virtual network interfaces, virtual block devices (disk), virtual camera devices, and others.

The guest OS interacts with the VIFs via VIF device drivers. Each VIF consists of two message queues. One is for outgoing messages to the devices and the other is for incoming messages from the devices. In addition, each VIF has a unique ID for identifying it in SV-IO.

Example 3.7 VMware Workstation for I/O Virtualization

The VMware Workstation runs as an application. It leverages the I/O device support in guest OSes, host OSes, and VMM to implement I/O virtualization. The application portion (VMApp) uses a driver loaded into the host operating system (VMDriver) to establish the privileged VMM, which runs directly on the hardware. A given physical processor is executed in either the host world or the VMM world, with the VMDriver facilitating the transfer of control between the two worlds. The VMware Workstation employs full device emulation to implement I/O virtualization. Figure 3.15 shows the functional blocks used in sending and receiving packets via the emulated virtual NIC.

The virtual NIC models an AMD Lance Am79C970A controller. The device driver for a Lance controller in the guest OS initiates packet transmissions by reading and writing a sequence of virtual I/O ports; each read or write switches back to the VMApp to emulate the Lance port accesses. When the last OUT instruction of the sequence is encountered, the Lance emulator calls a normal write() to the VMNet driver. The VMNet driver then passes the packet onto the network via a host NIC and then the VMApp switches back to the VMM. The switch raises a virtual interrupt to notify the guest device driver that the packet was sent. Packet receives occur in reverse.

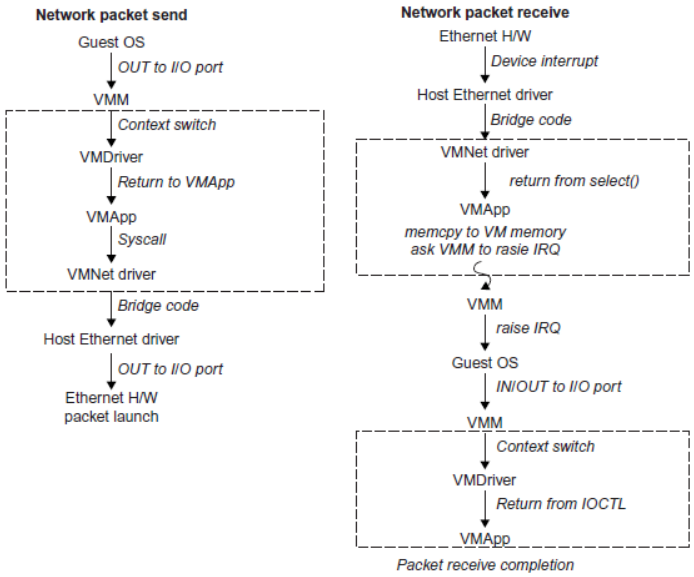


FIGURE 3.15 Functional blocks involved in sending and receiving network packets.

9. **Explain the detail Virtualization in Multi-Core Processors ?**

Virtualizing a multi-core processor is relatively more complicated than virtualizing a uni-core processor. Though multicore processors are claimed to have higher performance by integrating multiple processor cores in a single chip, muti-core virtualiuzation has raised some new challenges to computer architects, compiler constructors, system designers, and application programmers. There are mainly two difficulties: Application programs must be parallelized to use all cores fully, and software must explicitly assign tasks to the cores, which is a very complex problem.

Concerning the **first** challenge, new programming models, languages, and libraries are needed to make parallel programming easier. The **second** challenge has spawned research involving scheduling algorithms and resource management policies. Yet these efforts cannot balance well among performance, complexity, and other issues. What is worse, as technology scales, a new challenge called dynamic heterogeneity is emerging to mix the fat CPU core and thin GPU cores on the same chip, which further complicates the multi-core or many-core resource management. The dynamic heterogeneity of hardware infrastructure mainly comes from less reliable transistors and increased complexity in using the transistors

1. Physical versus Virtual Processor Cores

Wells, et al. proposed a multicore virtualization method to allow hardware designers to get an abstraction of the low-level details of the processor cores. This technique alleviates the burden and inefficiency of managing hardware resources by software. It is located under the ISA and remains unmodified by the operating system or VMM (hypervisor). Figure 3.16 illustrates the technique of a software-visible VCPU moving from one core to another and temporarily suspending execution of a VCPU when there are no appropriate cores on which it can run.

2. Virtual Hierarchy

The emerging many-core chip multiprocessors (CMPs) provides a new computing landscape. Instead of supporting time-sharing jobs on one or a few cores, we can use the abundant cores in a space-sharing, where single-threaded or multithreaded jobs are simultaneously assigned to separate groups of cores for long time intervals. This idea was originally suggested by Marty and Hill. To optimize for space-shared workloads, they propose using virtual hierarchies to overlay a coherence and caching hierarchy onto a physical processor. Unlike a fixed physical hierarchy, a virtual hierarchy can adapt to fit how the work is space shared for improved performance and performance isolation.

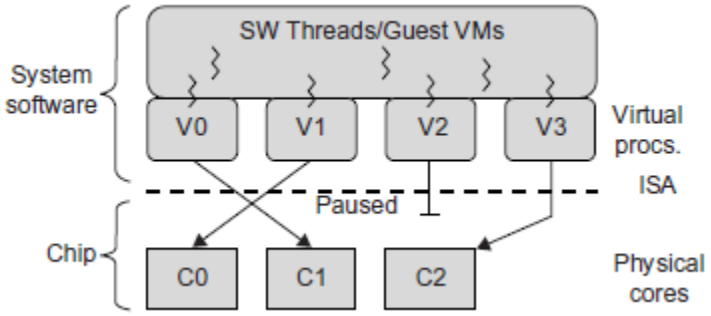
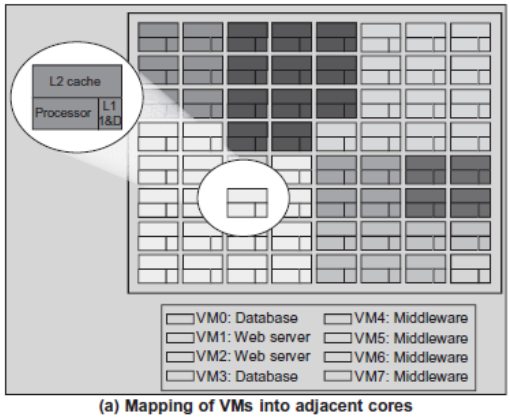


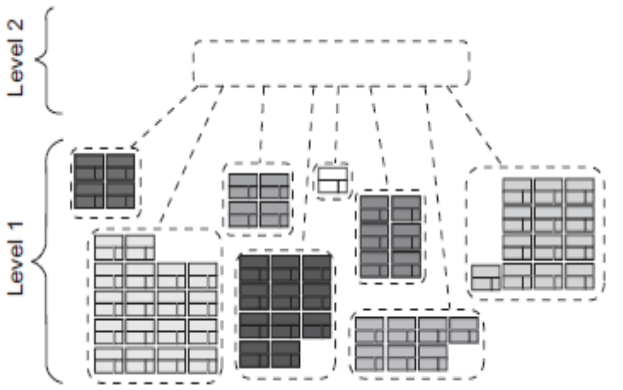
FIGURE 3.16 : Multicore virtualization method that exposes four VCPUs to the software, when only three cores are actually Present

Today's many-core CMPs use a physical hierarchy of two or more cache levels that statically determine the cache allocation and mapping. A virtual hierarchy is a cache hierarchy that can adapt to fit the workload or mix of workloads . The hierarchy's first level locates data blocks close to the cores needing them for faster access, establishes a shared-cache domain, and establishes a point of coherence for faster communication. When a miss leaves a tile, it first attempts to locate the block (or sharers) within the first level. The first level can also provide isolation between independent workloads. A miss at the L1 cache can invoke the L2 access.



(a) Mapping of VMs into adjacent cores

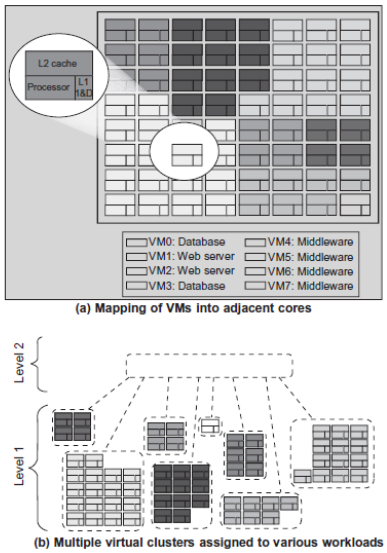
The idea is illustrated in Figure 3.17(a). Space sharing is applied to assign three workloads to three clusters of virtual cores: namely VM0 and VM3 for database workload, VM1 and VM2 for web server workload, and VM4–VM7 for middleware workload. The basic assumption is that each workload runs in its own VM. However, space sharing applies equally within a single operating system. Statically distributing the directory among tiles can do much better, provided operating systems or hypervisors carefully map virtual pages to physical frames. Marty and Hill suggested a two-level virtual coherence and caching hierarchy that harmonizes with the assignment of tiles to the virtual clusters of VMs.



(b) Multiple virtual clusters assigned to various workloads

Figure 3.17(b) illustrates a logical view of such a virtual cluster hierarchy in two levels. Each VM operates in a isolated fashion at the first level. This will minimize both miss access time and performance interference with other workloads or VMs. Moreover, the shared resources of cache capacity, inter-connect links, and miss handling are mostly isolated between VMs. The second level maintains a globally shared memory. This facilitates dynamically repartitioning resources without costly cache flushes. Furthermore, maintaining globally shared memory minimizes changes to existing system software and allows virtualization features such as content-based page sharing. A virtual hierarchy adapts to space-shared workloads like multiprogramming and server consolidation.

Figure 3.17 shows a case study focused on consolidated server workloads in a tiled architecture. This many-core mapping scheme can also optimize for space-shared multiprogrammed workloads in a single-OS environment.



(a) Mapping of VMs into adjacent cores

(b) Multiple virtual clusters assigned to various workloads

FIGURE 3.17 CMP server consolidation by space-sharing of VMs into many cores forming multiple virtual clusters to execute various workloads.

VIRTUAL CLUSTERS AND RESOURCE MANAGEMENT

10. Compare virtual and physical clusters .Explain how resource management done for virtual clusters? (jan 2105) or Explain Physical versus Virtual Clusters in detail ?

A physical cluster is a collection of servers (physical machines) interconnected by a physical network such as a LAN. When a traditional VM is initialized, the administrator needs to manually write configuration information or specify the configuration sources. When more VMs join a network, an inefficient configuration always causes problems with overloading or underutilization. Amazon's Elastic Compute Cloud (EC2) is a good example of a web service that provides elastic computing power in a cloud. EC2 permits customers to create VMs and to manage user accounts over the time of their use.

Most virtualization platforms, including XenServer and VMware ESX Server, support a bridging mode which allows all domains to appear on the network as individual hosts. By using this mode, VMs can communicate with one another freely through the virtual network interface card and configure the network automatically.

Physical versus Virtual Clusters

Virtual clusters are built with VMs installed at distributed servers from one or more physical clusters. The VMs in a virtual cluster are interconnected logically by a virtual network across several physical networks. Figure 3.18 illustrates the concepts of virtual clusters and physical clusters. Each virtual cluster is formed with physical machines or a VM hosted by multiple physical clusters. The virtual cluster boundaries are shown as distinct boundaries.

The provisioning of VMs to a virtual cluster is done dynamically to have the following interesting properties:

- The virtual cluster nodes can be either physical or virtual machines. Multiple VMs running with different OSes can be deployed on the same physical node.
- A VM runs with a guest OS, which is often different from the host OS, that manages the resources in the physical machine, where the VM is implemented.
- The purpose of using VMs is to consolidate multiple functionalities on the same server. This will greatly enhance server utilization and application flexibility.

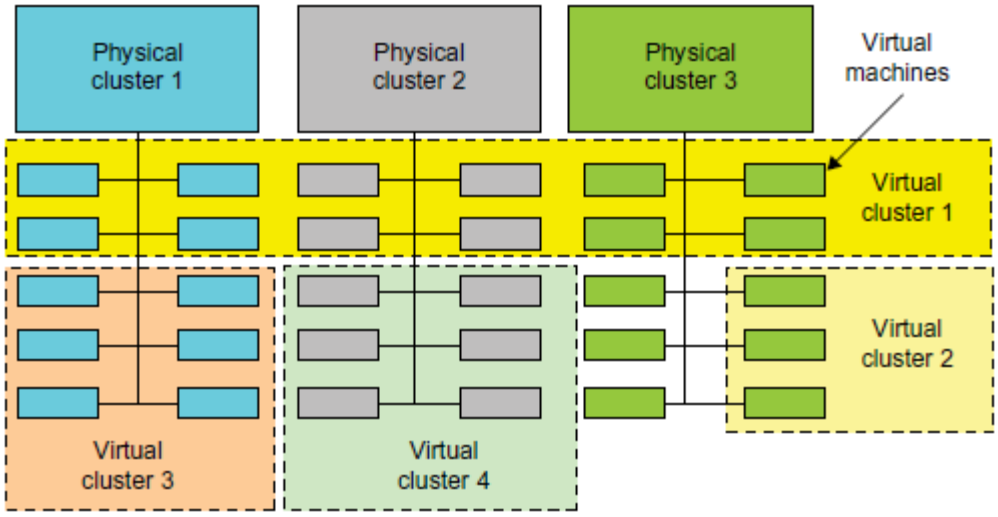


FIGURE 3.18 A cloud platform with four virtual clusters over three physical clusters shaded differently.

- VMs can be colonized (replicated) in multiple servers for the purpose of promoting distributed parallelism, fault

tolerance, and disaster recovery.

- The size (number of nodes) of a virtual cluster can grow or shrink dynamically, similar to the way an overlay network varies in size in a peer-to-peer (P2P) network.
- The failure of any physical nodes may disable some VMs installed on the failing nodes. But the failure of VMs will not pull down the host system.

Since system virtualization has been widely used, it is necessary to effectively manage VMs running on a mass of physical computing nodes (also called virtual clusters) and consequently build a high-performance virtualized computing environment. This involves virtual cluster deployment, monitoring and management over large-scale clusters, as well as resource scheduling, load balancing, server consolidation, fault tolerance, and other techniques. The different node colors in Figure 3.18 refer to different virtual clusters. In a virtual cluster system, it is quite important to store the large number of VM images efficiently. Figure 3.19 shows the concept of a virtual cluster based on application partitioning or customization.

The different colors in the figure represent the nodes in different virtual clusters. As a large number of VM images might be present, the most important thing is to determine how to store those images in the system efficiently. There are common installations for most users or applications, such as operating systems or user-level programming libraries.

These software packages can be preinstalled as templates (called ***template VMs***). With these templates, users can build their own software stacks. New OS instances can be copied from the template VM. User-specific components such as programming libraries and applications can be installed to those instances. Three physical clusters are shown on the left side of Figure 3.18. Four virtual clusters are created on the right, over the physical clusters. The physical machines are also called host systems.

In contrast, the VMs are guest systems. The host and guest systems may run with different operating systems. Each VM can be installed on a remote server or replicated on multiple servers belonging to the same or different physical clusters. The boundary of a virtual cluster can change as VM nodes are added, removed, or migrated dynamically over time.

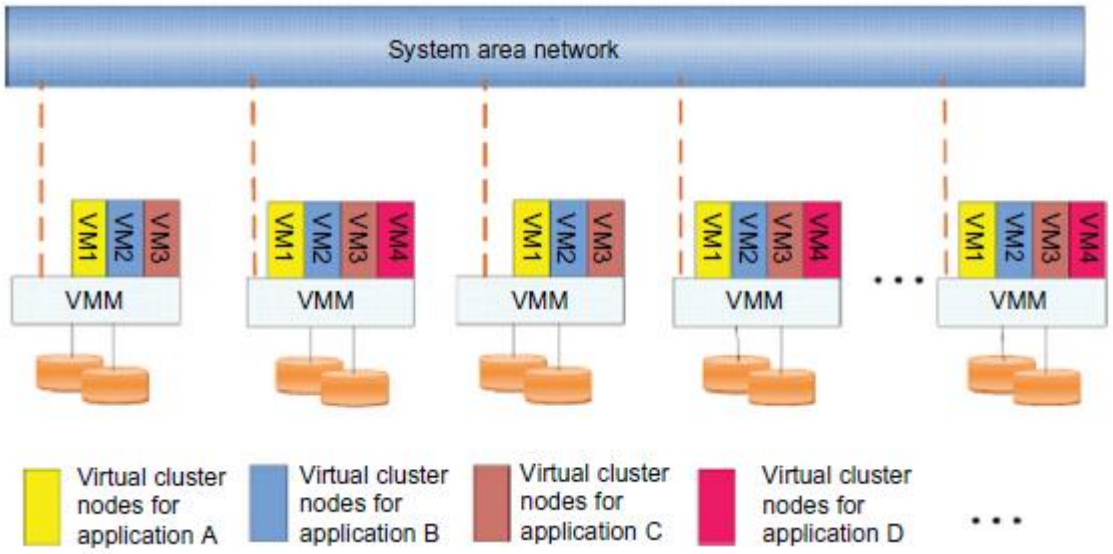


FIGURE 3.19 The concept of a virtual cluster based on application partitioning.

1. Fast Deployment and Effective Scheduling

The system should have the capability of fast deployment. Here, deployment means two things:

i. To construct and distribute software stacks (OS, libraries, applications) to a physical node inside clusters as fast as possible

ii. To quickly switch runtime environments from one user's virtual cluster to another user's virtual cluster.

If one user finishes using his system, the corresponding virtual cluster should shut down or suspend quickly to save the resources to run other VMs for other users.

- The concept of "green computing" has attracted much attention recently. However, previous approaches have focused on saving the energy cost of components in a single workstation without a global vision. Consequently, they do not necessarily reduce the power consumption of the whole cluster. Other cluster-wide energy-efficient techniques can only be applied to homogeneous workstations and specific applications.
- The live migration of VMs allows workloads of one node to transfer to another node. However, it does not guarantee that VMs can randomly migrate among themselves. In fact, the potential overhead caused by live migrations of VMs cannot be ignored.
- The overhead may have serious negative effects on cluster utilization, throughput, and QoSissues. Therefore, the challenge is to determine how to design migration strategies to implement green computing without influencing the performance of clusters.
- Another advantage of virtualization is load balancing of applications in a virtual cluster. Load balancing can be achieved using the load index and frequency of user logins.
- The automatic scale-up and scale-down mechanism of a virtual cluster can be implemented based on this model. Consequently, we can increase the resource utilization of nodes and shorten the response time of systems.
- Mapping VMs onto the most appropriate physical node should promote performance.
- Dynamically adjusting loads among nodes by live migration of VMs is desired, when the loads on cluster nodes become quite unbalanced.

2. High-Performance Virtual Storage

The template VM can be distributed to several physical hosts in the cluster to customize the VMs. In addition, existing software packages reduce the time for customization as well as switching virtual environments. It is important to efficiently manage the disk spaces occupied by template software packages.

Some storage architecture design can be applied to reduce duplicated blocks in a distributed file system of virtual clusters. Hash values are used to compare the contents of data blocks. Users have their own profiles which store the identification of the data blocks for corresponding VMs in a user-specific virtual cluster. New blocks are created when users modify the corresponding data. Newly created blocks are identified in the users' profiles.

Basically, there are four steps to deploy a group of VMs onto a target cluster:

i. Preparing the disk image
ii. Configuring the VMs
iii. Choosing the destination nodes
iv. Executing the VM deployment command on every host

A **template** is a disk image that includes a preinstalled operating system with or without certain application software. Users choose a proper template according to their requirements and make a duplicate of it as their own disk image.

Templates could implement the ***COW (Copy on Write)*** format. A new COW backup file is very small and easy to create and transfer. Therefore, it definitely reduces disk space consumption. In addition, VM deployment time is much shorter than that of copying the whole raw image file.

Every VM is configured with a name, disk image, network setting, and allocated CPU and memory. One needs to record each VM configuration into a file. However, this method is inefficient when managing a large group of VMs. VMs with the same configurations could use pre edited profiles to simplify the process.

In this scenario, the system configures the VMs according to the chosen profile. Most configuration items use the same settings, while some of them, such as UUID, VM name, and IP address, are assigned with automatically calculated values. Normally, users do not care which host is running their VM.

11. Explain the Live VM Migration Steps and its Performance Effects?

In a cluster built with mixed nodes of host and guest systems, the normal method of operation is to run everything on the physical machine. When a VM fails, its role could be replaced by another VM on a different node, as long as they both run with the same guest OS. In other words, a physical node can fail over to a VM on another host. This is different from physical-to-physical failover in a traditional physical cluster. The advantage is enhanced failover flexibility.The potential drawback is that a VM must stop playing its role if its residing host node fails.

However, this problem can be mitigated with VM life migration. Figure 3.20 shows the process of life migration of a VM from host A to host B. The migration copies the VM state file from the storage area to the host machine. There are ***four ways to manage a virtual cluster***.

First, you can use a guest-based manager, by which the cluster manager resides on a guest system. In this case, multiple VMs form a virtual cluster. For example, openMosix is an open source Linux cluster running different guest systems on top of the Xen hypervisor. Another example is Sun's cluster Oasis, an experimental Solaris cluster of VMs supportedby a VMware VMM.

Second, you can build a cluster manager on the host systems. The host-basedmanager supervises the guest systems and can restart the guest system on another physical machine. A good example is the VMware HA system that can restart a guest system after failure. These two cluster management systems are either guest-only or host-only, but they do not mix.

Third way to manage a virtual cluster is to use an independent cluster manager on both the host and guest systems. This will make infrastructure management more complex, however.

Finally, you can use an integrated cluster on the guest and host systems.

This means the manager must be designed to distinguish between virtualized resources and physical resources. VMs can be live-migrated from one physical machine to another; in case of failure, one VM can be replaced by another VM.

Virtual clusters can be applied in computational grids, cloud platforms, and high-performance computing (HPC) systems. In particular, virtual clustering plays a key role in cloud computing. When a VM runs a live service, it is necessary to make a trade-off to ensure that the migration occurs in a manner that minimizes all three metrics.

The motivation is to design a live VM migration scheme with negligible downtime, the lowest network bandwidth consumption possible, and a reasonable total migration time. we should ensure that the migration will not disrupt other active services residing in the same host through resource contention (e.g., CPU, network bandwidth).

A VM can be in one of the following four states.

- An inactive state is defined by the virtualization platform, under which the VM is not enabled.
- An active state refers to a VM that has been instantiated at the virtualization platform to perform a real task.
- A paused state corresponds to a VM that has been instantiated but disabled to process a task or paused in a waiting state.
- A VM enters the suspended state if its machine file and virtual resources are stored back to the disk.

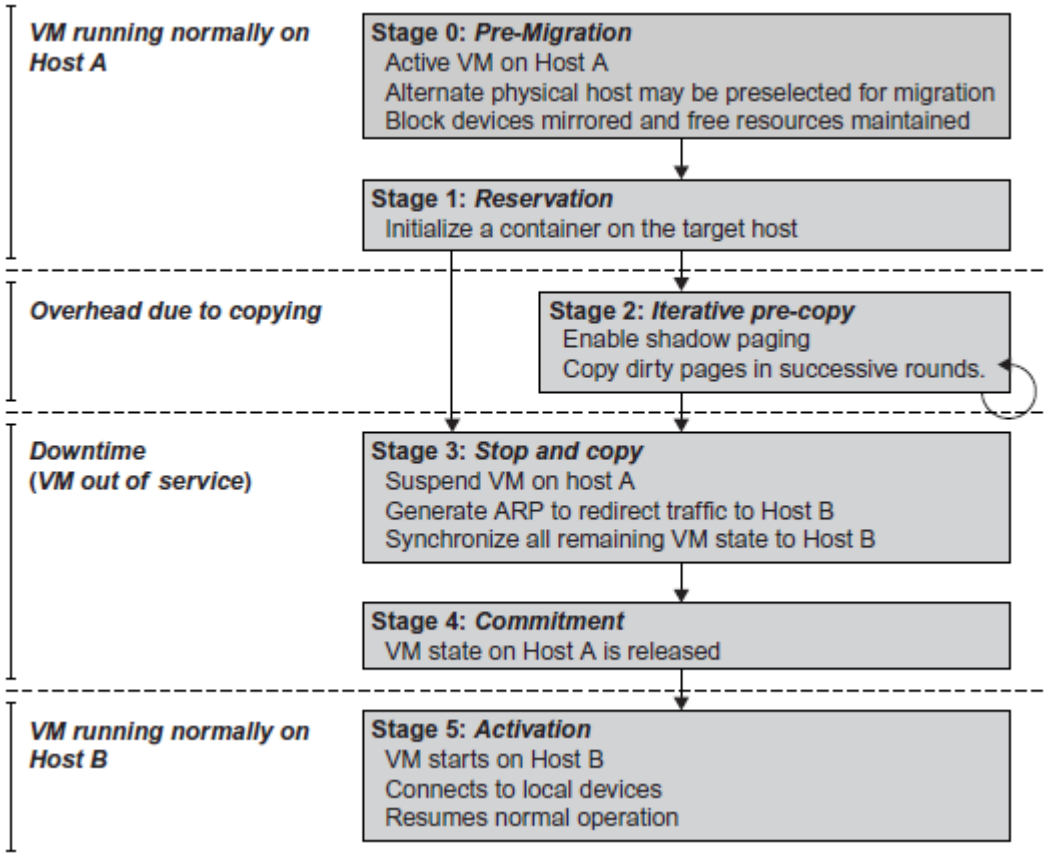


FIGURE 3.20 Live migration process of a VM from one host to another

As shown in Figure 3.20, live migration of a VM consists of the following six steps:

Steps 0 and 1: Start migration. This step makes preparations for the migration, including determining the migrating VM and the destination host. Although users could manually make a VM migrate to an appointed host, in most circumstances, the migration is automatically started by strategies such as load balancing and server consolidation.

Steps 2: Transfer memory. Since the whole execution state of the VM is stored in memory, sending the VM's memory to the destination node ensures continuity of the service provided by the VM. All of the memory data is transferred in the first round, and then the migration controller recopies the memory data which is changed in the last round. These steps keep iterating until the dirty portion of the memory is small enough to handle the final copy. Although precopying memory is performed iteratively, the execution of programs is not obviously interrupted.

Step 3: Suspend the VM and copy the last portion of the data. The migrating VM's execution is suspended when the last round's memory data is transferred. Other nonmemory data such as CPU and network states should be sent as well. During this step, the VM is stopped and its applications will no longer run. This "service navailable" time is called the "downtime" of migration, which should be as short as possible so that it can be negligible to users.

Steps 4 and 5: Commit and activate the new host. After all the needed data is copied, on the destination host, the VM reloads the states and recovers the execution of programs in it, and the service provided by this VM continues. Then the network connection is redirected to the new VM and the dependency to the source host is cleared. The whole migration process finishes by removing the original VM from the source host.

12. Explain in detail the Migration of Memory, Files, and Network Resources?
Write short notes on Memory Migration (or) Write short notes on File System Migration (or) Write short notes on Network Migration

Since clusters have a high initial cost of ownership, including space, power conditioning, and cooling equipment, leasing or sharing access to a common cluster is an attractive solution when demands vary over time. Shared clusters offer economies of scale and more effective utilization of resources by multiplexing. Early configuration and management systems focus on expressive and scalable mechanisms for defining clusters for specific types of service, and physically partition cluster nodes among those types. When one system migrates to another physical node, we should consider the following issues.

1. Memory Migration

This is one of the most important aspects of VM migration. Moving the memory instance of a VM from one physical host to another can be approached in any number of ways. But traditionally, the concepts behind the techniques tend to share common implementation paradigms. The techniques employed for this purpose depend upon the characteristics of application/workloads supported by the guest OS.

Memory migration can be in a range of hundreds of megabytes to a few gigabytes in a typical system today, and it needs to be done in an efficient manner. The Internet Suspend-Resume (ISR) technique exploits temporal locality as memory states are likely to have considerable overlap in the suspended and the resumed instances of a VM.

Temporal locality refers to the fact that the memory states differ only by the amount of work done since a VM was last suspended before being initiated for migration. To exploit temporal locality, each file in the file system is represented as a tree of small subfiles.

A copy of this tree exists in both the suspended and resumed VM instances. The advantage of using a tree-based representation of files is that the caching ensures the transmission of only those files which have been changed.

The ISR technique deals with situations where the migration of live machines is not a necessity. Predictably, the downtime is high, compared to some of the other techniques discussed later.

2. File System Migration

To support VM migration, a system must provide each VM with a consistent, location-independent view of the file system that is available on all hosts. A simple way to achieve this is to provide each VM with its own virtual disk which the file system is mapped to and transport the contents of this virtual disk along with the other states of the VM.

However, due to the current trend of high capacity disks, migration of the contents of an entire disk over a network is not a viable solution. Another way is to have a global file system across all machines where a VM could be located. This way removes the need to copy files from one machine to another because all files are network accessible.

A distributed file system is used in ISR serving as a transport mechanism for propagating a suspended VM state. The actual file systems themselves are not mapped onto the distributed file system. Instead, the VMM only accesses its local file system. The relevant VM files are explicitly copied into the local file system for a resume operation and taken out of the local file system for a suspend operation.

This approach relieves developers from the complexities of implementing several different file system calls for different distributed file systems. It also essentially disassociates the VMM from any particular distributed file system semantics.

However, this decoupling means that the VMM has to store the contents of each VM's virtual disks in its local files, which have to be moved around with the other state information of that VM. In smart copying, the VMM exploits spatial locality.

Typically, people often move between the same small number of locations, such as their home and office. In these conditions, it is possible to transmit only the difference between the two file systems at suspending and resuming locations. This technique significantly reduces the amount of actual physical data that has to be moved.

3. Network Migration

A migrating VM should maintain all open network connections without relying on forwarding mechanisms on the original host or on support from mobility or redirection mechanisms. To enable remote systems to locate and communicate with a VM, each VM must be assigned a virtual IP address known to other entities. This address can be distinct from the IP address of the host machine where the VM is currently located. Each VM can also have its own distinct virtual MAC address.

The VMM maintains a mapping of the virtual IP and MAC addresses to their corresponding VMs. In general, a migrating VM includes all the protocol states and carries its IP address with it. If the source and destination machines of a VM migration are typically connected to a single switched LAN, an unsolicited ARP reply from the migrating host is provided advertising that the IP has moved to a new location.

This solves the open network connection problem by reconfiguring all the peers to send future packets to a new location. Although a few packets that have already been transmitted might be lost, there are no other problems with this mechanism.

Alternatively, on a switched network, the migrating OS can keep its original Ethernet MAC address and rely on the network switch to detect its move to a new port.

- Live migration means moving a VM from one physical node to another while keeping its OS environment and applications unbroken. This capability is being increasingly utilized in today's enterprise environments to provide efficient online system maintenance, reconfiguration, load balancing, and proactive fault tolerance.
- Traditional migration suspends VMs before the transportation and then resumes them at the end of the process. By importing the precopy mechanism, a VM could be livemigrated without stopping the VM and keep the applications running during the migration.
- Live migration is a key feature of system virtualization technologies. Here, We focus on VMmigration within a cluster environment where a network-accessible storage system, such as storage area network (SAN) or network attached storage (NAS), is employed. Only memory and CPU status needs to be transferred from the source node to the target node.
- Live migration techniques mainly use the precopy approach, which first transfers all memory pages, and then only copies modified pages during the last round iteratively. The VM service downtime is expected to be minimal by using iterative copy operations. When applications' writable working set becomes small, the VM is suspended and only the CPU state and dirty pages in the last round are sent out to the destination.
- In the precopy phase, although a VM service is still available, much performance degradation will occur because the migration daemon continually consumes network bandwidth to transfer dirty pages in each round. An adaptive rate limiting approach is employed to mitigate this issue, but total migration time is prolonged by nearly 10 times.
- the maximum number of iterations must be set because not all applications' dirty pages are ensured to converge to a small writable working set over multiple rounds . these issues with the precopy approach are caused by the large amount of transferred data during the whole migration process.
- A checkpointing/recovery and trace/replay approach (CR/ TR-Motion) is proposed to provide fast VM migration. This approach transfers the execution trace file in iterations rather than dirty pages, which is logged by a trace daemon. Apparently, the total size of all log files is much less than that of dirty pages.
- total migration time and downtime of migration are drastically reduced. However, CR/TR-Motion is valid only when the log replay rate islarger than the log growth rate. The inequality between source and target nodes limits the applicationscope of live migration in clusters.
- Another strategy of postcopy is introduced for live migration of VMs. Here, all memory pages are transferred only once during the whole migration process and the baseline total migration time is reduced. But the downtime is much higher than that of precopy due to the latency of fetching pages from the source node before the VM can be resumed on the target.

- With the advent of multicore or many-core machines, abundant CPU resources are available. Even if several VMs reside on a same multicore machine, CPU resources are still rich because physical CPUs are frequently amenable to multiplexing.

13. Explain with an example for Live Migration of VM Using Xen?

Xen as a VMM or hypervisor, allows multiple commodity OSes to share x86 hardware in a safe and orderly fashion. The following example explains how to perform live migration of a VM between two Xen-enabled host machines. Domain 0 (or Dom0) performs tasks to create, terminate, or migrate to another host. Xen uses a send/recv model to transfer states across VMs.

Example 3.8 Live Migration of VMs between Two Xen-Enabled Hosts

Xen supports live migration. It is a useful feature and natural extension to virtualization platforms that allows for the transfer of a VM from one physical machine to another with little or no downtime of the services hosted by the VM. Live migration transfers the working state and memory of a VM across a network when it is running. Xen also supports VM migration by using a mechanism called Remote Direct Memory Access (RDMA). RDMA speeds up VM migration by avoiding TCP/IP stack processing overhead. RDMA implements a different transfer protocol whose origin and destination VM buffers must be registered before any transfer

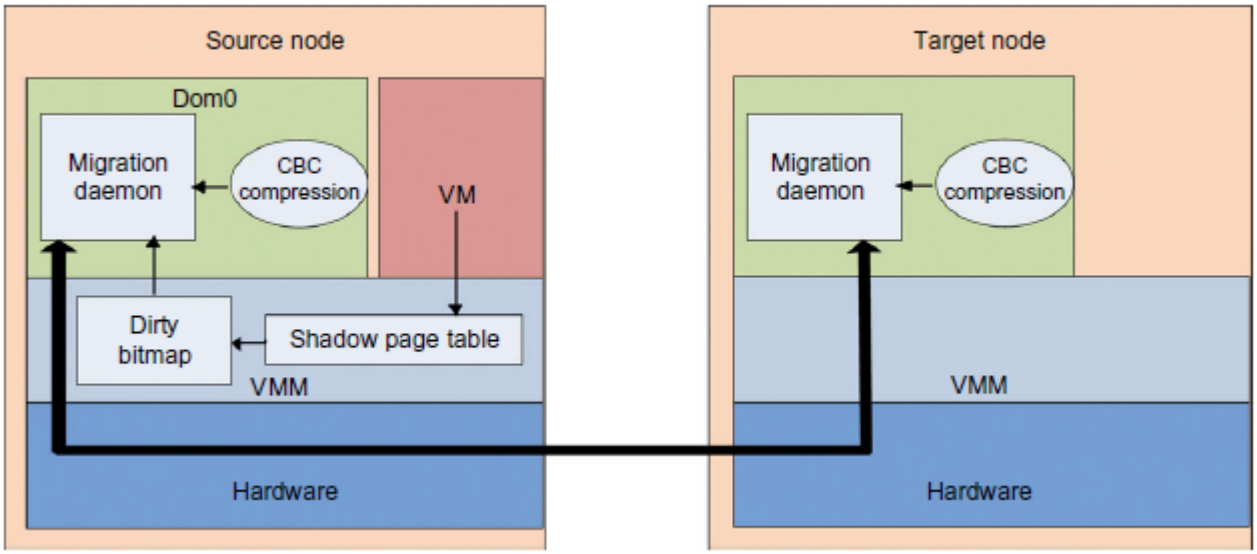


FIGURE 3.22 Live migration of VM from the Dom0 domain to a Xen-enabled target host

operations occur, reducing it to a "one-sided" interface. Data communication over RDMA does not need to involve the CPU, caches, or context switches. This allows migration to be carried out with minimal impact on guest operating systems and hosted applications. Figure 3.22 shows the a compression scheme for VM migration.

This design requires that we make trade-offs between two factors. If an algorithm embodies expectations about the kinds of regularities in the memory footprint, it must be very fast and effective. A single compression algorithm for all memory data is difficult to achieve the win-win status that we expect. Therefore, it is necessary to provide compression algorithms to pages with different kinds of regularities.

The structure of this live migration system is presented in Dom0. Migration daemons running in the management VMs are responsible for performing migration. Shadow page tables in the VMM layer trace modifications to the memory page in migrated VMs during the precopy phase.

Corresponding flags are set in a dirty bitmap. At the start of each precopy round, the bitmap is sent to the migration daemon. Then, the bitmap is cleared and the shadow page tables are destroyed and re-created in the next round. The system resides in Xen's management VM. Memory pages denoted by bitmap are extracted and compressed before they are sent to the destination. The compressed data is then decompressed on the target.

VIRTUALIZATION FOR DATA-CENTER AUTOMATION

14. Write short notes on Server Consolidation in Data Centers?

In data centers, a large number of heterogeneous workloads can run on servers at various times. These heterogeneous workloads can be roughly divided into two categories:

Chatty workloads

Chatty workloads may burst at some point and return to a silent state at some other point. A web video service is an example of this, whereby a lot of people use it at night and few people use it during the day

Noninteractiveworkloads.

Noninteractive workloads do not require people's efforts to make progress after they are submitted. High-performance computing is a typical example of this. At various stages, the requirements for resources of these workloads are dramatically different

- To guarantee that a workload will always be able to cope with all demand levels, the workload is statically allocated enough resources so that peak demand is satisfied. Figure 3.29 illustrates server virtualization in a data center. In this case, the granularity of resource optimization is focused on the CPU, memory, and network interfaces.

- It is common that most servers in data centers are underutilized. A large amount of hardware, space, power, and management cost of these servers is wasted. Server consolidation is an approach to improve the low utility ratio of hardware resources by reducing the number of physical servers. Among several server onsolidation techniques such as centralized and physical consolidation, virtualization-based server consolidation is the most powerful.

- Data centers need to optimize their resource management. Yet these techniques are performed with the granularity of a full server machine, which makes resource management far from well optimized. Server virtualization enables smaller resource allocation than a physical machine.

In general, the use of VMs increases resource management complexity. This causes a challenge in terms of how to improve resource utilization as well as guarantee QoS in data centers. In detail, server virtualization has the following side effects:

• Consolidation enhances hardware utilization. Many underutilized servers are consolidated into fewer servers to enhance resource utilization. Consolidation also facilitates backup services and disaster recovery.

• This approach enables more agile provisioning and deployment of resources. In a virtual environment, the images of the guest OSes and their applications are readily cloned and reused.

• The total cost of ownership is reduced. In this sense, server virtualization causes deferred purchases of new servers, a smaller data-center footprint, lower maintenance costs, and lower power, cooling, and cabling requirements.

• This approach improves availability and business continuity.

- To automate data-center operations, one must consider resource scheduling, architectural support, power management, automatic or autonomic resource management, performance of analytical models, and so on. Scheduling and reallocations can be done in a wide range of levels in a set of data centers. The levels

match at least at the VM level, server level, and data-center level. Ideally, scheduling and resource reallocations should be done at all levels. However, due to the complexity of this, current techniques only focus on a single level or, at most, two levels.

- Dynamic CPU allocation is based on VM utilization and application-level QoS metrics.

One method considers both CPU and memory flowing as well as automatically adjusting resource overhead based on varying workloads in hosted services.

Another scheme uses a two-level resource management system to handle the complexity involved. A local controller at the VM level and a global controller at the server level are designed. They implement autonomic resource allocation via the interaction of the local and global controllers. Multicore and virtualization are two cutting techniques that can enhance each other.

One can design a virtual hierarchy on a CMP in data centers. One can consider protocols that minimize the memory access time, inter-VM interferences, facilitating VM reassignment, and supporting inter-VM sharing. One can also consider a VM-aware power budgeting scheme using multiple managers integrated to achieve better power management. Consequently, one must address the trade-off of power saving and data-center performance.

15. Explain with a diagram the Virtual Storage Management?

"storage virtualization" was widely used before the renaissance of system virtualization. Yet the term has a different meaning in a system virtualization environment. Previously, storage virtualization was largely used to describe the aggregation and repartitioning of disks at very coarse time scales for use by physical machines.

In system virtualization, virtual storage includes the storage managed byVMMs and guest OSes.

Generally, the data stored in this environment can be classified into two categories:

1. VM images
2. Application data.

- The VM images are special to the virtual environment, while application data includes all other data which is the same as the data in traditional OS environments .The most important aspects of system virtualization are encapsulation and isolation.
- Traditional operating systems and applications running on them can be encapsulated in VMs. Only one operating system runs in a virtualization while many applications run in the operating system.
- System virtualization allows multiple VMs to run on a physical machine and the VMs are completely isolated. To achieve encapsulation and isolation, both the system software and the hardware platform, such as CPUs and chipsets, are rapidly updated. However, storage is lagging.
- In virtualization environments, a virtualization layer is inserted between the hardware and traditional operating systems or a traditional operating system is modified to support virtualization. This procedure complicates storage operations. On the one hand, storage management of the guest OS performs as though it is operating in a real hard disk while the guest OSes cannot access the hard disk directly.
- On the other hand, many guest OSes contest the hard disk when many VMs are running on a single physical machine. Therefore, storage management of the underlying VMM is much more complex than that of guest OSes (traditional OSes).
- the storage primitives used by VMs are not nimble. Hence, operations such as remapping volumes across hosts and check pointing disks are frequently clumsy and esoteric, and sometimes simply unavailable.
- In data centers, there are often thousands of VMs, which cause the VM images to become flooded. Many researchers tried to solve these problems in virtual storage management.

The main purposes of their research are to make management easy while enhancing performance and reducing the amount of storage occupied by the VM images.

Parallax is a distributed storage system customized for virtualization environments. Content Addressable Storage (CAS) is a solution to reduce the total size of VM images, and therefore supports a large set of VM-based systems in data centers.

Traditional storage management techniques do not consider the features of storage in virtualization environments, Parallax designs a novel architecture in which storage features that have traditionally been implemented directly on high-end storage arrays and switchers are relocated into a federation of storage VMs.

These storage VMs share the same physical hosts as the VMs that they serve. Figure 3.30 provides an overview of the Parallax system architecture. It supports all popular system virtualization techniques, such as paravirtualization and full virtualization. For each physical machine, Parallax customizes a special storage appliance VM. The storage appliance VM acts as a block virtualization layer between individual VMs and the physical storage device. It provides a virtual disk for each VM on the same physical machine.

Example 3.11 Parallax Providing Virtual Disks to Client VMs from a Large Common Shared Physical Disk The architecture of Parallax is scalable and especially suitable for use in cluster-based environments.
Figure 3.26 shows a high-level view of the structure of a Parallax-based cluster. A cluster-wide administrative domain manages all storage appliance VMs, which makes storage management easy. The storage appliance

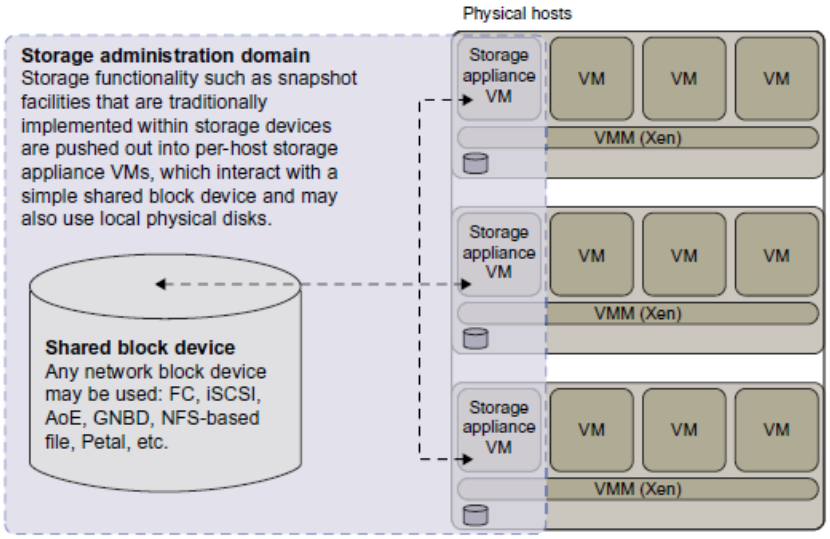


FIGURE 3.26 Parallax is a set of per-host storage appliances that share access to a common block device and presents virtual disks to client VMs

VM also allows functionality that is currently implemented within data-center hardware to be pushed out and implemented on individual hosts. This mechanism enables advanced storage features such as snapshot facilities to be implemented in software and delivered above commodity network storage targets.

Parallax itself runs as a user-level application in the storage appliance VM. It provides virtual disk images (VDIs) to VMs. A VDI is a single-writer virtual disk which may be accessed in a location-transparent manner from any of the physical hosts in the Parallax cluster. The VDIs are the core abstraction provided by Parallax. Parallax uses Xen's block tap driver to handle block requests and it is implemented as a tapdisk library. This library acts as a single block virtualization service for all client VMs on the same physical host. In the Parallax system, it is the storage appliance VM that connects the physical hardware device for block and network access.. This implementation enables a storage administrator to live-upgrade the block device drivers in an active cluster.

16. Write short notes on Cloud OS for Virtualized Data Centers?

Data centers must be virtualized to serve as cloud providers. Table 3.6 summarizes four virtual infrastructure (VI) managers and OSes. These VI managers and OSes are specially tailored for virtualizing data centers which often own a large number of servers in clusters. Nimbus, Eucalyptus, and OpenNebula are all open source software available to the general public. Only vSphere 4 is a proprietary OS for cloud resource virtualization and management over data centers.

These VI managers are used to create VMs and aggregate them into virtual clusters as elastic resources. Nimbus and Eucalyptus support essentially virtual networks. OpenNebula has additiona features to provision dynamic resources and make advance reservations. All three public VI managers apply Xen and KVM for virtualization. vSphere 4 uses the hypervisors ESX and ESXi from VMware. Only vSphere 4 supports virtual storage in addition to virtual networking and data protection. We will study Eucalyptus and vSphere 4 in the next two examples.

Table 3.6 VI Managers and Operating Systems for Virtualizing Data Centers [9]

Manager/ OS, Platforms, License	Resources Being Virtualized, Web Link	Client API, Language	Hypervisors Used	Public Cloud Interface	Special Features
Nimbus Linux, Apache v2	VM creation, virtual cluster, www .nimbusproject.org/	EC2 WS, WSRF, CLI	Xen, KVM	EC2	Virtual networks
Eucalyptus Linux, BSD	Virtual networking (Example 3.12 and [41]), www .eucalyptus.com/	EC2 WS, CLI	Xen, KVM	EC2	Virtual networks
OpenNebula Linux, Apache v2	Management of VM, host, virtual network, and scheduling tools, www.opennebula.org/	XML-RPC, CLI, Java	Xen, KVM	EC2, Elastic Host	Virtual networks, dynamic provisioning
vSphere 4 Linux, Windows, proprietary	Virtualizing OS for data centers (Example 3.13), www .vmware.com/ products/vsphere/ [66]	CLI, GUI, Portal, WS	VMware ESX, ESXi	VMware vCloud partners	Data protection, vStorage, VMFS, DRM, HA

17. Discuss Eucalyptus for Virtual Networking of Private Cloud?

Eucalyptus is an open source software system (Figure 3.27) intended mainly for supporting Infrastructure as a Service (IaaS) clouds. The system primarily supports virtual networking and the management of VMs; virtual storage is not supported. Its purpose is to build private clouds that can interact with end users through Ethernet or the Internet. The system also supports interaction with other private clouds or public clouds over the Internet.

The system is short on security and other desired features for general-purpose grid or cloud applications. The designers of Eucalyptus implemented each high-level system component as a stand-alone web service. Each web service exposes a well-defined language-agnostic API in the form of a WSDL document containing both operations that the service can perform and input/output data structures

Furthermore, the designers leverage existing web-service features such as WS-Security policies for secure communication between components.

The three resource managers in Figure 3.27 are specified below:

• Instance Manager controls the execution, inspection, and terminating of VM instances on the host where it runs.
• Group Manager gathers information about and schedules VM execution on specific instance managers, as well as manages virtual instance network.
• Cloud Manager is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, makes scheduling decisions, and implements them by making requests to group managers.

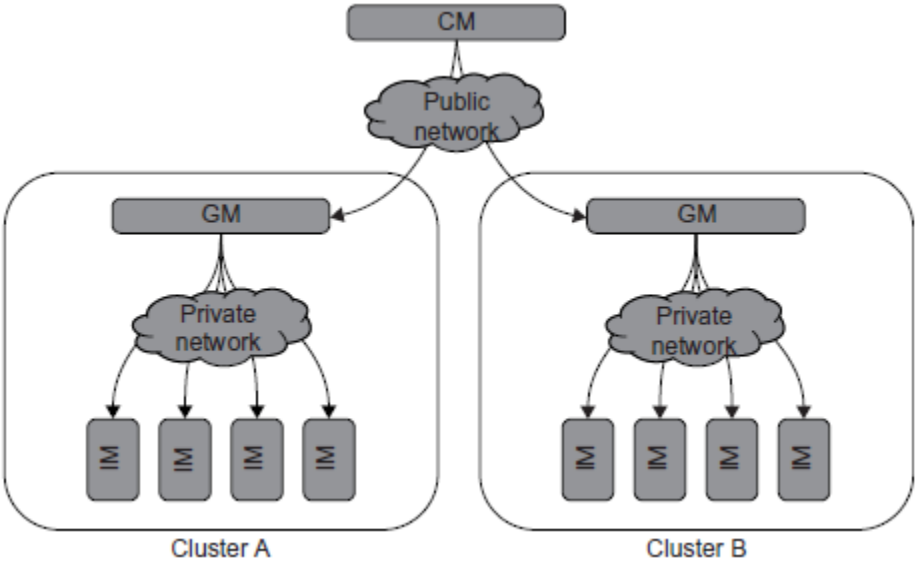


FIGURE 3.27 : Eucalyptus for building private clouds by establishing virtual networks over the VMs linking through Ethernet and the Internet.

In terms of functionality, Eucalyptus works like AWS APIs. Therefore, it can interact with EC2. It does provide a storage API to emulate the Amazon S3 API for storing user data and VM images. It is installed on Linux-based platforms, is compatible with EC2 with SOAP and Query, and is S3-compatible with SOAP and REST. CLI and web portal services can be applied with Eucalyptus.

18. Discuss VMware vSphere 4 as a Commercial Cloud OS with a neat diagram ?

The vSphere 4 offers a hardware and software ecosystem developed by VMware and released in April 2009. vSphere extends earlier virtualization software products by VMware, namely the VMware Workstation, ESX for server virtualization, and Virtual Infrastructure for server clusters. Figure 3.28 shows vSphere's overall architecture.

The system interacts with user applications via an interface layer, called vCenter. vSphere is primarily intended to offer virtualization support and resource management of data-center resources in building private clouds. VMware claims the system is the first cloud OS that supports availability, security, and scalability in providing cloud computing services

The vSphere 4 is built with two functional software suites: infrastructure services and application services. It also has three component packages intended mainly for virtualization purposes: vCompute is supported by ESX, ESXi, and DRS virtualization libraries from VMware; vStorage is supported by VMS and thin provisioning libraries; and vNetwork offers distributed switching and networking functions. These packages interact with the hardware servers, disks, and networks in the data center. These infrastructure functions also communicate with other external clouds

The application services are also divided into three groups: availability, security, and scalability. Availability support includes VMotion, Storage VMotion, HA, Fault Tolerance, and Data Recovery from VMware. The security package supports vShield Zones and VMsafe. The scalability package was built with DRS and Hot Add. Interested readers should refer to the vSphere 4 web site for more details regarding these component software functions. To fully understand the use of vSphere 4, users must also learn how to use the vCenter interfaces in order to link with existing applications or to develop new applications

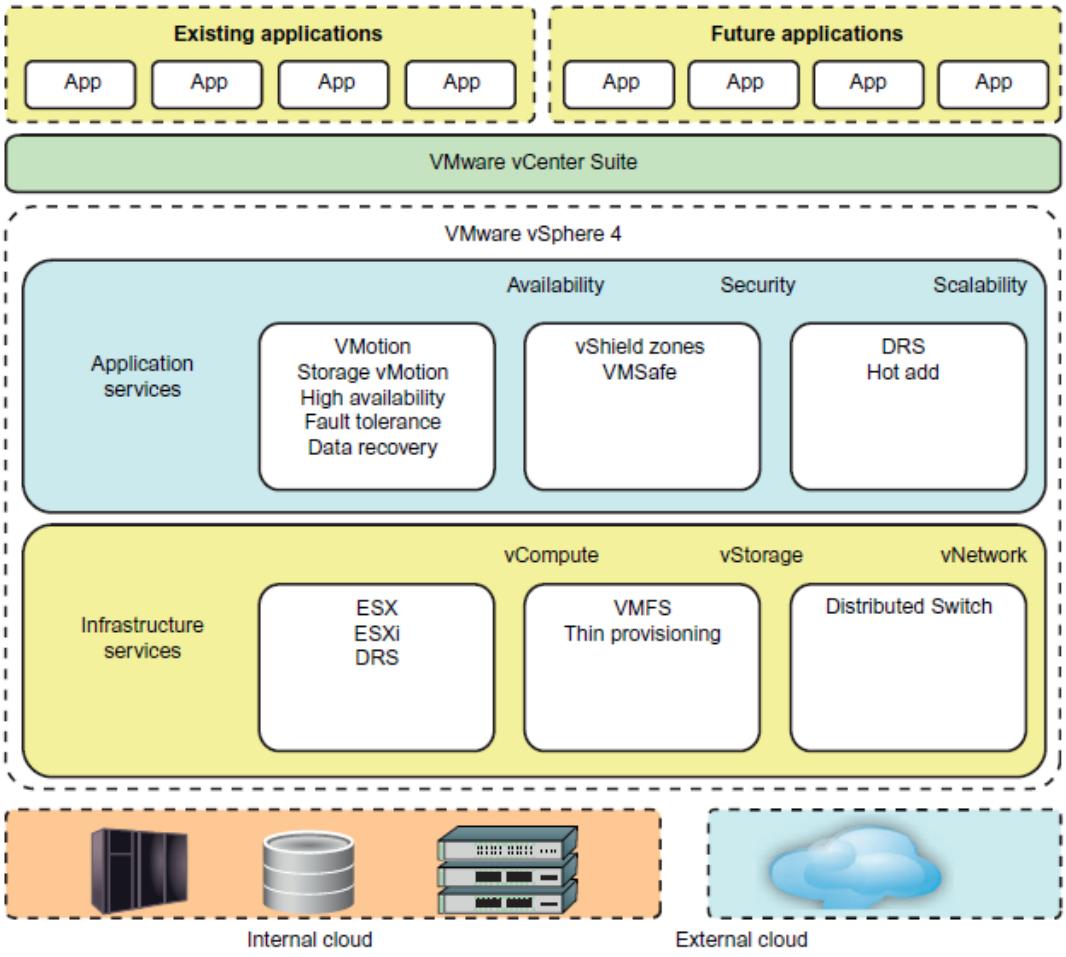


FIGURE 3.28 vSphere/4, a cloud operating system that manages compute, storage, and network resources over virtualized data centers.

19. Explain in detail the Trust Management in Virtualized Data Centers?

A VMM changes the computer architecture. It provides a layer of software between the operating systems and system hardware to create one or more VMs on a single physical platform. A VM entirely encapsulates the state of the guest operating system running inside it. Encapsulated machine state can be copied and shared over the network and removed like a normal file, which proposes a challenge to VM security. In general, a VMM can provide secure isolation and a VM accesses hardware resources through the control of the VMM, so the VMM is the base of the security of a virtual system. Normally, one VM is taken as a management VM to have some privileges such as creating, suspending, resuming, or deleting a VM.

Once a hacker successfully enters the VMM or management VM, the whole system is in danger. A subtler problem arises in protocols that rely on the "freshness" of their random number source for generating session keys. Considering a VM, rolling back to a point after a random number has been chosen, but before it has been used, resumes execution; the random number, which must be "fresh" for security purposes, is reused. With a stream cipher, two different plaintexts could be encrypted under the same key stream, which could, in turn, expose both plaintexts if the plaintexts have sufficient redundancy. Non cryptographic protocols that rely on freshness are also at risk. For example, the reuse of TCP initial sequence numbers can raise TCP hijacking attacks.

1. VM-Based Intrusion Detection

Intrusions are unauthorized access to a certain computer from local or network users and intrusion detection is used to recognize the unauthorized access. An intrusion detection system (IDS) is built on operating systems, and is based on the characteristics of intrusion actions. A typical IDS can be classified as a host-based IDS (HIDS) or a network-based IDS (NIDS), depending on the data source. A HIDS can be implemented on the monitored system. When the monitored system is attacked by hackers, the HIDS also faces the risk of being attacked. A NIDS is based on the flow of network traffic which can't detect fake actions.

Virtualization-based intrusion detection can isolate guest VMs on the same hardware platform. Even some VMs can be invaded successfully; they never influence other VMs, which is similar to the way in which a NIDS operates. Furthermore, a VMM monitors and audits access requests for hardware and system software. This can avoid fake actions and possess the merit of a HIDS. There are two different methods for implementing a VM-based IDS: Either the IDS is an independent process in each VM or a high-privileged VM on the VMM; or the IDS is integrated into the VMM

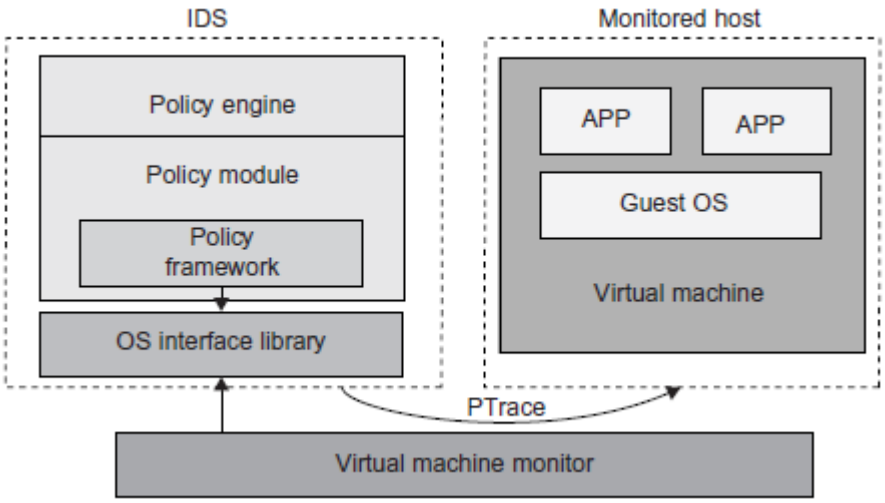


FIGURE 3.29 The architecture of livewire for intrusion detection using a dedicated VM.

and has the same privilege to access the hardware as well as the VMM. Garfinkel and Rosenblum have proposed an IDS to run on a VMM as a high-privileged VM. Figure 3.29 illustrates the concept.

The VM-based IDS contains a policy engine and a policy module. The policy framework can monitor events in different guest VMs by operating system interface library and PTrace indicates trace to secure policy of monitored host. It's difficult to predict and prevent all intrusions without delay. Therefore, an analysis of the intrusion action is extremely important after an intrusion occurs.

At the time of this writing, most computer systems use logs to analyze attack actions, but it is hard to ensure the credibility and integrity of a log. The IDS log service is based on the operating system kernel. Thus, when an operating system is invaded by attackers, the log service should be unaffected. Besides IDS, honeypots and honeynets are also prevalent in intrusion detection.

They attract and provide a fake system view to attackers in order to protect the real system. In addition, the attack action can be analyzed, and a secure IDS can be built. A honeypot is a purposely defective system that simulates an operating system to cheat and monitor the actions of an attacker. A honeypot can be divided into physical and virtual forms. A guest operating system and the applications running on it constitute a VM. The host operating system and VMM must be guaranteed to prevent attacks from the VM in a virtual honeypot.

Example 3.14 EMC Establishment of Trusted Zones for Protection of Virtual Clusters Provided to Multiple Tenants EMC and VMware have joined forces in building security middleware for trust management in distributed systems and private clouds. The concept of trusted zones was established as part of the virtual infrastructure. Figure 3.30 illustrates the concept of creating trusted zones for virtual clusters (multiple

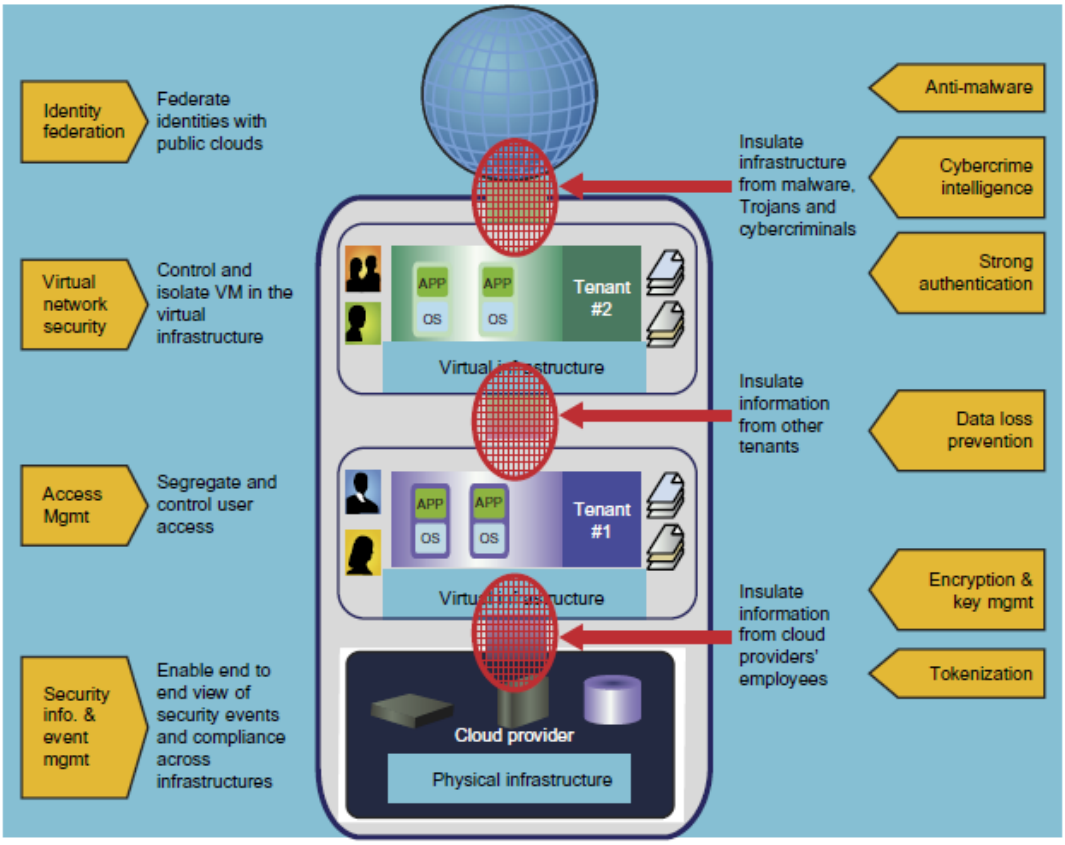


FIGURE 3.30 Techniques for establishing trusted zones for virtual cluster insulation and VM isolation.

applications and OSes for each tenant) provisioned in separate virtual environments. The physical infrastructure is shown at the bottom, and marked as a cloud provider. The virtual clusters or infrastructures are shown in the upper boxes for two tenants. The public cloud is associated with the global user communities at the top.

The arrowed boxes on the left and the brief description between the arrows and the zoning boxes are security functions and actions taken at the four levels from the users to the providers. The small circles between the four boxes refer to interactions between users and providers and among the users themselves. The arrowed boxes on the right are those functions and actions applied between the tenant environments, the provider, and the global communities.

Almost all available countermeasures, such as anti-virus, worm containment, intrusion detection, encryption and decryption mechanisms, are applied here to insulate the trusted zones and isolate the VMs for private tenants. The main innovation here is to establish the trust zones among the virtual clusters. The end result is to enable an end-to-end view of security events and compliance across the virtual clusters dedicated to different tenants.

IMPORTANT QUESTIONS
PART A

1) What is private cloud?
2) What is public cloud?
3) What is hybrid cloud?
4) What is a Community Cloud ?
5) Define IaaS?
6) Define PaaS?
7) Define SaaS?
8) What is meant by virtualization?
9) What are the implementation levels of virtualization?
10) What is User-Application Level virtualization (or) Process level virtualization?
11) What is hardware level virtualization?
12) What is library level virtualization?
13) Write short notes on xen architecture?
14) What are the types of hardware virtualization ?
15) Write short notes on KVM (Kernel-Based VM)
16) Write short notes about memory virtualization?
17) Write short notes on para-virtualization
18) What are virtual clusters?
19) What is network migration?
20) What is data center?
21) What is hardware abstraction level of virtualization?
22) What is operating system level of virtualization?
23) What is library support level of virtualization?
24) Explain Host OS and Guest OS?
25) Define KVM?
26) Define the three resource managers?
27) Define trust management in cloud computing?

PART B

1) Explain with a neat diagram public, private, hybrid and community clouds? Explain in detail the Cloud Ecosystem and Enabling Technologies? Why is cloud called as ecosystem? Justify **(may/june 2014)**
2) Enlist and explain the various services and deployment method of cloud computing environment **?(jan 2015)**
3) Discuss about the Pros and Cons of cloud computing? **(May/June 2014)**
4) Explain the characteristics and types of virtualization in cloud computing ? **(May/June 2014)**
5) Explain the detail Virtualization in Multi-Core Processors ?
6) Compare virtual and physical clusters .Explain how resource management done for virtual clusters? **(jan 2105)**
7) Explain in detail the Migration of Memory, Files, and Network Resources?

UNIT II
GRID SERVICES

Introduction to Open Grid Services Architecture (OGSA) – Motivation – Functionality Requirements –Practical & Detailed view of OGSA/OGSI – Data intensive grid service models – OGSA services.

PART-A

1) Define OGSA Framework

- ✓ The OGSA was built on two basic software technologies: the Globus Toolkit widely adopted as a grid technology solution for scientific and technical computing, and web services (WS 2.0) as a popular standards-based framework for business and network applications.
- ✓ The OGSA is intended to support the creation, termination, management, and invocation of stateful, transient grid services via standard interfaces and conventions.
- ✓ The OGSA framework specifies the physical environment, security, infrastructure profile, resource provisioning, virtual domains, and execution environment for various grid services and API access tools.
- ✓ Compared with the layered grid architecture, the OGSA is service-oriented.

2) What are the Two key properties of a grid service

The Two key properties of a grid service are

- ✓ Transience
- ✓ Statefulness

These properties have significant implications regarding how a grid service is named, discovered, and managed.

- ✓ Being transient means the service can be created and destroyed dynamically
- ✓ Statefulness refers to the fact that one can distinguish one service instance from another.

3) Define Grid Service Migration

- ✓ This is a mechanism for creating new services and specifying assertions regarding the lifetime of a service. The OGSA model defines a standard interface, known as a factor, to implement this reference.
- ✓ Any service that is created must address the former services as the reference of later services.
- ✓ This creates a requested grid service with a specified interface and returns the GSH and initial GSR for the new service instance. It should also register the new service instance with a handle resolution service.

4) Define Interoperability of resources

- ✓ Grid systems can be more easily and efficiently developed and deployed when utilizing a variety of languages and a variety of platforms. For example, it is desirable to mix service-provider components, work-dispatch tracking systems, and systems management; this makes it easier to dispatch work to service providers and for service providers to support grid services.

5) What are the case scenarios in OGSA that have been considered

- ✓ National fusion collaboration
- ✓ IT infrastructure and management
- ✓ Commercial data centers
- ✓ Service-based distributed query processing
- ✓ Severe storm prediction
- ✓ Online media and entertainment

6) List out the Basic Functionality Requirements

The following basic functions are universally fundamental:

- ✓ Discovery and brokering
- ✓ Metering and accounting
- ✓ Data sharing
- ✓ Deployment
- ✓ Virtual organizations (VOs)
- ✓ Monitoring
- ✓ Policy

7) List out the Security Requirements in OGSA

Grids also introduce a rich set of security requirements; some of these requirements are:

- ✓ Multiple security infrastructures
- ✓ Perimeter security solutions
- ✓ Authentication, Authorization, and Accounting
- ✓ Encryption
- ✓ Application and Network-Level Firewalls
- ✓ Certification

8) Give the Resource Management Requirements in OGSA.

Resource management is another multilevel requirement, encompassing SLA negotiation, provisioning, and scheduling for a variety of resource types and activities:

- ✓ Provisioning
- ✓ Resource virtualization
- ✓ Optimization of resource usage
- ✓ Transport management
- ✓ Access
- ✓ Management and monitoring
- ✓ Processor
- ✓ Scheduling of service tasks
- ✓ Load balancing
- ✓ Advanced reservation
- ✓ Notification and messaging
- ✓ Logging
- ✓ Workflow management
- ✓ Pricing

9) What are all the System Properties Requirements in OGSA .

A number of grid-related capabilities can be thought of as desirable system properties rather than functions:

- ✓ Fault tolerance
- ✓ Disaster recovery
- ✓ Self-healing capabilities of resources, services and systems are required.
- ✓ Strong monitoring for defects, intrusions, and other problems.
- ✓ Legacy application management
- ✓ Administration
- ✓ Agreement-based interaction

10) Define Grouping/aggregation of services

The ability to instantiate (compose) services using some set of existing services is a key requirement. There are two main types of composition techniques:

- Selection
- Aggregation

- ✓ Selection involves choosing to use a particular service among many services with the same operational interface.
- ✓ Aggregation involves orchestrating a functional flow (workflow) between services. For example, the output of an accounting service is fed into the rating service to produce billing records.

11) What is OGSA/OGSI? A practical view

- ✓ The OGSA standard defines what grid services are, what they should be capable of, and what technologies they are based on. OGSA, however, does not go into specifics of the technicalities of the specification; instead, the aim is to help classify what is and is not a grid system .
- ✓ It is called an *architecture* because it is mainly about describing and building a well-defined set of interfaces from which systems can be built, based on open standards such as WSDL.

12) Give two fundamental requirements for describing Web services based on the OGSI

1. The ability to describe interface inheritance—a basic concept with most of the distributed object systems.

2. The ability to describe additional information elements with the interface definitions.

13) What is Grid Data Services.

- ✓ Grid data services are also depicted in Figure 4.11. These interfaces support the concept of data virtualization and provide mechanisms related to distributed access to information of many types including databases, files, documents, content stores, and application-generated streams.

- ✓ Services that comprise the grid data services class complement the computing virtualization conventions specified by program execution services (OGSA placing data resources on an equivalent level with computing resources).

14) Give the incorporate the concepts of OGSI defines a component model that extends WSDL and XML schema definition

- Stateful Web services
- Extension of Web services interfaces
- Asynchronous notification of state change
- References to instances of services
- Collections of service instances
- Service state data that augment the constraint capabilities of XML schema definition

15) What are the core properties of grid service.

- Defining grid service description and grid service instance, as organizing principles for their extension and their use
- Defining how OGSI models time
- Defining the grid service handle and grid service reference constructs that are used to refer to grid service instances
- Defining a common approach for conveying fault information from operations.This approach defines a base XML schema definition and associated semantics for WSDL fault messages to support a common interpretation; the approach simply defines the base format for fault messages, without modifying the WSDL fault message model.
- Defining the life cycle of a grid service instance

16) Define Data-Intensive Grid Service Models.

- ✓ Applications in the grid are normally grouped into two categories: computation-intensive and dataintensive.
- ✓ For data-intensive applications, we may have to deal with massive amounts of data.
- ✓ For example, the data produced annually by a Large Hadron Collider may exceed several petabytes (1015 bytes).
- ✓ The grid system must be specially designed to discover, transfer, and manipulate these massive data sets. Transferring massive data sets is a time-consuming task. Efficient data management demands low-cost storage and high-speed data movement.
- ✓ This data access method is also known as caching, which is often applied to enhance data efficiency in a grid environment. By replicating the same data blocks and scattering them in multiple regions of a grid, users can access the same data with locality of references.

17) What are the four access models for organizing a data grid

- Monadic model
- Hierarchical model
- Federation model
- Hybrid model

18) Give the Four *base data interfaces* (WSDL portTypes) can be used to implement a variety of different data service behaviors:

1. ***DataDescription*** defines OGSI service data elements representing key parameters of the data virtualization encapsulated by the data service.
2. ***DataAccess*** provides operations to access and/or modify the contents of the data virtualization encapsulated by the data service.
3. ***DataFactory*** provides an operation to create a new data service with a data virtualization derived from the data virtualization of the parent (factory) data service.
4. ***DataManagement*** provides operations to monitor and manage the data service's data virtualization, including

19) Give the Other Data Services

- ✓ A variety of higher-level data interfaces can and must be defined on top of the base data interfaces, to address functions such as:
  - Data access and movement
  - Data replication and caching
  - Data and schema mediation
  - Metadata management and looking

20) What are the Additional items that may come within the scope of the CMM Specification.

- New data types or metadata to convey semantic meaning of manageability information,such as counter or gauge
- Versioning information
- Metadata to associate a metered usage (unit of measure) with manageability information
- Classification of properties such as metric and configuration
- Registries and locating fine-grained resources
- Managed resource identifier

21) Explain about Open Grid Services Architecture (OGSA) (Nov 10)

- ✓ The Open Grid Services Architecture (OGSA) provides a strategy for service providers to create service-oriented infrastructures which support more flexible resource management.
- ✓ Web Services supplies a paradigm that supports dynamic resource modeling, a fundamental requirement in pursuit of comprehensive management for evolutionary infrastructures.
- ✓ Peer-to-peer technology creates a mechanism for ad hoc relationships to be formed on demand, without a centralized controlling mechanism.

22) Explain about Grid Service Providers (GSP)

Grid Service Providers (GSP) thus supplies an enabling infrastructure which supports the user-driven services innovation of their customers, free from the delays associated with current infrastructural paradigms.

When each user has the ability to innovate network services based on their own needs, the promise of the past decade will have arrived.

23) Explain about Montague River Grid (MRG)

Montague River Grid (MRG) supplies the necessary functionality to support the inter-domain and inter-provider management of network-based services.

MRG is a self-organizing grid adapter/gateway for network-attached resources and is deployed in conjunction with technology specific domain managers. MRG acts as the community authority/virtual organization for locally advertised and controlled network based services.

Discovery, membership, registry, mapper, factory, notification, topology, and threading services are all supported.Each MRG supports an aggregate capabilities dictionary from which domain-level capabilities are inherited and re-advertised.

Furthermore, complex inter-domain services can be constructed and advertised as single service entities.Once deployed, MRG enables user controlled, end-to-end, inter-domain and inter-provider services.

24) What are the Components of the MRG? (Apr 11)

- Inter Domain Services—used to represent the persistent service datastore, service , etc.
- Inter Domain Factory—primary entrance factory for users
- Factory—operational interface; used to implement the process of managing network services
- Registry—used to identify existing persistent service instances; inherited operational functionality of network devices
- Mapper—used to extract detailed information about existing service instances
- Notifications—used to relay asynchronous alarms and notifications from the network to the pertinent registered users of the affected resources
- Membership—used to enhance path selection within a business relationship or service paradigm
- Discovery—used to identify and propagate existing services within a business relationship or service paradigm

25) Explain about Montague River Domain

Montague River Domain (MRD) supplies the necessary functionality to support device-specific, domain-level management for network-based services.MRD is a service fulfillment/configuration management platform for network-attached devices such as transport equipment, storage platforms, and computational servers.

Its dynamically coupled network model allows network and service evolution without the re-engineering of the platform. Each MRD implements a capabilities dictionary from which component and comprehensive services are composed for the specific devices within its domain.

Furthermore, MRD implements standard functionality such as journaled transaction management, inventory upload and reconciliation, service configuration and rollback, service and topology reporting, and alarm correlation.

26) What are the Components of the MRD?

- Domain Services—service configuration operations, e.g., provisioning, service discovery, service grooming, etc.
- Domain Factory—network configuration operations, e.g., upload, transaction management, etc.
- Security—service and network security, including resource tagging, user enablement, etc.
- Configuration—specific service configuration operations, e.g, partitionLightPath, findASPath, addXC, deleteXC, etc.
- Inventory—network device and component management, virtualized persistence of physical network, etc.
- Reporting—domain level network and service reporting.

27) What is Data Catalog?

The data catalog is meta-data that identifies the data sets being managed. Typical meta-data includes the name of the data set, its location, the date-time it was last modified, its size, its type, and the correct access method.

The catalog should be flexible enough to track everything from a subset of a file in a native operating system file format to a single record in a database manager to an entire database. Key issues in meta-data management include the completeness of the meta-data and the timeliness of updates to it.

Catalog management should be subject to strict access controls and most catalog maintenance activity should be assiduously logged.

28. What are Portals?

Web-based applications that encapsulate grid operations and present a uniform user interface to the grid user base can be a useful integration point for all this management capability.

For the average grid user, the portal provides a uniform user interface that masks the difference between the hardware and software resources available on the grid.

For the administrative user, the portal provides a coherent application environment that uniformly enforces access controls and ensures consistent logging.

29) What are the requirements for Grid-Enabling Software?

Two requirements must be met in order to modify software for grid deployment:Access to the application source code and the ability to modify it; in other words, both the legal right and the development expertise necessary to change an uncompiled application. There are three groups that meet these requirements.

- ✓ The first group consists of independent software vendors (ISVs) who develop and commercially distribute software applications. ISVs own their software code and have software developers in their employ.
- ✓ The second group is made up of academic institutions and enterprises in research-intensive industries such as life sciences that use open source software applications. Open source software licenses permit modifications of code, and in many cases allow redistribution of the modified version, subject to certain conditions.

- ✓ The third group consists of enterprises that have developed their own proprietary software applications for internal deployment, often with a view to securing competitive advantage through superior implementation of information technology. As these applications are proprietary, enterprises typically own or in some fashion retain intellectual property rights to the source code.

Both open source and proprietary software applications can be modified for grid deployment either by internal software developers or third-party solution integrators (SIs).

30) What are the process of Grid-enabling Software Applications?

- ✓ The process of grid-enabling a software application is fairly straightforward. Using the GridIron XLR8 application development tool, an experienced developer familiar with the software application to be grid-enabled should complete the code modifications in a reasonably short period of time.
- ✓ A distributable algorithm or job can be equivalently expressed as one or more steps. The most time-consuming, processing intensive steps that will be distributed for grid processing must have the following three characteristics:
  - They can be split into smaller tasks.
  - Each task can be processed on a separate computer.
  - The results from each task can be returned re-assembled into one final result.

31) Explain the Overview of GridIron XLR8

GridIron XLR8 is a product that allows software developers to add the speed of distributed computing to commercial software applications. XLR8 enables computationally intensive software applications to run faster on multiple computers.

GridIron XLR8 consists of two parts: An application developers' toolkit, or SDK, comprised of APIs that are added to the source code of a computationally intensive application, plus documentation, sample applications, and other tools and materials to assist software developers in modifying their code for processing by multiple computers; and runtime software that is installed on each computer in a network, providing additional processing power.

GridIron XLR8 reduces the complexity of embedding distributed computing within an application by providing all the necessary programmatic elements at a high level of abstraction. All of the job control logic can be defined and controlled through the use of just six XLR8 job control functions and four job execution methods provided by application plug-ins.

Additional XLR8 functions are available for administration, management, and data marshalling. By comparison, protocols such as MPI are significantly more complex, with some 380 primary calls.

Finally, GridIron XLR8 is embedded directly into the software applications. Once compiled and installed, users can benefit from the speed of distributed computing without having to change the way they use the application and without learning special skills.

32) What is Hyper threading?

There are a number of currently available technologies that provide the facility for performance improvement through coprocessor and software optimizations, such as vectorization

(e.g., AltiVec), Single Instruction Multiple Data (SIMD), Pthreads, SSE2, etc. One such technology is hyperthreading.

Hyperthreading is an evolving Intel processor technology (first available on Intel's XEON server processors and now being delivered on all desktop 3.06 GHz+ processors) that provides dual simultaneous execution of two threads on the same physical processor.

Performance improvements for most multi-threaded applications range from a typical 5 percent to a current theoretical maximum of approximately 30 percent.

Hyperthreading was utilized in this implementation to demonstrate that such technologies are complimentary to distributed computing and will achieve cumulative performance improvements.

33) What are the advantages a grid?

- Access—Seamless, transparent, remote, secure, wireless access to computing, data, experiments, instruments, sensors, etc.
- Virtualization—Access to compute and data services, not the servers themselves, without caring about the infrastructure.
- On Demand—Get resources you need, when you need them, at the quality you need.
- Sharing—Enable collaboration of (virtual) teams, over the Internet, to jointly work on one complex task.
- Failover—In case of system failure, migrate and restart applications automatically on another system.
- Heterogeneity—In large and complex grids, resources are heterogeneous (platforms, operating systems, devices, software, etc.). Users can choose the system that is best suited for their specific application.
- Utilization—Grids are known to increase average utilization from some 20 percent to 80 percent and more.

For example, our internal Sun Enterprise Grid (with currently more than 7,000 processors in three different locations) to design Sun's next-generation processors is utilized at more than 95 percent, on average.

34) Explain about the Globus Toolkit 2.0

The Globus Toolkit is an open architecture, open source software toolkit developed by the Globus Project. A brief explanation of GT2.0 is given here for completeness.

Full description of the Globus Toolkit can be found at the Globus Web site. GT3.0 re-implements much of the functionality of GT2.x but is based upon the Open Grid Services Architecture, OGSA.

In the following, the three core components of GT2.0 (and GT2.2).

- ✓ Globus Security Infrastructure (GSI)
- ✓ Globus Resource Allocation Manager (GRAM)
- ✓ Monitoring and Discovery Services (MDS)

35) What are the services provided by the Grid?

- ✓ Single sign-on—Globus creation using Grid Security Infrastructure and X509 certificates. This allows the user to seamlessly establish his or her identity across all campus grid resources.

- ✓ Resource information—Viewable status information on grid resources, both static and dynamic attributes such as operating systems,
- ✓ Job specification and submission—a GUI that enables the user to enter job specifications such as the compute resource, I/O, and queue requirements. Automated translation of these requirements into Resource specification language (RSL) and subsequent job submission to Globus Resource Allocation Managers (GRAM) are supported by the portal. Scripts have been implemented to enable job handoff to SGE via Globus services. Further, automated translation of some job requirements into SGE parameters is supported.
- ✓ Precise usage control—Policy-based authorization and accounting services to examine and evaluate usage policies of the resource providers. Such a model is critical when sharing resources in a heterogeneous environment such as the campus grid.
- ✓ Job management—Storage and retrieval of relevant application profile information, history of job executions, and related information. Application profiles are meta-data that can be composed to characterize the applications.
- ✓ Data handling—Users can transparently authenticate with and browse remote file systems of the grid resources. Data can be securely transferred between grid resources using the GSI-enabled data transport services.

36) Explain about Grid Engine Enterprise Edition

- ✓ Grid Engine Enterprise Edition (GEEE) is installed at each of the four nodes—Maxima, Snowdon, Titania, and Pascali. The command line and GUI of GEEE is the main access point to each node for local users. The Enterprise Edition version of Grid Engine provides policy driven resource management at the node level. There are four policy types which may be implemented:
- ✓ Share Tree Policy—GEEE keeps track of how much usage users/projects have already received. At each scheduling interval, the Scheduler adjusts all jobs' share of resources to ensure that users/groups and projects get very close to their allocated share of the system over the accumulation period.
- ✓ Functional Policy—Functional scheduling, sometimes called priority scheduling, is a non-feedback scheme for determining a job's importance by its association with the submitting user/project/department.
- ✓ Deadline Policy—Deadline scheduling ensures that a job is completed by a certain time by starting it soon enough and giving it enough resources to finish on time.
- ✓ Override Policy—Override scheduling allows the GEEE operator to dynamically adjust the relative importance of an individual job or of all the jobs associated with a user/department/project.

PART-B

1.Explain Open Grid Services Architecture (OGSA) in detail

- ✓ The OGSA is an open source grid service standard jointly developed by academia and the IT industry under coordination of a working group in the Global Grid Forum (GGF). The standard was specifically developed for the emerging grid and cloud service communities.

- ✓ The OGSA is extended from web service concepts and technologies. The standard defines a common framework that allows businesses to build grid platforms across enterprises and business partners. The intent is to define the standards required for both open source and commercial software to support a global grid infrastructure.

i) OGSA Framework
ii) OGSA Interfaces
iii) Grid Service Handle
iv) Grid Service Migration
v) OGSA Security Models

i) OGSA Framework

- ✓ The OGSA was built on two basic software technologies: the Globus Toolkit widely adopted as a grid technology solution for scientific and technical computing, and web services (WS 2.0) as a popular standards-based framework for business and network applications.
- ✓ The OGSA is intended to support the creation, termination, management, and invocation of stateful, transient grid services via standard interfaces and conventions.
- ✓ The OGSA framework specifies the physical environment, security, infrastructure profile, resource provisioning, virtual domains, and execution environment for various grid services and API access tools.
- ✓ Compared with the layered grid architecture, the OGSA is service-oriented.
- ✓ A service is an entity that provides some capability to its client by exchanging messages. We feel that greater flexibility is needed in grid service discovery and management.
- ✓ The service oriented architecture (SOA) serves as the foundation of grid computing services. The individual and collective states of resources are specified in this service standard.
- ✓ The standard also specifies interactions between these services within the particular SOA for grids. An important point is that the architecture is not layered, where the implementation of one service is built upon modules that are logically dependent.
- ✓ One may classify this framework as object-oriented. Many web service standards, semantics, and extensions are applied or modified in the OGSA.

ii) OGSA Interfaces

- ✓ The OGSA is centered on grid services. These services demand special well-defined application interfaces.
- ✓ These interfaces provide resource discovery, dynamic service creation, lifetime management,notification, and manageability. The conventions must address naming and upgradeability. Table 7.3 summarizes the interfaces proposed by the OGSA working group.
- ✓ While the OGSA defines a variety of behaviors and associated interfaces, all but one of these interfaces (the grid service) is optional.

Two key properties of a grid service are

- ✓ Transience
- ✓ Statefulness

These properties have significant implications regarding how a grid service is named, discovered, and managed.

- ✓ Being transient means the service can be created and destroyed dynamically
- ✓ Statefulness refers to the fact that one can distinguish one service instance from another.

Table 7.3 OGSA Grid Service Interfaces Developed by the OGSA Working Group

Port Type	Operation	Brief Description
Grid service	Find service data	Query a grid service instance, including the handle, reference, primary key, home handle map, interface information, and service-specific information. Extensible support for various query languages.
	Termination time	Set (and get) termination time for grid service instance.
	Destroy	Terminate grid service instance.
Notification source	Subscribe to notification topic	Subscribe to notifications of service events. Allow delivery via third-party messaging services.
Notification sink	Deliver notification	Carry out asynchronous delivery of notification messages.
Registry	Register service	Conduct soft-state registration of Grid Service Handles (GSHs).
	Unregister service	Unregister a GSH.
Factory	Create service	Create a new grid service instance.
Handle map	Find by handle	Return the Grid Service Reference (GSR) associated with the GSH.

iii) Grid Service Handle

- ✓ A GSH is a globally unique name that distinguishes a specific grid service instance from all others.
- ✓ The status of a grid service instance could be that it exists now or that it will exist in the future.
- ✓ These instances carry no protocol or instance-specific addresses or supported protocol bindings.
- ✓ Instead, these information items are encapsulated along with all other instance-specific information.
- ✓ In order to interact with a specific service instance, a single abstraction is defined as a GSR.
- ✓ Unlike a GSH, which is time-invariant, the GSR for an instance can change over the lifetime of the service.
- ✓ OGSA employs a "handle-resolution" mechanism for mapping from a GSH to a GSR. The GSH must be globally defined for a particular instance.
- ✓ However, the GSH may not always refer to the same network address. A service instance may be implemented in its own way,as long as it obeys the associated semantics. For example, the port type on which the service instance was implemented decides which operation to perform in Table 7.3.

iv) Grid Service Migration

- ✓ This is a mechanism for creating new services and specifying assertions regarding the lifetime of a service. The OGSA model defines a standard interface, known as a factor, to implement this reference.

- ✓ Any service that is created must address the former services as the reference of later services.The factory interface is labeled as a Create Service operation in Table 7.3
  - ✓ This creates a requested grid service with a specified interface and returns the GSH and initial GSR for the new service instance. It should also register the new service instance with a handle resolution service.
  - ✓ Each dynamically created grid service instance is associated with a specified lifetime.

v) OGSA Security Models

- ✓ The OGSA supports security enforcement at various levels, as shown in Figure 7.4.
- ✓ The grid works in a heterogeneous distributed environment, which is essentially open to the general public. We must be able to detect intrusions or stop viruses from spreading by implementing secure conversations, single logon, access control, and auditing for nonrepudiation.
- ✓ At the security policy and user levels, we want to apply a service or endpoint policy, resource mapping rules, authorized access of critical resources, and privacy protection.
- ✓ At the Public Key Infrastructure (PKI) service level, the OGSA demands security binding with the security protocol stack and bridging of certificate authorities (CAs), use of multiple trusted intermediaries, and so on.
- ✓ Trust models and secure logging are often practiced in grid platforms.

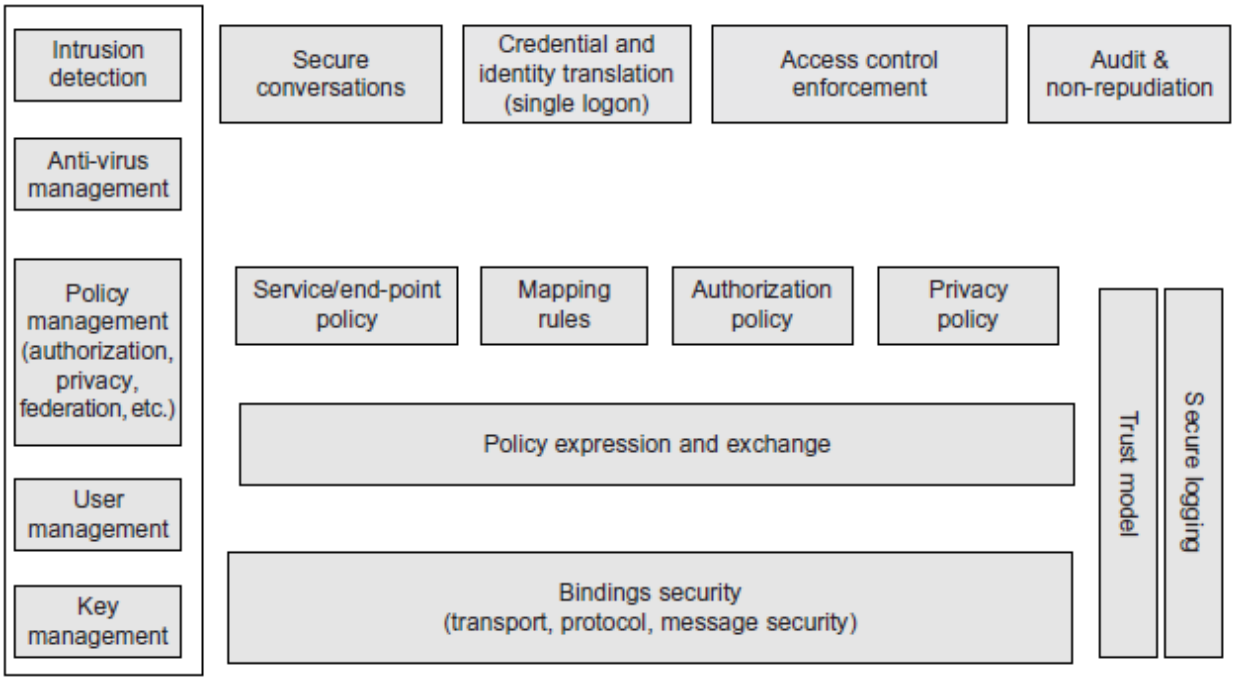


FIGURE 7.4

The OGSA security model implemented at various protection levels.

Example 7.3 Grid Service Migration Using GSH and GSR

Figure 7.3 shows how a service instance may migrate from one location to another during execution.

A GSH resolves to a different GSR for a migrated service instance before (on the left) and after (on the right) the migration at time T.

- ✓ The handle resolver simply returns different GSRs before and after the migration. The initial lifetime can be extended by a specified time period by explicitly requesting the client or another grid service acting on the client's behalf.
- ✓ If the time period expires without having received a reaffirmed interest from a client, the service instance can be terminated on its own and release the associated resources accordingly. The lifetime management enables robust termination and failure detection.

This is done by clearly defining the lifetime semantics of a service instance.

Similarly, a hosting environment is guaranteed to consume bounded resources under some system failures.

If the termination time of a service is reached, the hosting environment can reclaim all resources allocated.

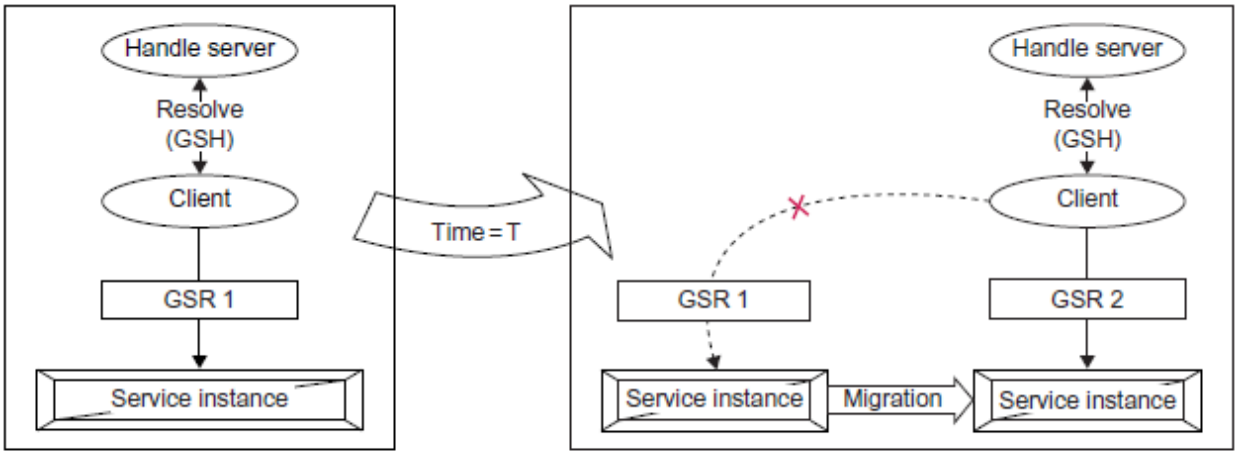


FIGURE 7.3

A GSH resolving to a different GSR for a migrated service instance before (shown on the left) and after (on the right) the migration at time T.

2.Explain the Motivations for standardization in OGSA

- ✓ The lack of standards has meant that companies, developers, and organizations have had to develop and support grid technology using proprietary techniques and solutions, hereby limiting its deployment potential.
- ✓ An effective grid relies on making use of computing power, whether via a LAN, over an extranet, or through the Internet. To use computing power efficiently, one needs to support a gamut of computing platforms; also; one needs a flexible mechanism for distributing and allocating the work to individual clients.
- ✓ The time is now ripe for grid standards. Telecommunications standards were developed (in critical-mass fashion) in the 1980s, Internet standards were developed (in critical-mass fashion) in the 1990s, and, hopefully, grid computing standards will developed (in critical-mass fashion) in the 2000s.

- ✓ A closed, proprietary environment limits the ease with which one can distribute work, who can become a service provider, who can be a service requester, and how one can find out about the available grid resources.
- ✓ The reader can grasp the limitations of a nonstandard approach by considering any of the better-known grid computing projects on the Internet.
- ✓ For example, consider distributed.net. This organization is a "loosely knit" group of computer users located all over the world, that takes up challenges and run projects that require a lot of computing power. It solves these by utilizing the combined idle processing cycles of their members' computers.
- ✓ To illustrate the point about standards, to become a service provider for the distributed.net grid, one must download a specific client that is capable of processing the work units from the corresponding servers.
- ✓ However, with a distributed.net client installed, one can only process work units supplied by distributed.net.
- ✓ Furthermore, distributed.net service providers can only process those work units supplied by distributed.net. For example, if distributed.net wanted to allow their service providers (those people with the distributed.net client installed) to process SETI@Home work units, it would be problematic.
- ✓ Distributed.net would have to re- deploy their service provider functionality. They would also have to redesign many of their discovery and distribution systems to allow different work units to be deployed to service providers, and they would need to update their statistical analysis on completed units to track it all properly.
- ✓ As this example illustrates, standards are critical to making the computing utility concept a reality. On the other hand, a corporate user just looking to secure better utilization of its platforms and internal resources could start with a vendor-based solution and then move up to a standards-based solution in due course.

Some specific areas where a lack of grid standards limit deployment are

Data management

- ✓ For a grid to work effectively, there is a need to store information and distribute it. Without a standardized method for describing the work and how it should be exchanged, one quickly encounters limits related to the flexibility and interoperability of the grid.

Dispatch management

- ✓ There are a number of approaches that can be used to handle brokering of work units and to distribute these work units to client resources.
- ✓ Again, not having a standard method for this restricts the service providers that can connect to the grid and accept units of work from the grid;this also restricts the ability of grid services users to submit work.

Information services

- ✓ Metadata6 about the grid service helps the system to distribute information. The metadata is used to identify requesters (grid users), providers, and their respective requirements and resource availability.

- ✓ Again, without a standard, one can only use specific software and solutions to support the grid applications.

Scheduling

- ✓ Work must be scheduled across the service providers to ensure they are kept busy. To accomplish this, information about remote loads must be collected and administered.
- ✓ A standardized method of describing the grid service enables grid implementations to specify how work is to be scheduled.

Security

- ✓ Without a standard for the security of a grid service and for the secure distribution of work units, one runs the risk of distributing information to the "wrong" clients. Although proprietary methods can provide a level of security, they limit accessibility.

Work unit management

- ✓ Grid services require management of the distribution of work units to ensure that the work is uniformly distributed over the service providers. Without a standard way of advertising and managing this process, efficiencies are degraded.
- ✓ Looking from the perspective of the grid applications developer, a closed environment is similarly problematic because to make use of the computing resources across a grid, the developer must utilize a specific tool kit or environment to build,distribute, and process work units.
- ✓ The closed environment limits the choice of grid-resident *platforms* on which work units can be executed and, at the same time, also limits how one uses and distributes work and/or requests to the grid.
- ✓ It also means that one cannot combine or adopt other grid solutions for use within an organization's grid without redeploying the grid software. The generic advantages of the standardized approach are well known, since they apply across any number of disciplines.
- ✓ In the context of grid computing, they all reduce to one basic advantage: the extension and expansion of the resources available to the user for grid computing. From an end user's perspective, standardization translates into the ability to purchase middleware and grid-enabled applications from a variety of suppliers in an off-the-shelf, shrink-wrapped fashion. Figure 4.4 depicts an example of the environment that one aims to achieve.

For example, standard APIs enable application *portability;* without standard APIs, application portability is difficult to accomplish (different platforms access protocols in different ways). Standards enable cross-site *interoperability;* without standard protocols, interoperability is difficult to achieve. Standards also enable the deployment of a *shared infrastructure* .Use of the OGSI standard, therefore, provides the following benefits .

Increased effective computing capacity

- ✓ When the resources utilize the same conventions, interfaces, and mechanisms, one can transparently switch jobs among grid systems, both from the perspective of the server as well as from the perspective of the client.

- ✓ allows grid users to use more capacity and allows clients a more extensive choice of projects that can be supported on the grid. Hence, with a gamut of platforms and environments supported, along with the ability to more easily publish the services available, there will be an increase in the effective computing capacity.

Interoperability of resources

- ✓ Grid systems can be more easily and efficiently developed and deployed when utilizing a variety of languages and a variety of platforms. For example, it is desirable to mix service-provider components, work-dispatch tracking systems, and systems management; this makes it easier to dispatch work to service providers and for service providers to support grid services.

Speed of application development

- ✓ Using middleware (and/or toolkits) based on a standard expedites the development of grid-oriented applications supporting a business environment. Rather than spending time developing communication and management systems to help support the grid system, the planner can, instead, spend time optimizing the business/algorithmic logic related to the processing the data.
- ✓ For useful applications to be developed, a rich set of grid services (the OGSA architected services) need to be implemented and delivered by both open source efforts (such as the Globus project) and by middleware software companies.
- ✓ In a way, OGSI and the extensions it provides for Web services are necessary but insufficient for the maturation of the service-oriented architecture; the next required step is that these standards be fully implemented and truly observed (in order to provide portability and interoperability)

Figure 4.5 depicts a simple environmentto put the network-based services in context.

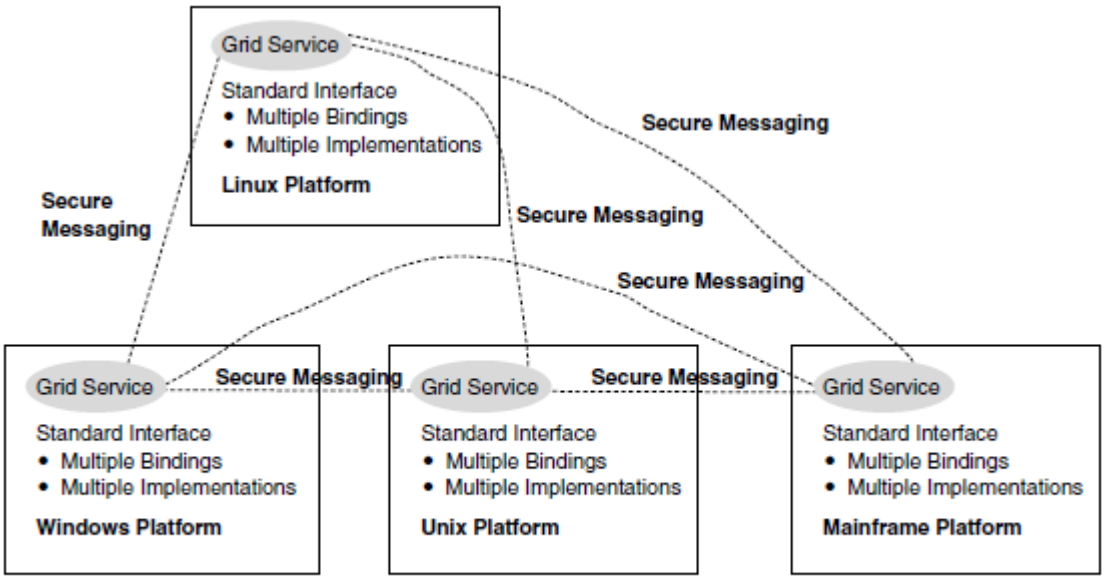


Figure 4.4 Example of a service-oriented architecture.

3. Explain functionality requirements of OGSA

- ✓ The development of the OGSA document has been based on a variety of use case scenarios. The use cases have not been defined with a view to expressing formal requirements (and do not contain the level of detail that would be required for formal requirements), but have provided useful input to the definition process.
- ✓ Analysis of the use cases, other input from OGSA-WG participants, and other studies of grid technology requirements lead the Working Group to identify important and broadly relevant characteristics of grid environments and applications, along with functionalities that appear to have general relevance to a variety of application scenarios.
  - ✓ Although this material does not represent a comprehensive or formal statement of functionality requirements from our use cases, it does provide useful input for subsequent development of OGSA functions.

The case scenarios that have been considered include

- ✓ National fusion collaboration
- ✓ IT infrastructure and management
- ✓ Commercial data centers
- ✓ Service-based distributed query processing
- ✓ Severe storm prediction
- ✓ Online media and entertainment

(i) Basic Functionality Requirements

The following basic functions are universally fundamental:

Discovery and brokering

- ✓ Mechanisms are required for discovering and/or allocating services, data, and resources with desired properties.
- ✓ For example, clients need to discover network services before they are used, service brokers need to discover hardware and software availability, and service brokers must identify codes and platforms suitable for execution requested by the client [55].

Metering and accounting

- ✓ Applications and schemas for metering, auditing, and billing for IT infrastructure and management use cases. The metering function records the usage and duration, especially metering the usage of licenses.
- ✓ The auditing function audits usage and application profiles on machines, and the billing function bills the user based on metering.

Data sharing

- ✓ Data sharing and data management are common as well as important grid applications. Mechanisms are required for accessing and managing data archives, for caching data and managing its consistency, and for indexing and discovering data and metadata.

Deployment

- ✓ Data is deployed to the hosting environment that will execute the job (or made available in or via a high-performance infrastructure). Also, applications (executable) are migrated to the computer that will execute them.

Virtual organizations (VOs)

- ✓ The need to support collaborative VOs introduces a need for mechanisms to support VO creation and management, including group membership services. For the commercial data center use case ,the grid creates a VO in a data center that provides IT resources to the job upon the customer's job request.
- ✓ Depending on the customer's request,the grid will negotiate with another grid on a remote commercial data center and create a VO across the commercial data centers. Such a VO can be used to achieve the necessary scalability and availability.

Monitoring

- ✓ A global, cross-organizational view of resources and assets for project and fiscal planning, troubleshooting, and other purposes. The users want to monitor their applications running on the grid. Also, the resource or service owners need to surface certain states so that the user of those resources or services may manage the usage using the state information.

Policy

- ✓ An error and event policy guides self-controlling management, including failover and provisioning. It is important to be able to represent policy at multiple stages in hierarchical systems, with the goal of automating the enforcement of policies that might otherwise be implemented as organizational processes or managed manually.
- ✓ There may be policies at every level of the infrastructure: from low-level policies that govern how the resources are monitored and managed, to high-level policies that govern how business process such as billing are managed. High-level policies are sometimes decomposable into lower-level policies.

(ii) Security Requirements

Grids also introduce a rich set of security requirements; some of these requirements are:

Multiple security infrastructures

- ✓ Distributed operation implies a need to interoperate with and manage multiple security infrastructures. For example, for a commercial data center application, isolation of customers in the same commercial data center is a crucial requirement; the grid should provide not only access control but also performance isolation.
- ✓ For another example, for an online media and entertainment use case, proper isolation between content offerings must be ensured; this level of isolation has to be ensured by the security of the infrastructure.

Perimeter security solutions

- ✓ Many use cases require applications to be deployed on the other side of firewalls from the intended user clients.

- ✓ Intergrid collaboration often requires crossing institutional firewalls. OGSA needs standard, secure mechanisms that can be deployed to protect institutions while also enabling cross-firewall interaction.

Authentication, Authorization, and Accounting

- ✓ Obtaining application programs and deploying them into a grid system may require authentication/authorization.
- ✓ In the commercial data center use case, the commercial data center authenticates the customer and authorizes the submitted request when the customer submits a job request. The commercial data center also identifies his/her policies (including but not limited to SLA, security, scheduling, and brokering policies).

Encryption

- ✓ The IT infrastructure and management use case requires encrypting of the communications, at least of the payload.

Application and Network-Level Firewalls

- ✓ This is a long-standing problem; it is made particularly difficult by the many different policies one is dealing with and the particularly harsh restrictions at international sites.

Certification

- ✓ A trusted party certifies that a particular service has certain semantic behavior. For example, a company could establish a policy of only using e-commerce services certified by Yahoo.

(iii) Resource Management Requirements

Resource management is another multilevel requirement, encompassing SLA negotiation, provisioning, and scheduling for a variety of resource types and activities:

Provisioning

- ✓ Computer processors, applications, licenses, storage, networks, and instruments are all grid resources that require provisioning. OGSA needs a framework that allows resource provisioning to be done in a uniform, consistent manner.

Resource virtualization

- ✓ Dynamic provisioning implies a need for resource virtualization mechanisms that allow resources to be transitioned flexibly to different tasks as required; for example, when bringing more Web servers on line as demand exceeds a threshold.

Optimization of resource usage

- ✓ while meeting cost targets (i.e., dealing with finite resources). Mechanisms to manage conflicting demands from various organizations, groups, projects, and users and implement a fair sharing of resources and access to the grid.

Transport management

- ✓ For applications that require some form of real-ticheduling, it can be important to be able to schedule or provision bandwidth dynamically for data transfers or in support of the other data sharing applications.
- ✓ In many (if not all) commercial applications, reliable transport management is essential to obtain the end-to-end QoS required by the application.

Access

- ✓ Usage models that provide for both batch and interactive access to resources.

Management and monitoring

- ✓ Support for the management and monitoring of resource usage and the detection of SLA or contract violations by all relevant parties.
- ✓ Also, conflict management is necessary; it resolves conflicts between management disciplines that may differ in their optimization objectives (availability goals versus performance goals, for example).

Processor scavenging is an important tool that allows an enterprise or VO to use to aggregate computing power that would otherwise go to waste.

- ✓ How can OGSA provide service infrastructure that will allow the creation of applications

that use scavenged cycles? For example, consider a collection of desktop computers running software that supports integration into processing and/or storage pools managed via systems such as Condor, Entropia, and United Devices. Issues here include maximizing security in the absence of strong trust.

Scheduling of service tasks

- ✓ Long recognized as an important capability for any information processing system, scheduling becomes extremely important and difficult for distributed grid systems. In general, dynamic scheduling is an essential component. Computer resources must be provisioned on-demand to satisfy the need to complete a forecast on time.

Load balancing

- ✓ In many applications, it is necessary to make sure make sure deadlines are met or resources are used uniformly. These are both forms of load balancing that must be made possible by the underlying infrastructure.
- ✓ For example, for the commercial data center use case, monitoring the job performance and adjusting allocated resources to match the load and fairly distributing end users' requests to all the resources are necessary.
- ✓ For the online media and entertainment use case, the amount of workload is a direct result of how many concurrent online game players are being hosted on a game server.
- ✓ If the game server (server A) is responsible for a 20 square mile area in the game world, and a battle occurred in that area, many players will rush to that area, causing workload on that server to increase. As players enter that area and leave other areas, other servers' workloads will decrease.

- ✓ Hence, when the workload of server A gets above certain threshold, a load balancing routine needs to be triggered to rebalance the resources (i.e., servers). That is, workloads must be redistributed across servers with idle capacity.

Advanced reservation

- ✓ This functionality may be required in order to execute the application on reserved resources. For example, for the commercial data center use case, the grid decides when to start the request processing based on the customer's request.
- ✓ It interprets the job specification description language in which the request is written and it checks to see if the customer has the right to perform the request.

Notification and messaging

- ✓ Notification and messaging are critical in most dynamic scientific problems. Notification and messaging are event driven.

Logging

- ✓ It may be desirable to log processes such as obtaining/deploying application

programs because, for example, the information might be used for accounting. This functionality is represented as "metering and accounting."

Workflow management

- ✓ Many applications can be wrapped in scripts or processes that require licenses and other resources from multiple sources. Application coordinate using the file system based on events.

Pricing

- ✓ Mechanisms for determining how to render appropriate bills to users of a grid.

(iv) System Properties Requirements

A number of grid-related capabilities can be thought of as desirable system properties rather than functions:

Fault tolerance

- ✓ Support is required for failover, load redistribution, and other techniques used to achieve fault tolerance. Fault tolerance is particularly im- portant for long running queries that can potentially return large amounts of data, for dynamic scientific applications, and for commercial data center applications.

Disaster recovery

- ✓ Disaster recovery is a critical capability for complex distributed grid infrastructures. For distributed systems, failure must be considered one of the natural behaviors and disaster recovery mechanisms must be considered an essential component of the design. Autonomous system principles must be embraced as one designs grid applications and should be reflected in OGSA.
- ✓ In case of commercial data center applications if the data center becomes unavailable due to a disaster such as an earthquake or fire, the remote backup data center needs to take over the application systems.

Self-healing capabilities of resources, services and systems are required. Significant manual effort should not be required to monitor, diagnose, and repair faults.

- ✓ There is a need for the ability to integrate intelligent self-aware hardware such as disks, networking devices, and so on.

Strong monitoring for defects, intrusions, and other problems. Ability to migrate attacks away from critical areas.

Legacy application management

- ✓ Legacy applications are those that cannot be changed, but they are too valuable to give up or to complex to rewrite. Grid infrastructure has to be built around them so that they can continue to be used.

Administration

- ✓ Be able to "codify" and "automate" the normal practices used to administer the environment. The goal is that systems should be able to selforganize and self-describe to manage low-level configuration details based on higher-level configurations and management policies specified by administrators.

Agreement-based interaction

- ✓ Some initiatives require agreement-based interactions capable of specifying and enacting agreements between clients and servers (not necessarily human) and then composing those agreements into higher-level end-user structures.

Grouping/aggregation of services

The ability to instantiate (compose) services using some set of existing services is a key requirement. There are two main types of composition techniques:

- Selection
- Aggregation

- ✓ Selection involves choosing to use a particular service among many services with the same operational interface.
- ✓ Aggregation involves orchestrating a functional flow (workflow) between services. For example, the output of an accounting service is fed into the rating service to produce billing records.
- ✓ One other basic function required for aggregation services is to transform the syntax and/or semantics of data or interfaces.

(v) Other Functionality Requirements

- ✓ Although some use cases involve highly constrained environments (that may well motivate specialized OGSA profiles), it is clear that in general grid environments tend to be heterogeneous and distributed:

Platforms

- ✓ The platforms themselves are heterogeneous, including a variety of operating systems (Unixes, Linux, Windows, and, presumably, embedded systems), hosting environments (J2EE, .NET, others), and devices (computers, instruments, sensors, storage systems, databases, networks, etc.).

Mechanisms

- ✓ Grid software can need to interoperate with a variety of distinct implementation mechanisms for core functions such as security.

Administrative environments

- ✓ Geographically distributed environments often feature varied usage, management, and administration policies (including policies applied by legislation) that need to be honored and managed.
- ✓ A wide variety of application structures are encountered and must be supported by other system components, including the following:
  - *single -process*
  - *multiprocess*

(both local and distributed) applications covering a wide range of resource requirements.

Flows, that is, multiple interacting applications that can be treated as a single transient service instance working on behalf of a client or set of clients.

Workloads comprising potentially large numbers of applications with a number of characteristics just listed.

4) What is OGSA/OGSI? and explain A practical view of this method in detail .

- ✓ The OGSA standard defines what grid services are, what they should be capable of, and what technologies they are based on. OGSA, however, does not go into specifics of the technicalities of the specification; instead, the aim is to help classify what is and is not a grid system .
- ✓ It is called an *architecture* because it is mainly about describing and building a well-defined set of interfaces from which systems can be built, based on open standards such as WSDL.
  - Manage resources across distributed heterogeneous platforms.
  - Support QoS-oriented Service Level Agreements (SLAs). The topology ofgrids is often complex; the interactions between/among grid resources are almost invariably dynamic. It is critical that the grid provide robust services such as authorization, access control, and delegation.
  - Provide a common base for autonomic management. A grid can contain a plethora of resources, along with an abundance of combinations of resource configurations, conceivable resource-to-resource interactions, and a litany of changing state and failure modes. Intelligent self-regulation and autonomic management of these resources is highly desirable.
  - Define open, published interfaces and protocols for the interoperability of diverse resources. OGSA is an open standard managed by a standards body.

- Exploit industry standard integration technologies and leverage existing solutions where appropriate. The foundation of OGSA is rooted in Web services, for example, SOAP and WSDL, are a major part of this specification.

✓ OGSA's companion OGSI document consists of specifications on how work is managed, distributed, and how service providers and grid services are described. The Web services component is utilized to facilitate the distribution and the management of work across the grid.

✓ Because Web services offer a transparent method of communication between hosts (irrespective of the underlying language or platform), one can utilize these services to transfer work, to describe resources and configuration information, and to communicate and dispatch grid information.

✓ WSDL provides a simple method of describing and advertising the Web services that support the grid's application .

✓ Summarizing these observations, OGSA is the blueprint, OGSI is a technical specification, and Globus Toolkit is an implementation of the framework.

✓ OGSA describes and defines a Web-services-based architecture composed of a set of interfaces and their corresponding behaviors to facilitate distributed resource sharing and accessing in heterogeneous dynamic environments.

✓ A set of services based on open and stable protocols can hide the complexity of service requests by users or by other elements of a grid. Grid services enable *virtualization;* virtualization, in turn, can transform computing into a ubiquitous infrastructure that is more akin to an electric or water utility, as envisioned in the opening paragraph of this chapter .

✓ OGSA relies on the definition of grid services in WSDL, which, as noted, defines, for this context, the *operations names, parameters,* and their *types* for grid service access.

✓ Based on the OGSI specification, a grid service instance is a Web service that conforms to a set of conventions expressed by the WSDL as service interfaces, extensions, and behaviors [49]. Because the OGSI standard is based on a number of existing standards (XML, Web services, WSDL), it is an open and standards-based solution. This implies that, in the future, grid services can be built that are compatible with the OGSI standard, even though they may be based on a variety of different languages and platforms .

✓ Specifically, the grid service interface (see Table 4.6 and [84, 114]) is described by WSDL, which defines how to use the service. A new tag, gsdl, has been added to the WSDL document for grid service description. The UDDI registry and WSIL document are used to locate grid services.

✓ The transport protocol SOAP is used to connect data and applications for accessing grid services. All services adhere to specified grid service interfaces and behaviors. Grid service interfaces correspond to portTypes in WSDL used in current Web services solutions .

✓ The interfaces of grid services address discovery, dynamic service-instance creation, lifetime management, notification, and manageability; the conventions of grid services address naming and upgrading issues.

✓ The standard interface of a grid service includes multiple bindings and implementations ("implementations" include Java and procedural/object-oriented computer programming languages).

✓ Grid services, such as the ones just cited, can, therefore, be deployed on different hosting environments, even different operating systems. OGSA also provides a grid security mechanism to ensure that all the communications between services are secure.

Table 4.6 Proposed OGSA grid service interfaces*

Port type	Operation	Description
GridService	FindServiceData	Query a variety of information about the grid service instance, including basic introspection information (handle, reference, primary key, home handle map: terms to be defined), richer per-interface information, and service-specific information (e.g., service instances known to a registry). Extensible support for various query languages.
	SetTermination Time	Set (and get) termination time for grid service instance
	Destroy	Terminate grid service instance.
Notification-Source	SubscribeTo-NotificationTopic	Subscribe to notifications of service-related events, based on message type and interest statement. Allows for delivery via third-party messaging services.
Notification-Sink	Deliver Notification	Carry out asynchronous delivery of notification messages.
Registry	RegisterService UnregisterService	Conduct soft-state registration of grid service handles. Deregister a grid service handle.
Factory	CreateService	Create new grid service instance.
Handle Map	FindByHandle	Return grid service reference currently associated with supplied grid service handle.

*Interfaces for authorization, policy management, manageability, and likely other purposes remain to be defined.

- ✓ The definition of standard service interfaces and the identification of the protocol(s) are addressed in current OGSA specifications .
- ✓ Service capabilities (that is, the services offered by a particular company or organization)are widely used in existing Web services solutions.
- ✓ Likewise, grid servicesare characterized by the capabilities they afford. A grid service capability could be comprised of computational resources, storage resources, networks, programs, databases, and so on.
- ✓ A grid service implements one or more interfaces, where each interface defines a set of method operations that is invoked by constructing a method call through, method signature adaptation using SOAP .
- ✓ Like the majority of Web services, OGSI services use WSDL as a service description mechanism.

There are two fundamental requirements for describing Web services based on the OGSI

1.The ability to describe interface inheritance—a basic concept with most of the distributed object systems.

2. The ability to describe additional information elements with the interface definitions.

- ✓ The WSDL 1.1 specification lacks the two abilities just enumerated in its definition of portType. Hence, at press time, OGSI was utilizing an extension WSDL called GWSDL (Grid-extension to WSDL); however, there was a consensus among OGSI Work Group members to eventually use the WSDL 1.2 specification (when WSDL 1.2 reaches the recommendation stage, it may eliminate the need to use GWSDL).
- ✓ The WSDL 1.2 working group has agreed to support the just-listed features through portType inheritance and an open content model for portTypes. As an interim decision, OGSI has developed a new schema for portType definition (extended from normal WSDL 1.1 schema portType Type) under the new GWSDL namespace definition.
- ✓ Another noteworthy aspect of OGSI is the naming convention adopted for the portType operations and the lack of support for operator overloading.
- ✓ In these cases, OGSI follows the same conventions as described in the suggested WSDL 1.2 specification .
- ✓ From a near-term implementation perspective, the Globus Toolkit is the primary solution that supports the new standards of the OGSA/OGSI system (we cover some details of this in Chapter 6.) IBM is also deploying a version of WebSphere, the Web development platform, that makes use of grid technology to help spread the load of requests for a Web application .

Grid Program Execution Services

- ✓ Grid program execution services are depicted in Figure 4.11. Mechanisms for job scheduling and workload management implemented as part of this class of services are central to grid computing and the ability to virtualize processing resources.
- ✓ Although OGSI and core grid services are generally applicable to any distributed computing system, the grid program execution service class is unique to the grid model of distributed task execution that supports high-performance computing, parallelism, and distributed collaboration

Grid Data Services

- ✓ Grid data services are also depicted in Figure 4.11. These interfaces support the concept of data virtualization and provide mechanisms related to distributed access to information of many types including databases, files, documents, content stores, and application-generated streams.
- ✓ Services that comprise the grid data services class complement the computing virtualization conventions specified by program execution services (OGSA placing data resources on an equivalent level with computing resources).
- ✓ Grid data services will exploit and vir- tualize data using placement methods like data replication, caching, and high-performance data movement to give applications required QoS access across the distributed grid.
- ✓ Methods for federating multiple disparate, distributed data sources may also provide integration of data stored under differing schemas such as files and relational databases .

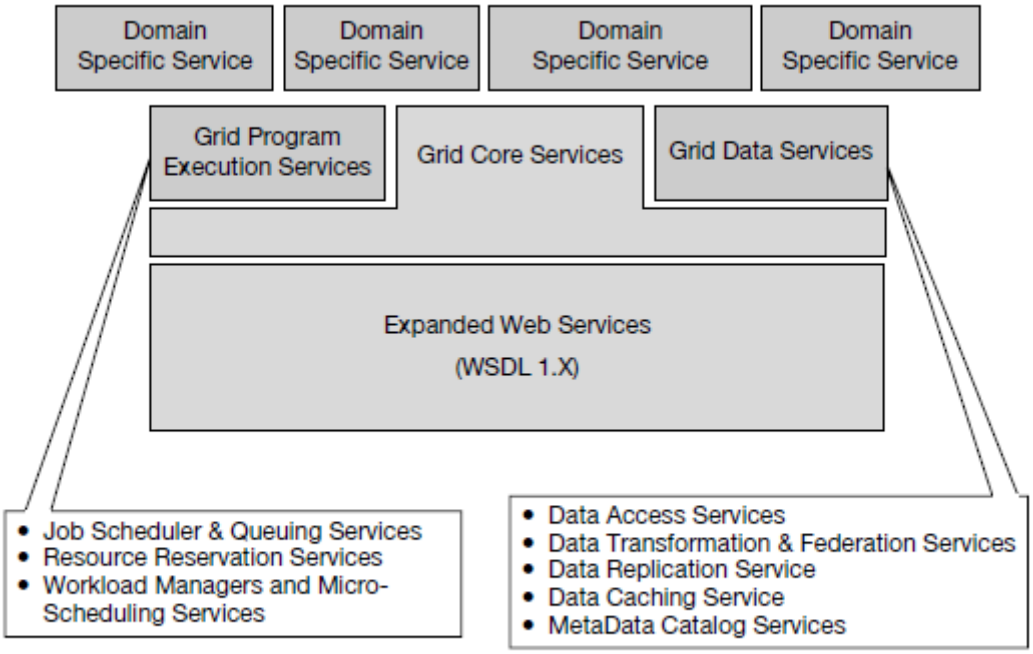


Figure 4.11 Grid program execution services and grid data services.

5) WHAT IS OGSA/OGSI? A MORE DETAILED VIEW EXPLAIN IN DETAIL

(i) Introduction
(ii) Setting the Context
(iii) The Grid Service
(iv) WSDL Extensions and Conventions
(v) Service Data
(vi) Core Grid Service Properties

(i) Introduction

- ✓ The OGSA integrates key grid technologies (including the Globus Toolkit) with Web services mechanisms to create a distributed system framework based on the OGSI.
- ✓ A *grid service instance* is a (potentially transient) service that conforms to a set of conventions, expressed as WSDL interfaces, extensions, and behaviors, for such purposes as lifetime management, discovery of characteristics, and notification.
- ✓ Grid services provide for the controlled management of the distributed and often long-lived state that is commonly required in sophisticated distributed applications. OGSI also introduces standard factory and registration interfaces for creating and discovering grid services.

OGSI defines a component model that extends WSDL and XML schema definition to incorporate the concepts of

- Stateful Web services
- Extension of Web services interfaces
- Asynchronous notification of state change
- References to instances of services
- Collections of service instances
- Service state data that augment the constraint capabilities of XML schema definition

✓ The OGSI specification (V1.0 at press time) defines the minimal, integrated set of extensions and interfaces necessary to support definition of the services that will compose OGSA. The OGSI V1.0 specification proposes detailed specifications for the conventions that govern how clients create, discover, and interact with a grid service instance.

That is, it specifies

(1) How grid service instances are named and referenced;
(2) The base, common interfaces (and associated behaviors) that all grid services implement;
(3) The additional (optional) interfaces and behaviors associated with factories and service groups.

✓ The specification does *not* address how grid services are created, managed, and destroyed within any particular hosting environment. Thus, services that conform to the OGSI specification are not necessarily portable to various hosting environments, but any client program that follows the conventions can invoke any grid service instance conforming to the OGSI specification (of course, subject to policy and compatible protocol bindings).

✓ The term *hosting environment* is used in the OGSI specification to denote the server in which one or more grid service implementations run. Such servers are typically language or platform specific; examples include native Unix and Windows processes, J2EE application servers, and Microsoft .NET.

(i) Setting the Context

✓ GGF calls OGSI the "base for OGSA." Specifically, there is a relationship between OGSI and distributed object systems and also a relationship between OGSI and the existing (and evolving) Web services framework.

✓ One needs to examine both the client-side programming patterns for grid services and a conceptual hosting environment for grid services. The patterns described in this section are enabled but not *required* by OGSI.

a)Relationship to Distributed Object Systems

✓ A given grid service implementation is an addressable and potentially stateful instance that implements one or more interfaces described by WSDL portTypes. Grid service factories can be used to create instances implementing a given set of portType(s).

✓ Each grid service instance has a notion of identity with respect to the other instances in the distributed grid.

✓ Each instance can be characterized as state coupled with behavior published through type-specific operations. The architecture also supports introspection in that a client application can

ask a grid service instance to return information describing itself, such as the collection of portTypes that it implements.

- ✓ Grid service instances are made accessible to (potentially remote) client applications through the use of a grid service handle and a grid service reference (GSR). These constructs are basically network-wide pointers to specific grid service instances hosted in (potentially remote) execution environments.
- ✓ A client application can use a grid service reference to send requests, represented by the operations defined in the portType(s) of the target service description directly to the specific instance at the specified network-attached service endpoint identified by the grid service reference.
- ✓ In many situations, client stubs and helper classes isolate application programmers from the details of using grid service references. Some client-side infrastructure software assumes responsibility for directing an operation to a specific instance that the GSR identifies.
- ✓ The characteristics introduced above (stateful instances, typed interfaces, global names, etc.) are frequently also cited as fundamental characteristics of *distributed object-based systems.*
- ✓ There are, however, also various other aspects of distributed object models (as traditionally defined) that are specifically not required or prescribed by OGSI. For this reason, OGSI does not adopt the term distributed object model or distributed object system when describing these concepts, but instead uses the term "open grid services infrastructure," thus emphasizing the connections that are established with both Web services and grid technologies.
- ✓ Among the object-related issues that are not addressed within OGSI are implementation inheritance, service instance mobility, development approach, and hosting technology.
- ✓ The grid service specification does not require, nor does it prevent, implementations based upon object technologies that support inheritance at either the interface or the implementation level.
- ✓ There is no requirement in the architecture to expose the notion of implementation inheritance either at the client side or at the service provider side of the usage contract. In addition, the grid service specification does not prescribe, dictate, or prevent the use of any particular development approach or hosting technology for grid service instances.
- ✓ Grid service providers are free to implement the semantic contract of the service description in any technology and hosting architecture of their choosing.
- ✓ OGSI envisions implementations in J2EE, .NET, traditional commercial transaction management servers, traditional procedural Unix servers, and so forth. It also envisions service implementations in a wide variety of both object-oriented and nonobject-oriented programming languages.

b) Client-Side Programming Patterns

- ✓ Another important issue is how OGSI interfaces are likely to be invoked from client applications. OGSI exploits an important component of the Web services framework: the use of WSDL to describe multiple protocol bindings, encoding styles, messaging styles (RPC versus document oriented), and so on, for a given Web service. The Web Services Invocation Framework (WSIF) and Java API for XML RPC (JAX-RPC) are among the many examples of infrastructure software that provide this capability.
- ✓ Figure 4.14 depicts a possible (but not required) client-side architecture for OGSI. In this approach, a clear separation exists between the client application and the client-side

representation of the Web service (proxy), including components for marshaling the invocation of a Web service over a chosen binding.

- ✓ In particular, the client application is insulated from the details of the Web service invocation by a higher-level abstraction: the client-side interface.

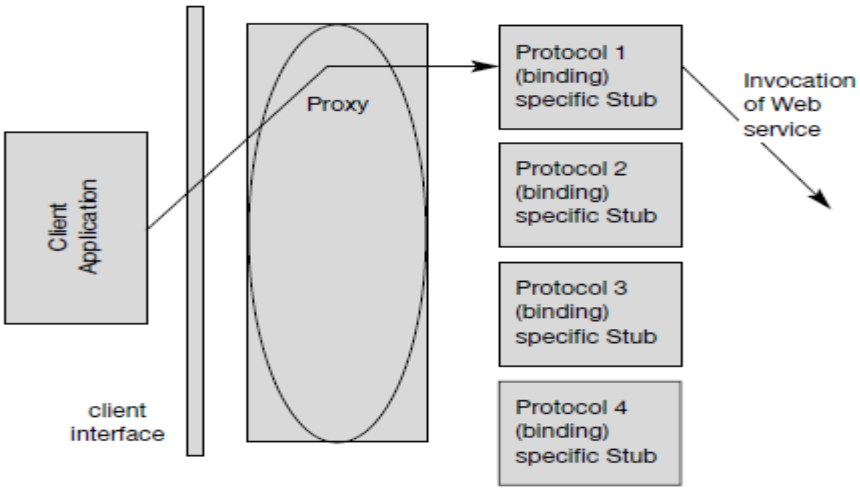


Figure 4.14 Possible client-side runtime architecture.

- ✓ Various tools can take the WSDL description of the Web service and generate interface definitions in a wide range of programming-language-specific constructs (e.g., Java interfaces and C#).
- ✓ This interface is a front end to specific parameter marshaling and message routing that can incorporate various binding options provided by the WSDL. Further, this approach allows certain efficiencies, for example,
- ✓ Detecting that the client and the Web service exist on the same network host, therefore avoiding the overhead of preparing for and executing the invocation using network protocols.
- ✓ Within the client application runtime, a *proxy* provides a client-side representation of remote service instance's interface. Proxy behaviors specific to a particular encoding and network protocol (*binding*, in Web services terminology) are encapsulated in a *protocol-specific (binding-specific) stub*.
- ✓ Details related to the binding specific access to the grid service instance, such as correct formatting and authentication mechanics, happen here; thus, the application is not required to handle these details itself.
- ✓ It is possible, but not recommended, for developers to build customized code that directly couples client applications to fixed bindings of a particular grid service instance. Although certain circumstances demand potential efficiencies gained by this style of customization, this approach introduces significant inflexibility into a system and therefore should only be used under extraordinary circumstances.
- ✓ The developers of the OGSI specification expect the stub and client-side infrastructure model that we describe to be a common approach to enabling client access to grid services.

- ✓ This includes both application-specific services and common infrastructure services that are defined by OGSA. Thus, for most software developers using grid services, the infrastructure and application-level services appear in the form of a class library or programming language interface that is natural to the caller. WSDL and the GWSDL extensions provide support for enabling heterogeneous tools and enabling infrastructure software.

C) Client Use of Grid Service Handles and References

- ✓ As noted, a client gains access to a grid service instance through grid service handles and grid service references. A grid service handle (GSH) can be thought of as a permanent network pointer to a particular grid service instance.
- ✓ The GSH does not provide sufficient information to allow a client to access the service instance; the client needs to "resolve"a GSH into a grid service reference (GSR). The GSR contains all the necessary information to access the service instance.
- ✓ The GSR is not a "permanent" network pointer to the grid service instance because a GSR may become invalid for various reasons;for example, the grid service instance may be moved to a different server.
- ✓ OGSI provides a mechanism, the HandleResolver to support client resolution of a grid service handle into a grid service reference. Figure 4.15 shows a client application that needs to resolve a GSH into a GSR.

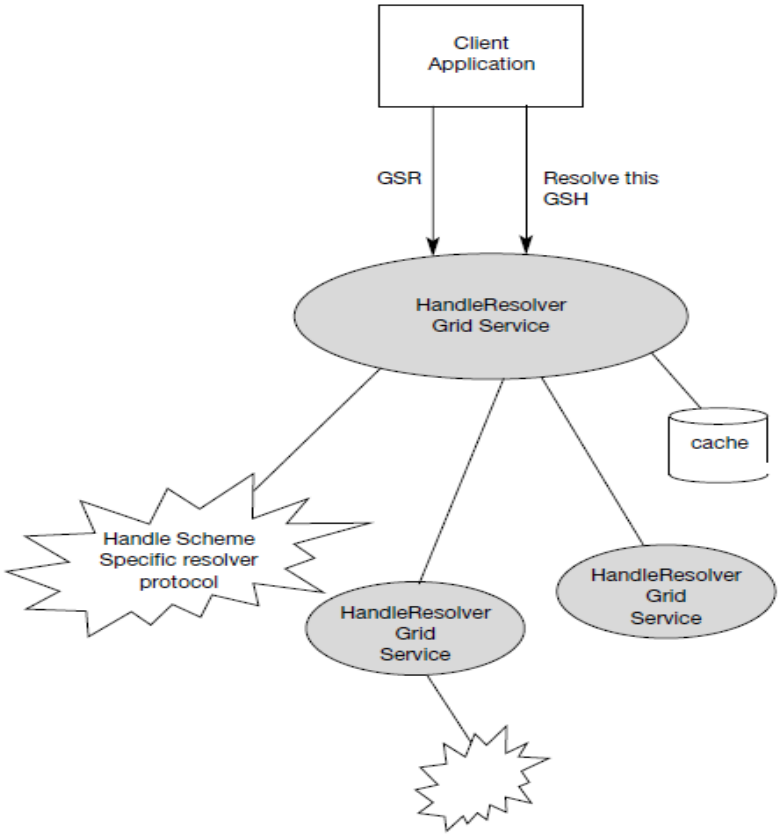


Figure 4.15 Resolving a GSH.

- ✓ The client resolves a GSH into a GSR by invoking a HandleResolver grid service instance identified by some out-of-band mechanism. The HandleResolver can use various means to do the resolution; some of these means are depicted in Figure 4.15.
- ✓ The HandleResolver may have the GSR stored in a local cache. The HandleResolver may need to invoke another HandleResolver to resolve the GSH.
- ✓ The HandleResolver may use a handle resolution protocol, specified by the particular kind (or scheme) of the GSH to resolve to a GSR. The HandleResolver protocol is specific to the kind of GSH being resolved.
- ✓ For example, one kind of handle may suggest the use of HTTP GET to a URL encoded in the GSH in order to resolve to a GSR.

d) Relationship to Hosting Environment

- ✓ OGSI does not dictate a particular service-provider-side implementation architecture. A variety of approaches are possible, ranging from implementing the grid service instance directly as an operating system process to a sophisticated server-side component model such as J2EE.
- ✓ In the former case, most or even all support for standard grid service behaviors (invocation, lifetime management, registration, etc.) is encapsulated within the user process; for example, via linking with a standard library.
- ✓ In the latter case, many of these behaviors are supported by the hosting environment.
- ✓ Figure 4.16 illustrates these differences by showing two different approaches to the implementation of argument demarshaling functions.
- ✓ One can assume that, as is the case for many grid services, the invocation message is received at a network protocol termination point (e.g., an HTTP servlet engine) that converts the data in the invocation message into a format consumable by the hosting environment.
- ✓ The top part of Figure 4.16 illustrates two grid service instances (the oval) associated with container-managed components (e.g., EJBs within a J2EE container).
- ✓ Here,the message is dispatched to these components, with the container frequently providing facilities for demarshaling and decoding the incoming message from a format (such as an XML/SOAP message) into an invocation of the component in native programming language.
- ✓ In some circumstances (the oval), the entire behavior of a grid service instance is completely encapsulated within the component.

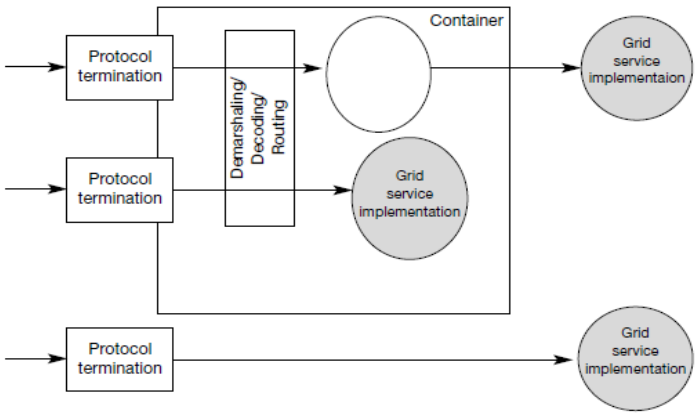


Figure 4.16 Two approaches to the implementation of argument demarshaling functions in a grid service hosting environment.

- ✓ In other cases (the oval), a component will collaborate with other server-side executables, perhaps through an adapter layer, to complete the implementation of the grid service behavior.
- ✓ The bottom part of Figure 4.16 depicts another scenario wherein the entire behavior of the grid service instance, including the demarshaling/ decoding of the network message, has been encapsulated within a single executable.
- ✓ Although this approach may have some efficiency advantages, it provides little opportunity for reuse of functionality between grid service implementations. A container implementation may provide a range of functionality beyond simple argument demarshaling.
  - ✓ For example, the container implementation may provide lifetime management functions, automatic support for authorization and authentication,request logging, intercepting lifetime management functions, and terminating service instances when a service lifetime expires or an explicit destruction request is received. Thus, one avoids the need to reimplement these common behaviors in different grid service implementations.

(ii) The Grid Service

- ✓ The purpose of the OGSI document is to specify the (standardized) interfaces and behaviors that define a *grid service.* In brief, a grid service is a WSDL-defined service that conforms to a set of conventions relating to its interface definitions and behaviors.
  Thus, every grid service is a Web service, though the converse of this statement is not true.

The OGSI document expands upon this brief statement by

- Introducing a set of WSDL conventions that one uses in the grid service specification; these conventions have been incorporated in WSDL 1.2 .
- Defining *service data* that provide a standard way for representing and querying metadata and state data from a service instance

Introducing a series of core properties of grid service, including:

- Defining grid service description and grid service instance, as organizing principles for their extension and their use
- Defining how OGSI models time
- Defining the grid service handle and grid service reference constructs that are used to refer to grid service instances
- Defining a common approach for conveying fault information from operations.This approach defines a base XML schema definition and associated semantics for WSDL fault messages to support a common interpretation; the approach simply defines the base format for fault messages, without modifying the WSDL fault message model.
- Defining the life cycle of a grid service instance

(iii)WSDL Extensions and Conventions

- ✓ As should be clear by now, OGSI is based on Web services; in particular, it uses WSDL as the mechanism to describe the public interfaces of grid services.
- ✓ However, WSDL 1.1 is deficient in two critical areas: lack of interface (portType) extension and the inability to describe additional information elements on a portType (lack of open content).

- ✓ These deficiencies have been addressed by the W3C Web Services Description Working Group.Because WSDL 1.2 is a "work in progress," OGSI cannot directly incorporate the entire WSDL 1.2 body of work.
- ✓ Instead,OGSI defines an extension to WSDL 1.1, isolated to the wsdl:portType element,which provides the minimal required extensions to WSDL 1.1.
- ✓ These extensions to WSDL 1.1 match equivalent functionality agreed to by the W3C Web Services Description Working Group.
- ✓ Once WSDL 1.2 [150] is published as a recommendation by the W3C, the Global Grid Forum is committed to defining a follow-on version of OGSI that exploits WSDL 1.2, and to defining a translation from this OGSI v1.0 extension to WSDL 1.2.

(iv) Service Data

- ✓ The approach to *stateful* Web services introduced in OGSI identified the need for a common mechanism to expose a service instance's state data to service requestors for query, update, and change notification.
- ✓ Since this concept is applicable to any Web service including those used outside the context of grid applications, one can propose a common approach to exposing Web service state data called "serviceData."
- ✓ The GGF is endeavoring to introduce this concept to the broader Web services community.In order to provide a complete description of the interface of a stateful Web service (i.e., a *grid service*), it is necessary to describe the elements of its state that are externally observable.
- ✓ By externally observable, one means that the state of the service instance is exposed to clients making use of the declared service interface,where those clients are outside of what would be considered the internal implementation of the service instance itself.
- ✓ The need to declare service data as part of the service's external interface is roughly equivalent to the idea of declaring attributes as part of an object-oriented interface described in an object-oriented interface-definition language.
- ✓ Service data can be exposed for read, update, or subscription purposes. Since WSDL defines operations and messages for portTypes, the declared state of a service must be externally accessed only through service operations defined as part of the service interface.
- ✓ To avoid the need to define serviceData-specific operations for each serviceData element, the grid service portType provides base operations for manipulating serviceData elements by name.

a) Motivation and Comparison to JavaBean Properties

- ✓ The OGSI specification introduces the serviceData concept to provide a flexible, properties-style approach to accessing state data of a Web service.
- ✓ The serviceData concept is similar to the notion of a public instance variable or field in object-oriented programming languages such as Java, Smalltalk, and C++. ServiceData is similar to JavaBean™ properties.
- ✓ The JavaBean model defines conventions for method signatures (getXXX/setXXX) to access properties, and helper classes (BeanInfo) to document properties. The OGSI model uses the serviceData elements and XML schema types to achieve a similar result.

- ✓ The OGSI specification has chosen not to require getXXX and setXXX WSDL operations for each serviceData element, although service implementers may choose to define such safe get and set operations themselves.
- ✓ Instead, OGSI defines extensible operations for querying (get), updating (set), and subscribing to notification of changes in serviceData elements. Simple expressions are required by OGSI to be supported by these operations, which allows for access to serviceData elements by their names, relative to a service instance.
- ✓ This by-name approach gives functionality roughly equivalent to the getXXX and setXXX approach familiar to JavaBean and Enterprise JavaBean programmers. However, these OGSI operations may be extended by other service interfaces to support richer query, update, and subscription semantics, such as complex queries that span multiple serviceData elements in a service instance.
- ✓ The serviceDataName element in a GridService portType definition corresponds to the BeanInfo class in JavaBeans. However, OGSI has chosen an XML (WSDL) document that provides information about the serviceData, instead of using a serializable implementation class as in the BeanInfo model.

(b) Extending portType with serviceData

- ✓ ServiceData defines a new portType child element named serviceData, used to define serviceData elements, or SDEs, associated with that portType. These serviceData element definitions are referred to as serviceData declarations, or SDDs.
- ✓ Initial values for those serviceData elements (marked as "static" serviceData elements) may be specified using the staticServiceDataValues element within portType. The values of any serviceData element,
- ✓ whether declared statically in the portType or assigned during the life of the Web service instance, are called serviceData element values, or SDE values.

(c) serviceDataValue

- ✓ Each service instance is associated with a collection of serviceData elements: those serviceData elements defined within the various portTypes that form the service's interface, and also, potentially, additional service- Data elements added at runtime.
- ✓ OGSI calls the set of serviceData elements associated with a service instance its "serviceData set."
- ✓ A serviceData set may also refer to the set of serviceData elements aggregated from all serviceData elements declared in a portType interface hierarchy.Each service instance must have a "logical" XML document, with a root element of serviceDataValues that contains the serviceData element values.
- ✓ An example of a serviceDataValues element was given above. A service implementation is free to choose how the SDE values are stored; for example, it may store the SDE values not as XML but as instance variables that are converted into XML or other encodings as necessary.
- ✓ The wsdl:binding associated with various operations manipulating serviceData elements will indicate the encoding of that data between service requestor and service provider.
- ✓ For example, a binding might indicate that the serviceData element values are encoded as serialized Java objects.

(d) SDE Aggregation within a portType Interface Hierarchy

- ✓ WSDL 1.2 has introduced the notion of multiple portType extension, and one can model that construct within the GWSDL namespace. A portType can extend zero or more other portTypes. There is no direct relationship between a wsdl:service and the portTypes supported by the service modeled in the WSDL syntax.
- ✓ Rather, the set of portTypes implemented by the service is derived through the port element children of the service element and binding elements referred to from those port elements.
- ✓ This set of portTypes, and all portTypes they extend, defines the complete interface to the service.The serviceData set defined by the service's interface is the set union of the serviceData elements declared in each portType in the complete interface implemented by the service instance.
    - ✓ Because serviceData elements are uniquely identified by QName, the set union semantic implies that a serviceData element can appear only once in the set of serviceData elements. For example, if a portType named "pt1" and portType named "pt2" both declare a serviceData named "tns:sd1," and a port- Type named "pt3" extends both "pt1 and "pt2," then it has one (not two) serviceData elements named "tns:sd1."

(e) Dynamic serviceData Element

- ✓ Although many serviceData elements are most naturally defined in a service's interface definition, situations can arise in which it is useful to add or move serviceData elements dynamically to or from an instance.
- ✓ The means by which such updates are achieved are implementation specific; for example, a service instance may implement operations for adding a new serviceData element.
- ✓ The grid service portType illustrates the use of dynamic SDEs. This contains a serviceData element named "serviceDataName" that lists the serviceData elements currently defined.
- ✓ This property of a service instance may return a superset of the serviceData elements declared in the GWSDL defining the service interface, allowing the requestor to use the subscribe operation if this serviceDataSet changes, and the findServiceData operation to determine the current serviceDataSet value.

vi) Core Grid Service Properties

- ✓ This subsection discusses a number of properties and concepts common to all grid services.

(a) Service Description and Service Instance

- ✓ One can distinguish in OGSI between the *description* of a grid service and an *instance* of a grid service:
- ✓ **A *grid service description*** describes how a client interacts with service instances.This description is independent of any particular instance. Within aWSDL document, the grid service description is embodied in the most derived portType (i.e., the portType referenced by the wsdl:service element's port children, via referenced binding elements, describing the service) of the instance, along with its associated portTypes (including serviceData declarations),bindings, messages, and types definitions.

- ✓ A grid service description may be simultaneously used by any number of ***grid service instances***, each of which
- Embodies some state with which the service description describes how to interact
- Has one or more grid service handles
- Has one or more grid service references to it

A service description is used primarily for two purposes.

- ✓ First, as a description of a service interface, it can be used by tooling to automatically generate client interface proxies, server skeletons, and so forth.
- ✓ Second, it can be used for discovery, for example,to find a service instance that implements a particular service description, or to find a factory that can create instances with a particular service description.

The service description is meant to capture both interface syntax and (in a very rudimentary, nonnormative fashion) semantics. *Interface syntax* is described by WSDL portTypes. *Semantics* may be inferred through the name assigned to the portType.

For example, when defining a grid service, one defines zero or more uniquely named portTypes. Concise semantics can be associated with each of these names in specification documents, and, perhaps in the future, through Semantic Web or other more formal descriptions.

These names can then be used by clients to discover services with desired semantics, by searching for service instances and factories with the appropriate names. The use of namespaces to define these names also provides a vehicle for assuring globally unique names.

(b) Modeling Time in OGSI

- ✓ The need arises at various points throughout this specification to represent time that is meaningful to multiple parties in the distributed Grid.
- ✓ For example, information may be tagged by a producer with timestamps in order to convey that information's useful lifetime to consumers. Clients need to negotiate service instance lifetimes with services, and multiple services may need a common understanding of time in order for clients to be able to manage their simultaneous use and interaction.
- ✓ The GMT global time standard is assumed for grid services, allowing operations to refer unambiguously to absolute times. However, assuming the GMT time standard to represent time does *not* imply any particular level of clock synchronization between clients and services in the grid.
- ✓ In fact, no specific accuracy of synchronization is specified or expected by OGSI, as this is a service-quality issue. Grid service hosting environments and clients should utilize the Network Time Protocol (NTP) or equivalent function to synchronize their clocks to the global standard GMT time.
- ✓ However, clients and services must accept and act appropriately on messages containing time values that are out of range because of inadequate synchronization, where "appropriately" may include refusing to use the information associated with those time values.
- ✓ Furthermore, clients and services requiring global ordering or synchronization at a finer granularity than their clock accuracies or resolutions allow for must coordinate through the use of additional synchronization service interfaces, such as through transactions or synthesized global clocks.

- ✓ In some cases, it is required to represent both zero time and infinite time. Zero time should be represented by a time in the past. However, infinite time requires an extended notion of time. One therefore introduces the following type in the OGSI namespace that may be used in place of xsd:dateTime when a special value of "infinity"is appropriate.

(c) XML Element Lifetime Declaration Properties

- ✓ Since serviceData elements may represent instantaneous observations of the dynamic state of a service instance, it is critical that consumers of serviceData be able to understand the valid lifetimes of these observations.
- ✓ The client may use this time-related information to reason about the validity and availability of the serviceData element and its value, though the client is free to ignore the information.
- ✓ One can define three XML attributes that together describe the lifetimes associated with an XML element and its sub elements. These attributes may be used in any XML element that allows for extensibility attributes, including the servicData element.

The three lifetime declaration properties are:

1. ogsi:goodFrom. Declares the time from which the content of the element is said to be valid. This is typically the time at which the value was created.

2. ogsi:goodUntil. Declares the time until which the content of the element is said to be valid. This property must be greater than or equal to the goodFrom time.

3. ogsi:availableUntil. Declares the time until which this element itself is expected to be available, perhaps with updated values. Prior to this time, a client should be able to obtain an updated copy of this element. After this time, a client may no longer be able to get a copy of this element (while still observing cardinality and mutability constraints on this element). This property must be greater than or equal to the goodFrom time.

6) Explain in detail about Data intensive grid service models of OGSA?

Data-Intensive Grid Service Models

- ✓ Applications in the grid are normally grouped into two categories: computation-intensive and dataintensive.
- ✓ For data-intensive applications, we may have to deal with massive amounts of data.
- ✓ For example, the data produced annually by a Large Hadron Collider may exceed several petabytes (1015 bytes).
- ✓ The grid system must be specially designed to discover, transfer, and manipulate these massive data sets. Transferring massive data sets is a time-consuming task. Efficient data management demands low-cost storage and high-speed data movement.

Listed in the following paragraphs are several common methods for solving data movement problems.

(i) Data Replication and Unified Namespace
(ii) Grid Data Access Models

(iii)Parallel versus Striped Data Transfers

(i) Data Replication and Unified Namespace

- ✓ This data access method is also known as caching, which is often applied to enhance data efficiency in a grid environment. By replicating the same data blocks and scattering them in multiple regions of a grid, users can access the same data with locality of references. Furthermore, the replicas of the same data set can be a backup for one another. Some key data will not be lost in case of failures.
- ✓ However, data replication may demand periodic consistency checks. The increase in storage requirements and network bandwidth may cause additional problems.

- ✓ Replication strategies determine when and where to create a replica of the data. The factors to consider include data demand, network conditions, and transfer cost. The strategies of replication can be classified into method types: dynamic and static.
- ✓ For the static method, the locations and number of replicas are determined in advance and will not be modified. Although replication operations require little overhead, static strategies cannot adapt to changes in demand, bandwidth,and storage vailability.
- ✓ Dynamic strategies can adjust locations and number of data replicas according to changes in conditions (e.g., user behavior).
- ✓ However, frequent data-moving operations can result in much more overhead than in static strategies.The replication strategy must be optimized with respect to the status of data replicas. For static replication, optimization is required to determine the location and number of data replicas.
- ✓ For dynamic replication, optimization may be determined based on whether the data replica is being created, deleted, or moved. The most common replication strategies include preserving locality, minimizing update costs, and maximizing profits.

(ii) Grid Data Access Models

- ✓ Multiple participants may want to share the same data collection. To retrieve any piece of data, we need a grid with a unique global namespace. Similarly, we desire to have unique file names.
- ✓ To achieve these, we have to resolve inconsistencies among multiple data objects bearing the same name. Access restrictions may be imposed to avoid confusion. Also, data needs to be protected to avoid leakage and damage. Users who want to access data have to be authenticated first and then authorized for access.

In general, there are four access models for organizing a data grid, as listed here and shown in Figure 7.5.

- **Monadic model**
- **Hierarchical model**
- **Federation model**
- **Hybrid model**

Monadic model:

- ✓ This is a centralized data repository model, shown in Figure 7.5(a). All the data is saved in a central data repository. When users want to access some data they have to submit requests directly to the central repository.
- ✓ No data is replicated for preserving data locality. This model is the simplest to implement for a small grid. For a large grid, this model is not efficient in terms of performance and reliability. Data replication is permitted in this model only when fault tolerance is demanded.

Hierarchical model:

- ✓ The hierarchical model, shown in Figure 7.5(b), is suitable for building a large data grid which has only one large data access directory. The data may be transferred from the source to a second-level center.
- ✓ Then some data in the regional center is transferred to the third-level center. After being forwarded several times, specific data objects are accessed directly by users.
- ✓ Generally speaking, a higher-level data center has a wider coverage area. It provides higher bandwidth for access than a lower-level data center. PKI security services are easier to implement in this hierarchical data access model.

Federation model:

- ✓ This data access model shown in Figure 7.5(c) is better suited for designing a data grid with multiple sources of data supplies. Sometimes this model is also known as a mesh model.
- ✓ The data sources are distributed to many different locations. Although the data is shared, the data items are still owned and controlled by their original owners. According to predefined access policies, only authenticated users are authorized to request data from any data source. This mesh model may cost the most when the number of grid institutions becomes very large.

Hybrid model:

- ✓ This data access model is shown in Figure 7.5(d). The model combines the best features of the hierarchical and mesh models. Traditional data transfer technology, such as FTP, applies for networks with lower bandwidth.
- ✓ Network links in a data grid often have fairly high bandwidth, and other data transfer models are exploited by high-speed data transfer tools such as GridFTP developed with the Globus library. The cost of the hybrid model can be traded off between the two extreme models for hierarchical and mesh-connected grids.

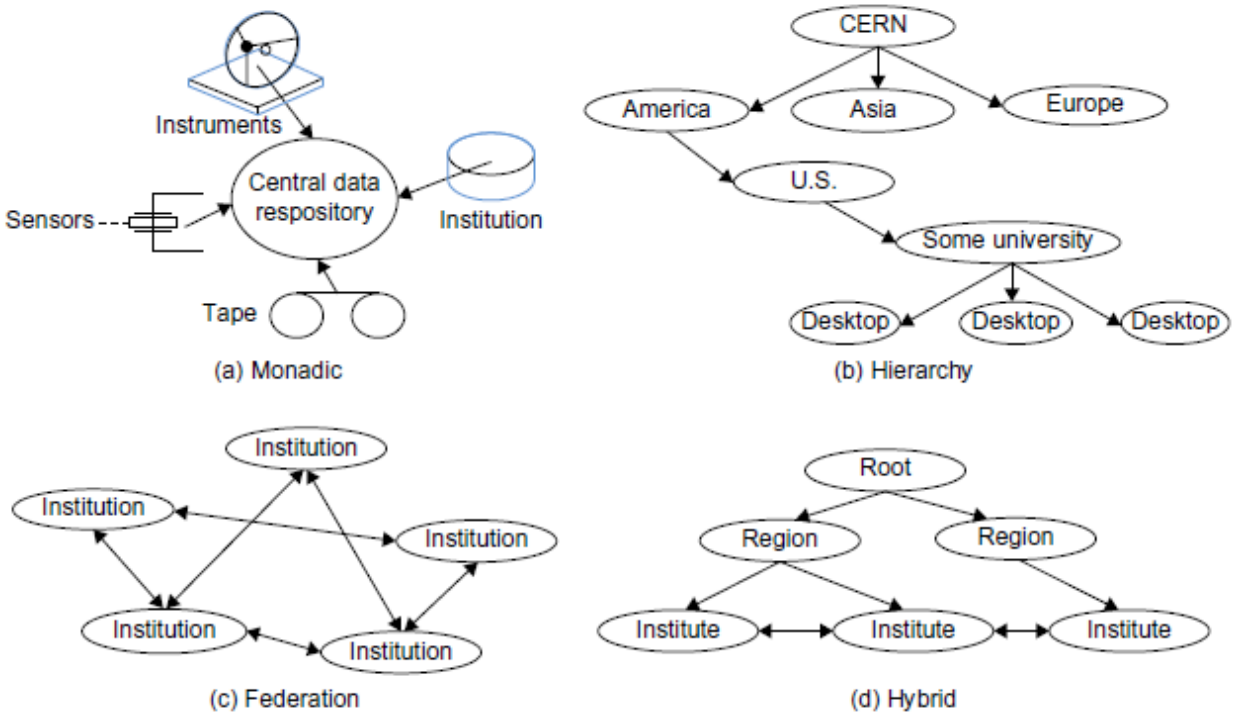


FIGURE 7.5

Four architectural models for building a data grid.

(iii) Parallel versus Striped Data Transfers

- ✓ Compared with traditional FTP data transfer, parallel data transfer opens multiple data streams for passing subdivided segments of a file simultaneously. Although the speed of each stream is the same as in sequential streaming, the total time to move data in all streams can be significantly reduced compared to FTP transfer.
- ✓ In striped data transfer, a data object is partitioned into a number of sections, and each section is placed in an individual site in a data grid.
- ✓ When a user requests this piece of data, a data stream is created for each site, and all the sections of data objects are transferred simultaneously. Striped data transfer can utilize the bandwidths of multiple sites more efficiently to speed up data transfer.

7. Explain in detail about the OGSA SERVICES

(1) Handle Resolution

- ✓ OGSI defines a two-level naming scheme for grid service instances based on abstract, long-lived *grid service handles* (GSHs) that can be mapped by HandleMapper services to concrete, but potentially less long lived, *grid service references* (GSRs).
- ✓ These constructs are basically network-wide pointers to specific grid service instances hosted in (potentially remote) execution environments.

- ✓ A client application can use a grid service reference to send requests (represented by the perations defined in the interfaces of the target service) directly to the specific instance at the specified network-attached service endpoint identified by that GSR.
- ✓ The format of the GSH is a URL, where the schema directive indicates the naming scheme used to express the handle value. Based on the GSH naming scheme, the application should find an associated naming-scheme-specific HandleMapper service that knows how to resolve that name to the associated GSR.
- ✓ This allows different naming scheme implementations to coexist, and to provide different QoS properties through their implementation. OGSI defines the basic GSH format and portType for the HandleMapper service that resolve a GSH to a GSR.
- ✓ The expectation is that different implementations of the two-level naming scheme with the naming directive will enable features such as transparent service instance migration, fault tolerance through transparent failover, high availability through mirroring, and advanced security through fine-grained access control on the name resolution.
- ✓ All these features come with an associated cost, and their applicability will, therefore, depend on the application itself. The OGSI Working Group decided to leave the registration of GSHs and associated GSRs undefined for possible standardization elsewhere (the same applies to the authorization of the naming resolution as well as the features that deal with scalability and robustness.)
- ✓ Another unspecified issue is how the bootstrap mechanism should work. In other words, given the handle of the very first service to contact, how would a party find the associated resolution service?
- ✓ Currently, it is left to the implementation, which may decide to use custom configuration data and external naming services; for example, DNS or the Handle System.
- ✓ The handle resolutions require service invocations and could, therefore, affect overall performance. This issue could be addressed by local caching, or possibly by the definition of a generic service that maintains caching information about the GSH–GSR mappings independent from the handle resolution scheme.

(2) Virtual Organization Creation and Management

- ✓ VOs are a concept that supplies a "context" for operation of the grid that can be used to associate users, their requests, and resources. VO contexts permit the grid resource providers to associate appropriate policy and agreements with their resources.
- ✓ Users associated with a VO can then exploit those resources consistent with those policies and agreements. VO creation and management functions include mechanisms for associating users/groups with a VO, manipulation of user roles (administration, configuration, use, etc.) within the VO, association of services (encapsulated resources) with the VO, and attachment of agreements and policies to the VO as a whole or to individual services within the VO.
- ✓ Finally, creation of a VO requires a mechanism by which the "VO context" is referenced and associated with user requests (this is most likely via a GSH, since a "service" is the likely embodiment of the VO).

(3)Service Groups and Discovery Services

- ✓ GSHs and GSRs together realize a two-level naming scheme, with HandleResolver services mapping from handles to references; however, GSHs are not intended to contain emantic information and indeed may be viewed for most purposes as opaque.
- ✓ Thus, other entities (both humans and applications) need other means for discovering services with particular properties, whether relating to interface, function, availability, location, policy, or other criteria.
- ✓ Traditionally, in distributed systems this problem is addressed by creating a third-level "human-readable" or "semantic" name space that is then mapped (bound) to abstract names (in this case, GSHs) via registry, discovery, metadata catalog, or other similar services.
- ✓ It is important that OGSA defines standard functions for managing such name spaces, otherwise services and clients developed by different groups cannot easily discover each other's existence and properties.
- ✓ These functions must address the creation, maintenance, and querying of name mappings. Two types of such semantic name spaces are common—naming by attribute, and naming by path.

***Attribute naming* schemes**

- ✓ Associate various metadata with services and support retrieval via queries on attribute values. A registry implementing such a scheme allows service providers to publish the existence and properties of the services that they provide, so that service consumers can discover them.
- ✓ One can envision special-purpose registries being built on the base service group mechanisms provided by the OGSI definition. In other words, an OGSA-compliant registry is a concrete specialization of the OGSI service group.
- ✓ A ServiceGroup is a collection of entries, where each entry is a grid service implementing the ServiceGroupEntry interface. The ServiceGroup interface also extends the GridService interface.
- ✓ There is a ServiceGroupEntry for each service in the group (i.e., for each group member). Each ServiceGroupEntry contains a serviceLocator for the referred-to service and information (content) about that service.
- ✓ The content element is an XML element advertising some information about the member service. The type of the content element conforms to one of the QName elements in the ContentModelType SDE of the ServiceGroup interface.
- ✓ It is the content model of the service group definition that suggests the concrete type and specific use of the registry being offered. The content model of the serviceGroupEntry for a given service group is published in the service data of the service group.
- ✓ The content model is the basis on which search predicates can be formed and executed against the service group with the findServiceData operation.
- ✓ In other words, it is the content model that forms the basis of the registry index upon which registry searches can be executed. It is envisioned that many application-specific, special-purpose registries will be developed. It is also envisioned that many registries will inherit and implement the notificationSource interface so as to facilitate client subscription to register state changes.

- ✓ Again, specific state change subscriptions will be possible through the advertisement of the registry-specific service group content model of the service group on which the registry is built.
- ✓ As stated earlier, one can envision many application-specific registry implementations being defined. Whether or not one or more general-purpose registry types should be defined and adopted as part of OGSA is to be determined.

***Path naming* or *directory* schemes** (as used, for example, in file systems)

- ✓ Represent an alternative approach to attribute schemes for organizing services into a hierarchical name space that can be navigated.
- ✓ The two approaches can be combined, as in LDAP. Directory path naming can be accomplished by defining a PathName Interface that maps strings to GSHs.
- ✓ Thus, a string such as "/data/genomics_ dbs/mouse" could map to a service (GSH) that might deliver portions of the mouse genome, and perhaps also do BLAST searches against the mouse genome.
- ✓ Similarly, "/applications/biology/genomics/BLAST" could map to a GSH that has Interfaces for executing BLAST. The interface will have methods to insert, look up, and delete <string, GSH> pairs, and will in essence be a simple table.
- ✓ It is expected that path_name services will be "chained" together, so that evaluation of a path may involve traversing several path_name services, forming a directed graph. This can be used to link disjoint namespaces into namespace cliques.

(4) Choreography, Orchestration, and Workflow

Over these interfaces OGSA provides a rich set of behaviors and associated operations and attributes for business process management (additional work remains to be done in this area):

- Definition of a job flow, including associated policies
- Assignment of resources to a grid flow instance
- Scheduling of grid flows (and associated grid services)
- Execution of grid flows (and associated grid services)
- Common context and metadata for grid flows (and associated services)
- Management and monitoring for grid flows (and associated grid services)
- Failure handling for grid flows (and associated grid services); more generally,managing the potential transiency of grid services
- Business transaction and coordination services

(5) Transactions

- ✓ Transaction services are important in many grid applications, particularly in industries such as financial services and in application domains such as supply chain management. However, transaction management in a widely distributed, high-latency, heterogeneous RDBMS environment is more complicated than in a single data center with a single vendor's software.
- ✓ Traditional distributed transaction algorithms, such as two-phase distributed commit, may be too expensive in a wide-area grid, and other techniques such as optimistic protocols may be more appropriate.

- ✓ At the same time, different applications often have different characteristics and requirements that can be exploited when selecting a transaction technique to use. Thus, it is unlikely that there will be a "one size fits all" solution to the transaction problem.

(6) Metering Service

- ✓ Different grid deployments may integrate different services and resources and feature different underlying economic motivations and models; however, regardless of these differences, it is a quasiuniversal requirement that resource utilization can be monitored, whether for purposes of cost allocation (i.e., charge back), capacity and trend analysis, dynamic provisioning, grid-service pricing, fraud and intrusion detection, and/or billing. OGSA must address this requirement by defining standard monitoring, metering, rating, accounting, and billing interfaces.
- ✓ These interfaces can use or extend those defined within the Common Management Model (CMM) that provides access to basic resource performance and utilization instrumentation, exposed as serviceData.
- ✓ For example, an operating system might publish countervalues corresponding to the state of system activities such as processor utilization, memory usage, disk and tape I/O activity, network usage, and so on.
- ✓ Although interfaces do not provide the values needed for metering and accounting directly (since, for instance, they are not directly related to the consumers), they can provide basic monitoring data to be used by the metering and accounting services.
- ✓ We address metering in this subsection and rating, accounting, and billing in the sections that follow. A grid service may consume multiple resources and a resource may be shared by multiple service instances.
- ✓ Ultimately, the sharing of underlying resources is managed by middleware and operating systems. All modern operating systems and many middleware systems have metering subsystems for measuring resource consumption (i.e., monitored data) and for aggregating the results of those measurements.
- ✓ For example, all commercial Unix systems have provisions for aggregating prime-time and nonprime-time resource consumption by user and command. A metering interface provides access to a standard description of such aggregated data (metering serviceData).
- ✓ A key parameter is the time window over which measurements are aggregated. In commercial Unix systems, measurements are aggregated at administrator-defined intervals (chronological entry), usually daily, primarily for the purpose of accounting.
- ✓ On the other hand, metering systems that drive active workload management systems might aggregate measurements using time windows measured in seconds. Dynamic provisioning systems use time windows somewhere between these two examples. Several use cases require metering systems that support multitier, end-to-end flows involving multiple services.
- ✓ An OGSA metering service must be able to meter the resource consumption of configurable classes of these types of flows executing on widely distributed, loosely coupled server, storage, and network resources.
- ✓ Configurable classes should support, for example, a departmental charge-back scenario where incoming requests and their subsequent flows are partitioned into account classes determined by the department providing the service. The metering of end-to-end flows in

a grid environment is somewhat analogous to the metering of individual processes in a traditional OS.

- ✓ Since traditional middleware and operating systems do not support this type of metering, additional functions must be accommodated by OGSA. In addition to traditional accounting applications, it is anticipated that end-to-end resource consumption measurements will play an important role in dynamic provisioning and pricing grid services.
- ✓ Finally, in addition to metering resource consumption, metering systems must also accommodate the measurement and aggregation of application-related (e.g., licensed) resources.
- ✓ For example, a grid service might charge consuming services a per-use fee. The metering service must be able to support the measurement of this class of service (resource) consumption.

(7) Rating Service

- ✓ A rating interface needs to address two types of behaviors. Once the metered information is available, it has to be translated into financial terms. That is, for each unit of usage, a price has to be associated with it.
- ✓ This step is accomplished by the rating interfaces, which provide operations that take the metered information and a rating package as input and output the usage in terms of chargeable amounts.
- ✓ Furthermore, when a business service is developed, a rating service is used to aggregate the costs of the components used to deliver the service, so that the service owner can determine the pricing, terms, and conditions under which the service will be offered to subscribers.

(8) Accounting Service

- ✓ Once the rated financial information is available, an accounting service can manage subscription users and accounts information, calculate the relevant monthly charges and maintain the invoice information.
- ✓ This service can also generate and present invoices to the user. Account-specific information is also applied at this time.
- ✓ For example, if a user has a special offer of a 20% discount for his usage of the commercial UNIX system described above, this discount will be applied by the accounting service to indicate a final invoiced amount of 24 dollars.

(9) Billing and Payment Service

- ✓ Billing and payment service refers to the financial service that actually carries out the transfer of money; for example, a credit card authorization service.

(10) Installation, Deployment, and Provisioning

- ✓ Computer processors, applications, licenses, storage, networks, and instruments are all grid resources that require installation, deployment, and provisioning (other new resource types will be invented and added to this list.)

- ✓ OGSA affords a framework that allows resource provisioning to be done in a uniform, consistent manner.

(11) Distributed Logging

- ✓ Distributed logging can be viewed as a typical messaging application in which *message producers* generate *log artifacts,* (atomic expressions of diagnostic information) that may or may not be used at a later time by other independent *message consumers.*
- ✓ OGSA-based logging can leverage the notification mechanism available in OGSI as the transport for messages. However, it is desirable to move logging-specific functionality to intermediaries, or *logging services.*
- ✓ Furthermore, the secure logging of events is required for the audit trails needed to fulfill judiciary and organizational policy requirements, to reconcile security-related inconsistencies, and to provide for forensic evidence both after the fact and in real time.
- ✓ It is expected that standards and implementations for secure logging should be able to considerably leverage on the efforts associated with distributed logging.
- ✓ Because the invocation of operations on service instances are well defined and typed through their portType definitions, and because it is expected that most implementations will have a model in which the run-time invocation module is generated independent of the application, it follows that there is the opportunity to transparently provide logging of many natural logging-code points (see Figure 5.6).
- ✓ The logging capability includes not only the logging of information about the invocation of the actual service operations on both requestor and service provider, but also the information about the transparent invocations to the all the supporting services that were needed to enable the application service invocation (such as services for discovery, registration, directory, access control, privacy, identity translation, etc.)
- ✓ Most of the required logging and secure logging may be achieved transparent of the application and outside of the application logic, which greatly facilitates an implementation that adheres to the local policy.

Logging services provide the extensions needed to deal with the following issues:

Decoupling

- ✓ The logical separation of logging artifact creation from logging artifact consumption. The ultimate usage of the data (e.g., logging, tracing, management) is determined by the message consumer; the message producer should not be concerned with this.

Transformation and common representation

- ✓ Logging packages commonly annotate the data that they generate with useful common information such as category, priority, time stamp, and location.
- ✓ An OGSA logging service should not only provide the capability of annotating data, but also the capability of converting data from a range of (legacy) log formats into a common, standard canonical representation. Also, a general mechanism for transformation may be required.

Filtering and aggregation

- ✓ The amount of logging data generated can be large, whereas the amount of data actually consumed can be small. Therefore,it can be desirable to have a mechanism for

controlling the amount of data generated and for filtering out what is actually kept and where.

- ✓ Through the use of different filters, data coming from a single source can be easily separated into different repositories, and/or "similar" data coming from different sources can be aggregated into a single repository.

Configurable persistency

- ✓ Depending on consumer needs, data may have different durability characteristics. For example, in a real-time monitoring application, data may become irrelevant quickly, but be needed as soon as it is generated; data for an auditing program may be needed months or even years after it was generated.
- ✓ Hence, there is a need for a mechanism to create different data repositories, each with its own persistency characteristics. In addition, the artifact retention policy (e.g., determining which log artifacts to drop when a buffer reaches its size limit) should be configurable.

Consumption patterns

- ✓ Consumption patterns differ according to the needs of the consumer application. For example, a real-time monitoring application needs to be notified whenever a particular event occurs, whereas a postmortem problem determination program queries historical data, trying to find known patterns.
- ✓ Thus, the logging repository should support both synchronous query- (pull-) based consumption and asynchronous push-based (eventdriven) notification. The system should be flexible enough that consumers can easily customize the event mechanism—for example, by sending digests of messages instead of each one—and maybe even provide some predicate logic on log artifacts to drive the notifications.

These considerations lead to an architecture for OGSA logging services in which producers talk to *filtering and transformation* services either directly, or indirectly through adapters. Consumers also use this service to create custom message repositories (*baskets*) or look for existing producers and baskets; that is, this service should also function as a factory (of baskets) and a registry (of producers and baskets).

There is also a need for a configurable *storage and delivery* service, with which data from different filtering services is collected, stored, and, if required, delivered to interested consumers. Log service also implies the ability to access related log entries (by context) in chronological sequence.

(12) Messaging and Queuing

- ✓ OGSA extends the scope of the base OGSI Notification Interface to allow grid services to produce a range of event messages, not just notifications that a serviceData element has changed.

Several terms related to this work are:

Event—Some occurrence within the state of the grid service or its environment that may be of

interest to third parties. This could be a state change or it could be environmental, such as a timer event.

Message—An artifact of an event, containing information about an event that some entity wishes to communicate to other entities.

Topic—A "logical" communications channel and matching mechanism to which a requestor may subscribe to receive asynchronous messages and publishers may publish messages.

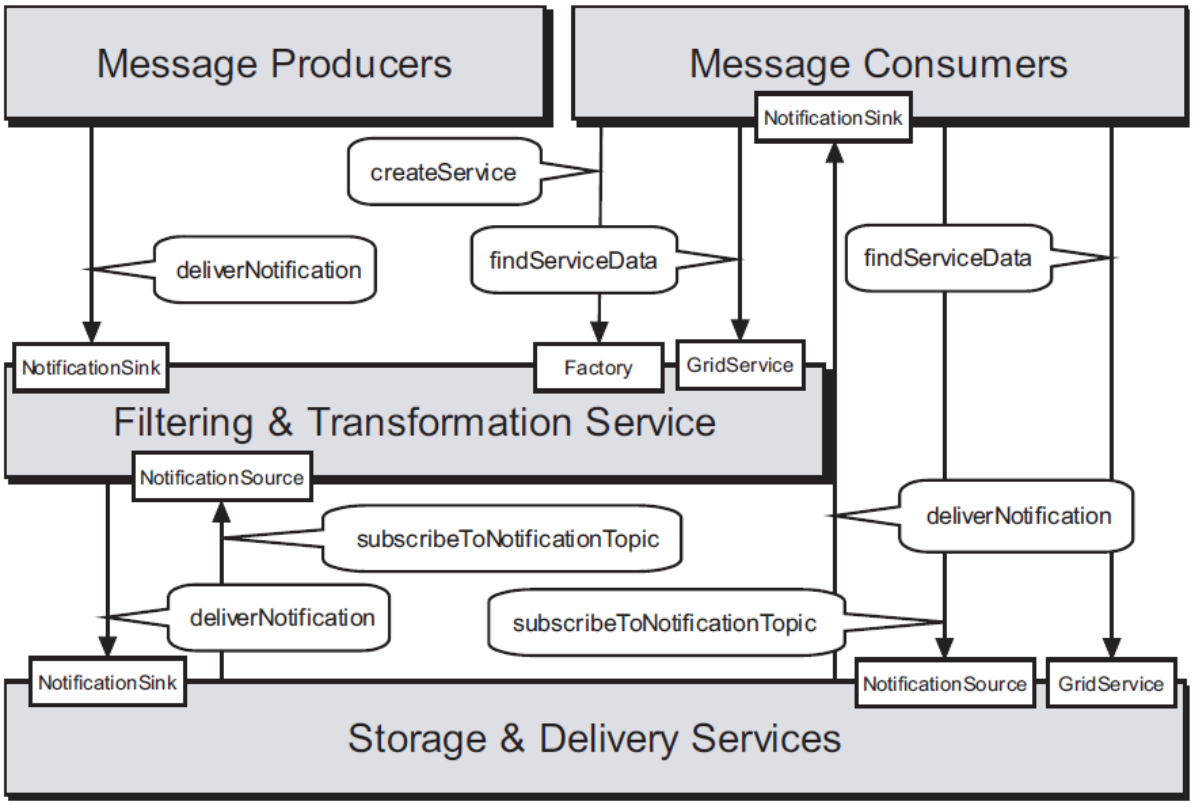


Figure 5.6 Schematic of a messaging service architecture.

- ✓ A message is represented as an XML element with a namespace-qualified QName, and an XML schema-defined complex type.
- ✓ A topic will be modeled as an XML element, describing its internal details, including expected messages associated with the topic. TopicSpaces, or collections of topics will also be modeled.

This work also defines:

- An interface to allow any grid service to declare its ability to accept subscriptions to topics and the topics its supports.
- An interface to describe a messaging intermediary (a message broker) that supports anonymous publication and subscription on topics.
- An interface (or set of interfaces) that describe the interface to other messaging services such as a queuing service.

- ✓ Note that queuing and message qualities of service such as reliability can be considered as both explicit services within an OGSA hosting environment and transport details modeled by the wsdl:binding element in the service description.

(13) Events

- ✓ Events are generally used as asynchronous signaling mechanisms. The most common form is "publish/subscribe," in which a service "publishes" the events that it exports (makes available to clients).
- ✓ The service may publish the events as reliable or best effort. Clients may then "subscribe" to the event, and when the event is raised, a call-back or message is sent to the client. Once again, the client can usually request either reliable or best effort, though the service may not be able to accept a reliable delivery request.
- ✓ There is also a distinction between the reliability of an event being raised and its delivery to a client. A service my attempt to deliver every occurrence of an event (reliable posting), but not be able to guarantee delivery (best-effort delivery).
- ✓ An event can be anything that the service decides it will be: a change in a state variable, entry into a particular code segment, an exception such as a security violation or floating point overflow, or the failure of some other expected event to occur.
- ✓ A second form of event is carried with service method invocations .The basic idea is simple: inside the SOAP message invoking a service method is an Event Interest Set (EIS). The EIS specifies the events in which the caller is interested and a callback associated with each event.
- ✓ If an event named in the EIS is raised during the execution of the method—including outcalls on other grid services—the corresponding call-back is invoked. The EIS is *first class* and can be thought of as a part of the calling context (closure); as such, it can be modified by the callee and propagated down the call chain.
- ✓ This allows the callee to express interest in different events further down the call chain, add himself/herself to a list of subscribers interested in events further down the call chain, or "catch" events and keep them from propagating further up the call chain.
- ✓ However, OGSI notifications are designed to capture state changes in SDE's only, and are not designed as a general event mechanism.
- ✓ Furthermore, notifications are not normally "edge triggered" since they are based on a model that assumes a minimum and maximum time between notifications; for example, notifications that may occur within the minimum time window will not be sent to the client.
- ✓ A complete event framework would allow for event queuing or batching to cover this kind of functionality. OGSA events will build on OGSI notifications. An event is a representation of an occurrence in a system or application component that may be of interest to other parties.
- ✓ Standard means of representing, communicating, transforming, reconciling, and recording events are important for interoperability.
- ✓ Thus, the OGSA Core should define standard schema for at least certain classes of OGSA events.

A detailed set of services include:

- Standard interface(s) for **communicating** events with specified QoS. These may be based directly on the Messaging interfaces.
- Standard interface(s) for **transforming (**mediating) events in a manner that is transparent to the endpoints. This should include **aggregation** of multiple events into a single event.
- Standard interface(s) for **reconciling** events from multiple sources.
- Standard interface(s) for **recording** events. These may be based directly on the Message logging interface(s).
- Standard interface(s) for **batching and queuing** events.

✓ It is possible that some forms of event services may be built directly on top of OGSI notification interfaces, although more study is needed.

(14) Policy and Agreements

- ✓ These services create a general framework for creation, administration, and management of policies and agreements for system operation, security, resource allocation, and so on, as well as an infrastructure for "policy aware" services to use the set of defined and managed policies to govern their operation.
- ✓ These services do not actually enforce policies but permit policies to be managed and delivered to resource managers that can interpret and operate on them. Agreements (OGSI-Agreement Specification) provide a mechanism for the representation and negotiation of terms between service providers and their clients (either user requests or other services).
- ✓ These terms include specifications of functional, performance, and quality requirements/objectives that the suppliers and consumers exchange and that they can then use to influence their interactions.
- ✓ The Agreement mechanism provides a general expression framework for these terms but leaves the specification of particular terms to individual disciplines such as performance, security, data quality, and so on.
- ✓ Agreements also contain information on priority, costs, penalties or consequences associated with violation of the "contract" a negotiated agreement represents as well as information on how the service provider and consumer have decided to measure compliance with the agreement.
- ✓ One can expect that many grid services will use policies to direct their actions. Thus, grids need to support the definition, discovery, communication, and enforcement of policies for such purposes as resource allocation, workload management, security, automation, and qualities of services.
- ✓ Some policies need to be expressed at the operational level, that is, at the level of the devices and resources to be managed, whereas higher-level policies express business goals and SLAs within and across administrative domains.
- ✓ Higher-level policies are hard to enforce without a canonical representation for their meaning to lower-level resources. Thus, business policies probably need to be translated into a canonical form that can then be used to derive lower-level policies that resources can understand. Standard mechanisms are also needed for managing and distributing

policies from producers (e.g., administrators, autonomic managers, and SLAs) to endpoints that consume and enforce them (i.e., devices and resources).

These services (interfaces) provide a framework for creating, managing, validating, distributing, transforming, resolving, and enforcing policies within a distributed environment.

A detailed set of services include:

- A management control point for policy life cycle: the Policy Service Manager interface
- An interface that policy consumers can use to retrieve required policies: the Policy Service Agent interface
- A way to express that a service is "policy aware": the Policy Enforcement Point interface
- A way to effect change on a resource, for example, by using Common Management Models

- ✓ *The **Policy Service Manager*** controls access to the policy repository. It also controls when notifications of policy changes are sent out so that multiple updates can be made and notifications are only sent after all updates are complete.
- ✓ *The **Policy Service Agent*** is a management service that other "policy aware" services depend on for delivery of their policies.
- ✓ The agent can provide additional services like understanding time-period conditions so it can inform policy consumers of when policies become active or inactive. Services that consume policies will implement the ***Policy Enforcement Point*** interface to allow them to be registered with policy agents, participate in the subscription to and notification of policy changes, and to allow policies to be pushed down onto them when needed.
- ✓ These enforcement points will need to interpret the policies and make the necessary configuration changes in the resource(s) they manage by using the Common Management Model.
- ✓ The OGSA Policy Service provides for a ***transformation service*** for this purpose and includes a canonical representation of policy in the form of an information model, grammar, and core XML schema (also see Figure 5.7).
- ✓ A set of secondary validation interfaces can allow automated managers and administrators to act on the same set of policies and validate consistency. An interface is also required for translating policies to and from the canonical form so that consumers that have their own policy formats can plug into the service.
- ✓ Finally, there is a need for run-time resolution of policy conflicts that may require specific application knowledge to determine the cost of violating an agreement and selecting the policy that will have the appropriate impact.

(15) Base Data Services

- ✓ OGSA data interfaces are intended to enable a service-oriented treatment of data so that data can be treated in the same way as other resources within the Web/grid services architecture.
- ✓ Thus, for example, one can integrate data into registries and co- ordinate operations on ata using service orchestration mechanisms.
- ✓ A service-oriented treatment of data also allows us to use OGSI grid service handles as global names for data, manage the lifetime of dynamically created data by using OGSI

lifetime management mechanisms, and represent agreements concerning data access via WS-Agreement.

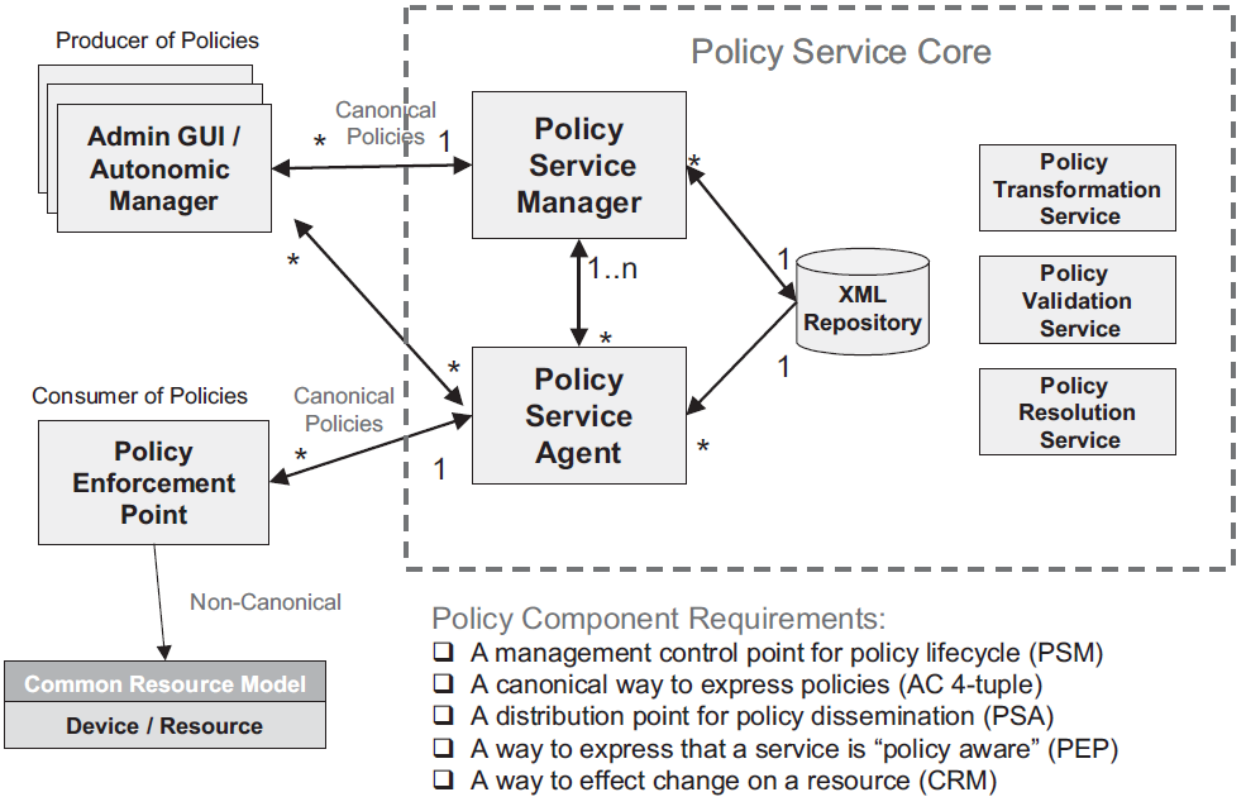


Figure 5.7 A set of potential policy service components.

- ✓ OGSA data services are intended to allow for the definition, application, and management of diverse abstractions—what can be called *data virtualizations*—of underlying data sources.
- ✓ A data virtualization is represented by, and encapsulated in, a *data service*, an OGSI grid service with SDEs that describe key parameters of the virtualization, and with *operations* that allow clients to inspect those SDEs, access the data using appropriate operations, derive new data virtualizations from old, and/or manage the data virtualization.
- ✓ For example, a file containing geographical data might be made accessible as an image via a data service that implements a "JPEG image" virtualization, with SDEs defining size, resolution, and color characteristics, and operations provided for reading and modifying regions of the image.
- ✓ Another virtualization of the same data could present it as a relational database of coordinate-based information, with various specifics of the schema (e.g., table names, column names, types) as SDEs, and SQL as its operations for querying and updating the geographical data. In both cases, the data service implementation is responsible for managing the mapping to the underlying data source.

Four *base data interfaces* (WSDL portTypes) can be used to implement a variety of different data service behaviors:

5. ***DataDescription*** defines OGSI service data elements representing key parameters of the data virtualization encapsulated by the data service.
6. ***DataAccess*** provides operations to access and/or modify the contents of the data virtualization encapsulated by the data service.
7. ***DataFactory*** provides an operation to create a new data service with a data virtualization derived from the data virtualization of the parent (factory) data service.
8. ***DataManagement*** provides operations to monitor and manage the data service's data virtualization, including (depending on the implementation) the data sources (such as database management systems) that underlie the data service.

A data service is any OGSI-compliant Web service that implements one or more of these base data interfaces.

(16) Other Data Services

- ✓ A variety of higher-level data interfaces can and must be defined on top of the base data interfaces, to address functions such as:
    - Data access and movement
    - Data replication and caching
    - Data and schema mediation
    - Metadata management and looking
- ✓ There are likely to be strong relationships to discovery, messaging, agreement, and coordination functions. Basic data access interfaces allow clients to directly access and manipulate data.
- ✓ A number of such interfaces are required, corresponding to different data types, for example, files, directories, file systems, RDBMS, XML data bases, object data bases, and streaming media.
- ✓ A "file access" service may export interfaces to *read, write,* or *truncate.* GridFTP, an existing data access service, provides mechanisms to *get* and *put* files, and supports third-party transfers.

Data replication, data caching, and schema transformation subservices are described below.

Data Replication

- ✓ Data replication can be important as a means of meeting performance objectives by allowing local computer resources to have access to local data. Although closely related to caching (indeed, a "replica store" and a "cache" may differ only in their policies), replicas may provide different interfaces.
- ✓ Services that may consume data replication are group services for clustering and failover, utility computing for dynamic resource provisioning, policy services ensuring various qualities of service, metering and monitoring services, and also higher-level workload management and disaster recovery solutions.
- ✓ Each may need to migrate data for computation or to replicate state for a given service. Work is required to define an OGSA-compliant set of data replication services that,

through the use of "adapters," can move data in and out of heterogeneous physical and logical environments without any changes needed to the underlying local data access subsystems.

- ✓ The adapters handle the native "reading" and "writing" of data and the replication software coordinates the run time (recoverability, monitoring, etc.) associated with every data transfer. A central "monitor" sets up and handles communication with the calling service or program and sets up a "subscription–pair" relationship between capture and apply services on a per-replication- request basis to ensure reliability.

Data Caching

- ✓ In order to improve performance of access to remote data items, caching services will be employed. At the minimum, caching services for traditional flat file data will be employed
- ✓ Caching of other data types, such as views on RDBMS data, streaming data, and application binaries, are also envisioned. Issues that arise include (but are not limited to):

Consistency—Is the data in the cache the same as in the source? If not, what is the coherence window? Different applications have very different requirements.

Cache invalidation protocols—How and when is cached data invalidated? Write through or write back? When are writes to the cache committed back to the original data source?

Security—How will access control to cached items be handled? Will access control enforcement be delegated to the cache, or will access control be somehow enforced by the original data source?

Integrity of cached data—Is the cached data kept in memory or on disk? How is it protected from unauthorized access? Is it encrypted?

How the cache service addresses these issues will need to available as service data.

Schema Transformation

- ✓ Schema transformation interfaces support the transformation of data from one schema to another. For example, XML transformations as specified in XSLT.

(17) Discovery Services

- ✓ Discovery interfaces address the need to be able to organize and search for information about various sorts of entities in various ways. In an OGSA environment, it is normally recommended that entities of whatever type be named by GSHs; thus, discovery services are concerned with mapping from user-specified criteria to appropriate GSHs.
- ✓ Different interface definitions and different implementation behaviors may vary according to how user requests are expressed, the information used to answer requests, and the mechanisms used to propagate and access that information.

(18) Job Agreement Service

- ✓ The job agreement service is created by the agreement factory service with a set of job terms, including command line, resource requirements, execution environment, data staging, job control, scheduler directives, and accounting and notification terms.
- ✓ The job agreement service provides an interface for placing jobs on a resource manager (i.e., representing a machine or a cluster), and for interacting with the job once it has been dispatched to the resource manager. The job agreement service provides basic matchmaking capabilities between the requirements of the job and the underlying resource manager available for running the job.
- ✓ More advanced job agreement services take into account more advanced job characteristics such as interactive execution, parallel jobs across resource managers, and jobs with requirements based on SLAs.

The interfaces provided by the job agreement service are:

Manageability interface

- Supported job terms: defines a set of service data used to publish the job terms supported by this job service, including the job definition (command line and application name), resource requirements, execution environment, data staging, job control, scheduler directives, and accounting and notification terms.
- Workload status: total number of jobs, statuses such as number of jobs running or pending and suspended jobs.

Job control: control the job after it has been instantiated. This would include the ability to suspend/resume, checkpoint, and kill the job.

The job agreement service makes use of information in the form of policies that are defined at the VO level, and resource information about available resource managers, queues, host, and job status as provided by the Global Information Service.

The job agreement service uses the WS-Agreement Protocol with the underlying resource managers in order to submit jobs to the underlying resource managers, and in order to control the running jobs and to access data.

(19) Reservation Agreement Service

- ✓ The reservation agreement service is created by the agreement factory service with a set of terms including time duration, resource requirement specification, and authorized user/project agreement terms.
- ✓ The reservation agreement service allows end users or a job agreement service to reserve resources under the control of a resource manager to guarantee their availability to run a job. The service allows reservations on any type of resource (e.g., hosts, software licenses, or network bandwidth).
- ✓ Reservations can be specific (e.g., provide access to host "A" from noon to 5 PM), or more general (e.g., provide access to 16 Linux cpus on Sunday). Once a reservation is made, a job service can send a job to a resource manager that is attached to the provided reservation.
- ✓ Some of the policy decisions made by the reservation service for use with a resource manager include notions of who can make reservations (e.g., administrators only), how

many hosts a particular user or user group can reserve at a time, when reservations can be made (i.e., define "blackout periods"), and what types of hosts can be reserved.

- ✓ The reservation agreement service provides one interface, manageability, which defines a set of service data that describe the details of a particular reservation, including resource terms, start time, end time, amount of the resource reserved, and the authorized users.
- ✓ The reservation service makes use of information about the existing resource managers available and any policies that might be defined at the VO level, and will make use of a logging service to log reservations.
- ✓ It will use the resource manager adapter interfaces to make reservations and to delete existing reservations.

(20) Data Access Agreement Service

- ✓ The data access agreement service is created by the agreement factory service with a set of terms, including (but not restricted to) source and destination file path, bandwidth requirements, and fault-tolerance terms (such as retrial times).
- ✓ The data access agreement service allows end users or a job agreement service to stage application or required data.

(21) Queuing Service

- ✓ The queuing service provides scheduling capability for jobs. Given a set of policies defined at the VO level, a queuing service will map jobs to resource managers based on the defined policies.
- ✓ For example, a queuing service might implement a fair-share policy that makes sure that all users within the VO get reasonable turnaround time on their jobs, rather than being starved out by other users' jobs ahead of them in the queue.
- ✓ The manageability interface defines a set of service data for accessing QoS terms supported by the queuing services. QoS terms for the queuing service can include whether the service supports on-line or batch capabilities, average turn-around time for jobs, throughput guarantees, the ability to meet deadlines, and the ability to meet certain economic constraints.

The following terms apply to the queuing service:

- Enqueue—add a job to a queue
- Dequeue—remove a job from a queue

(22) Open Grid Services Infrastructure

- ✓ As we now know, the OGSI defines fundamental mechanisms on which OGSA is constructed. These mechanisms address issues relating to the creation, naming, management, and exchange of information among entities called grid services.
- ✓ A grid service instance is a (potentially transient) service that conforms to a set of conventions (expressed as WSDL interfaces, extensions, and behaviors) for such purposes as lifetime management, discovery of characteristics, and notification. These conventions provide for the controlled management of the distributed and often long-lived state that is commonly required in distributed applications.

- ✓ OGSI also introduces standard factory and registration interfaces for creating and discovering grid services (OGSI was covered in detail in Chapter 4.) The following list recaps the key OGSI features and briefly discusses their relevance to OGSA.

Grid Service descriptions and instances

- ✓ OGSI introduces the twin concepts of the grid service description and grid service instance as organizing principles of distributed systems.
- ✓ A grid service description comprises the WSDL (with OGSI extensions) defining the grid service's interfaces and service data (see next item).
- ✓ A grid service instance is an addressable, potentially stateful, and potentially transient instantiation of such a description. These concepts provide the basic building blocks used to build OGSA-based distributed systems.
- ✓ Grid service descriptions define interfaces and behaviors, and a distributed system comprises a set of grid service instances that implement those behaviors, have a notion of identity with respect to the other instances in the system, and can be characterized as state coupled with behavior published through type-specific operations.

Service state, metadata, and introspection

- ✓ OGSI defines mechanisms for representing and accessing (via both queries and subscriptions) metadata and state data from a service instance (*service data*), as well as providing uniform mechanisms for accessing state.
- ✓ These mechanisms support introspection, in that a client application can ask a grid service instance to return information describing itself, such as the collection of interfaces that it implements.

Naming and name resolution

- ✓ OGSI defines a two-level naming scheme for grid service instances based on abstract, long-lived *grid service handles* thatcan be mapped by HandleMapper services to concrete but potentially lesslong-lived *grid service references.*
- ✓ These constructs are basically networkwide pointers to specific grid service instances hosted in (potentially remote) execution environments. A client application can use a grid service reference to send requests (represented by the operations defined in the interfaces of the target service) directly to the specific instance at the specified network-attached service endpoint identified by the grid service reference.

Fault model

- ✓ OGSI defines a common approach for conveying fault information from operations.

Life cycle

- ✓ OGSI defines mechanisms for managing the life cycle of a grid service instance, including both explicit destruction and soft-state lifetime management functions for grid service instances, and grid service factories that can be used to create instances implementing specified interfaces.

Service groups

- ✓ OGSI defines a means of organizing groups of service instances. OGSI does not address how grid services are created, managed, and destroyed within a particular hosting environment.
- ✓ Thus, services that conform to the OGSI specification are not necessarily portable across various hosting environments, but can be invoked by any client that conforms to this specification, subject, of course, to policy and compatible protocol bindings.
- ✓ Stateful instances, typed interfaces, and global names are frequently also cited as fundamental characteristics of so-called distributed object-based systems.

- ✓ However, various other aspects of distributed object models (as traditionally defined) are specifically not required or prescribed by OGSI. For this reason, one does not use the terms "distributed object model" or "distributed object system" when describing this work, but instead uses the term "open grid services infrastructure," thus emphasizing the connections with both Web services and grid technologies.
- ✓ Among the object-related issues that are not addressed within OGSI are implementation inheritance, service mobility, development approach, and hosting technology.
- ✓ The grid service specification does not require, nor does it prevent, implementations based upon object technologies that support inheritance at either the interface or the implementation level.
- ✓ There is no requirement in the architecture to expose the notion of implementation inheritance either at the client side or the service-provider side of the usage contract. In addition, the grid service specification does not prescribe, dictate, or prevent the use of any particular development approach or hosting technology for the grid service.
- ✓ For example, there is nothingabout OGSI that is Java specific; one can implement OGSI behaviors in C, Python, or other languages. Grid service providers are free to implement the semantic contract of the service in any technology and hosting architecture of their choosing.
- ✓ One can envision implementations in J2EE, .NET, traditional commercial transaction management servers, traditional procedural UNIX servers, and so on. One can also envision service implementations in a variety of programming languages that would include both object-oriented and nonobject-oriented *alternatives.*

(23) Common Management Model

- ✓ A fundamental requirement of grid management infrastructure is the ability to define the resources and resource management functions of the system in a standard and interoperable way.
- ✓ The capabilities of the grid management infrastructure rely on the ability to discover, compose, and interact with the resources and the resources managers responsible for them.
- ✓ The Common Management Model specification defines the base behavioral model for all resources and resource managers in the grid management infrastructure. A mechanism is defined by which resource managers can make use of detailed manageability information for a resource that may come from existing resource models and instrumentation, such as

those expressed in CIM, JMX, SNMP, and so on, combined with a set of canonical operations introduced by base CMM interfaces.

- ✓ The result is a manageable resource abstraction that introduces OGSI-compliant operations over an exposed underlying base-content model. CMM does not define yet another manageability (resource information) model.

The CMM specification defines

- The base manageable resource interface, which a resource or resource manager must provide to be manageable
- Canonical lifecycle states—the transitions between the states, and the operations necessary for the transitions that complement OGSI lifetime service data
- The ability to represent relationships among manageable resources (instances and types), including a canonical set of relationship types
- Life cycle metadata (XML attributes) common to all types of managed resources for monitoring and control of service data and operations based on life cycle state
- Canonical services factored out from across multiple resources or domainspecific resource managers, such as an operational port type (start/stop/pause/ resume/quiesce)

Additional items that may come within the scope of the CMM specification are

- New data types or metadata to convey semantic meaning of manageability information,such as counter or gauge
- Versioning information
- Metadata to associate a metered usage (unit of measure) with manageability information
- Classification of properties such as metric and configuration
- Registries and locating fine-grained resources
- Managed resource identifier

- ✓ To summarize, CMM represents the proposed industry standard for representing manageable resources and resource managers within the grid management infrastructure.

✓ CMM interfaces are OGSI compliant and are used as the base abstract interfaces from which specific manageable resource types are derived. Standardization of the base management behavior is required in order to integrate the vast number and types of resources and more limited set of resource managers introduced by multiple vendors.

IMPORTANT QUESTIONS
PART-A

1. Define OGSA Framework
2. What are the Two key properties of a grid service
3. Define Interoperability of resources
4. List out the Basic Functionality Requirements
5. Give the Resource Management Requirements in OGSA
6. What are the core properties of grid service.
7. Define Data-Intensive Grid Service Models.
8. Explain about Open Grid Services Architecture (OGSA) (Nov 10)
9. What are the Components of the MRG? (Apr 11)
10. What is OGSA services.

PART-B

1. Explain in detail about the Introduction to Open Grid Services Architecture (OGSA)
2. Give the Motivation concept in Open Grid Services Architecture (OGSA)
3. Explain in detail about the Functionality Requirements in Open Grid Services Architecture (OGSA)
4. Explain Practical view of OGSA/OGSI.
5. Explain Detailed view of OGSA/OGSI
6. Explain in detail about the Data intensive grid service models.
7. Briefly explain about the OGSA services.

UNIT I
INTRODUCTION

Evolution of Distributed computing: Scalable computing over the Internet – Technologies for network based systems – clusters of cooperative computers - Grid computing Infrastructures – cloud computing - service oriented architecture – Introduction to Grid Architecture and standards – Elements of Grid – Overview of Grid Architecture.

PART-A

1) Define Cloud Computing with example.

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

OR

Cloud computing is a phrase used to describe a variety of computing concepts that involve a large number of computers connected through a real-time communication network such as the Internet.

For example, Google hosts a cloud that consists of both smallish PCs and larger servers. Google's cloud is a private one (that is, Google owns it) that is publicly accessible (by Google's users).

2) What are the properties of Cloud Computing?

There are six key properties of cloud computing:

- User -centric
- Task -centric
- Powerful
- Accessible
- Intelligent
- Programmable

3) What is the working principle of Cloud Computing?

The cloud is a collection of computers and servers that are publicly accessible via the Internet. This hardware is typically owned and operated by a third party on a consolidated basis in one or more data center locations. The machines can run any combination of operating systems.

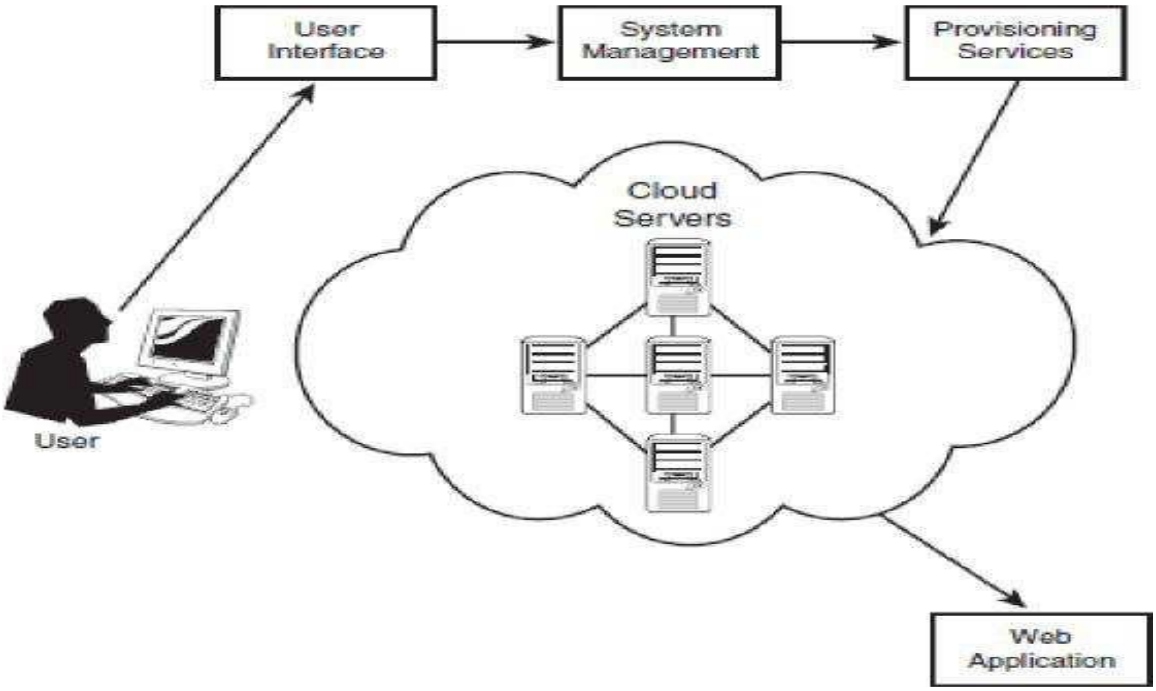
4) Why is Cloud Computing important?

There are many implications of cloud technology, for both developers and end users.

For developers, cloud computing provides increased amounts of storage and processing power to run the applications they develop. Cloud computing also enables new ways to access information, process and analyze data, and connect people and resources from any location anywhere in the world.

For users, documents hosted in the cloud always exist, no matter what happens to the user's machine. Users from around the world can collaborate on the same documents, applications, and projects, in real time. And cloud computing does all this at lower costs, because the cloud enables more efficient sharing of resources than does traditional network computing.

5) Draw the architecture of Cloud



6) What are the advantages and disadvantages of Cloud Computing?

Advantages

- Lower-Cost Computers for Users
- Improved Performance
- Lower IT Infrastructure Costs
- Fewer Maintenance Issues
- Lower Software Costs
- Instant Software Updates
- Increased Computing Power
- Unlimited Storage Capacity
- Increased Data Safety
- Improved Compatibility Between Operating Systems
- Improved Document Format Compatibility
- Easier Group Collaboration
- Universal Access to Documents
- Latest Version Availability
- Removes the Tether to Specific Devices

Disadvantages

- Requires a Constant Internet Connection
- Doesn't Work Well with Low-Speed Connections
- Can Be Slow
- Features Might Be Limited
- Stored Data Might Not Be Secure
- If the Cloud Loses Your Data, You're Screwed

7) What is distributed system?

A *distributed system* is a software system in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal.

Three significant characteristics of distributed systems are:

- ✓ Concurrency of components
- ✓ Lack of a global clock
- ✓ Independent failure of components

8) Write the short notes about centralized computing

Centralized computing is computing done at a central location, using terminals that are attached to a central computer.

The computer itself may control all the peripherals directly (if they are physically connected to the central computer), or they may be attached via a terminal server.

9) Write the short notes about utility computing

- ✓ Utility computing is the packaging of computing resources, such as computation, storage and services, as a metered service. This model has the advantage of a low or no initial cost to acquire computer resources; instead, computational resources are essentially rented.
- ✓ This repackaging of computing services became the foundation of the shift to "on demand" computing, software as a service and cloud computing models that further propagated the idea of computing, application and network as a service.

10) Write the short notes about parallel computing

Parallel computing is a form of computation in which many calculations are carried out simultaneously,operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel").

There are several different forms of parallel computing:

- ✓ Bit -level parallelism.
- ✓ Instruction level parallelism.
- ✓ Data parallelism
- ✓ Task parallelism.

11) Write the short notes about advances in CPU processor

- ✓ Processor speed growth from 1 MIPS for the VAX 780 in 1978 to 1,800 MIPS for the Intel Pentium 4 in 2002, up to a 22,000 MIPS peak for the Sun Niagara 2 in 2008.
- ✓ The clock rate for these processors increased from 10 MHz for the Intel 286 to 4 GHz for the Pentium 4 in 30 years.
- ✓ The clock rate reached its limit on CMOS-based chips due to power limitations. At the time of this writing, very few CPU chips run with a clock rate exceeding 5 GHz.

12) Write the short notes about advances in network bandwidth

- ✓ The rapid growth of Ethernet bandwidth from 10 Mbps in 1979 to 1 Gbps in 1999, and 40 ~ 100 GE in 2011.
- ✓ It has been speculated that 1 Tbps network links will become available by 2013.

13) Write the short notes about advances in hard disks and memory (RAM)

MEMORY

- ✓ The growth of DRAM chip capacity from 16 KB in 1976 to 64 GB in 2011.
- ✓ This shows that memory chips have experienced a 4x increase in capacity every three years. Memory access time did not improve much in the past.

HARD DISK

- ✓ Hard drives, capacity increased from 260 MB in 1981 to 250 GB in 2004.
- ✓ The Seagate Barracuda XT hard drive reached 3 TB in 2011.
- ✓ This represents an approximately 10x increase in capacity every eight years.

14) Write the short notes about GPU

- ✓ A GPU is a graphics coprocessor or accelerator mounted on a computer's graphics card or video card.
- ✓ A GPU offloads the CPU from tedious graphics tasks in video editing applications. The world's first GPU, the GeForce 256, was marketed by NVIDIA in 1999.
- ✓ These GPU chips can pro-cess a minimum of 10 million polygons per second, and are used in nearly every computer on the market today.
- ✓ Some GPU features were also integrated into certain CPUs.

15) Write the short notes about advances GPU power efficiency

- ✓ By extrapolating current technology and computer architecture, it was estimated that 60 Gflops/watt per core is needed to run an exaflops system. Power constrains what we can put in a CPU or GPU chip.
- ✓ Dally has estimated that the CPU chip consumes about 2 nJ/instruction, while the GPU chip requires 200 pJ/instruction, which is 1/10 less than that of the CPU.
- ✓ The CPU is optimized for latency in caches and memory, while the GPU is optimized for throughput with explicit management of on-chip memory.
- ✓ In 2010, the GPU had a value of 5 Gflops/watt at the core level, compared with less than 1 Gflop/watt per CPU core.

16) Define massive system

- ✓ Massive systems are considered highly scalable, and can reach web-scale connectivity, either physically or logically.
- ✓ In massive systems are classified into four groups: clusters, P2P networks, computing grids, and Internet clouds over huge data centers

17) What is cluster?

A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource. In the past, clustered computer systems have demonstrated

18) What is grid computing? (Apr 11)(Nov 10)

Grid Computing enables virtuals organizations to share geographically distributed resources as they pursue common goals, assuming the absence of central location, central control, omniscience, and an existing trust relationship.

(or)

- ✓ Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures.
- ✓ National grid projects are followed by industrial grid plat-form development by IBM, Microsoft, Sun, HP, Dell, Cisco, EMC, Platform Computing, and others. New grid service providers (GSPs) and new grid applications have emerged rapidly, similar to the growth of Internet and web services in the past two decades.
- ✓ grid systems are classified in essentially two categories: computational or data grids and P2P grids.
- ✓ Computing or data grids are built primarily at the national level.

19) What are computational grids?

A computational grid is hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities.

20) What are the business benefits in Grid Computing?

- ✓ Acceleration of implementation time frames in order to intersect with theanticipated business end results.
- ✓ Improved productivity and collaboration of virtual organizations and respective computing and data resources.
- ✓ Allowing widely dispersed departments and business to create virtual organizations to share data and resources.

21) What are the business areas needs in Grid computing?

- ✓ Life Sciences
- ✓ Financial services
- ✓ Higher Education
- ✓ Engineering Services
- ✓ Government
- ✓ Collaborative games

22) List out the Grid Applications:

- ✓ Application partitioning that involves breaking the problem into discrete pieces
- ✓ Discovery and scheduling of tasks and workflow
- ✓ Data communications distributing the problem data where and when it is required
- ✓ Provisioning and distributing application codes to specific system nodes
- ✓ Autonomic features such as self-configuration, self-optimization, self-recovery and self-management

23) List out the grid portal capabilities:

- ✓ Querying databases or LDAP servers for resource-specific information
- ✓ File transfer facilities such as file upload, down load, integration with customsoftware, and so on Manage job through job status feedbacks
- ✓ Allocate the resources for the execution of specific tasks.
- ✓ Security management
- ✓ Provide personalized solutions

24) What are the uses of Integrated Solutions?

The integrated solutions are a combination of the existing advances middleware and application functionalities, combined to provide more coherent and high performance results across the grid Computing environment.

25) What are the areas are difficult to implement in Grid Computing Infrastructure?

A Grid computing infrastructure component must address several potentially complicated areas in many stages of the implementation. These areas are

- ✓ Security
- ✓ Resource management
- ✓ Information services
- ✓ Data management

26) Define grid computing:

Grid concept is defined as control sharing of resources and problem solving in dynamic, multi institutional virtual organization. Grid computing is a open standard. It has well defined policies and conditions to solving the problem.

27) What are the grid applications?

- ✓ Scheduler
- ✓ Resource broker
- ✓ Grid portals
- ✓ Load Balancing
- ✓ Integrated solutions

28) What is meant by grid infrastructure?

Grid infrastructure is a complex combination of a number of capabilities and resources identified for the specific problem and environment being addressed. It forms the core foundations for successful grid applications.

29) List some grid computing toolkits and frameworks?

- ✓ Globus Toolkit Globus Resource Allocation Manager(GRAM)
- ✓ Grid Security Infrastructure(GSI)
- ✓ Information Services
- ✓ Legion, Condor and Condor-G
- ✓ NIMROD, UNICORE, NMI.

30) Write short notes on Legion:

Legion is a middleware project initiated by University of Virginia, is object based meta system software for grid applications.

The goal of legion project is to promote the principled design of distributed system software by providing standard object representations for processors, data systems, file system and so on.

31) What is High Performance computing?

- ✓ High-performance computing generally refers to what has traditionally been called supercomputing. There are hundreds of supercomputers deployed throughout the world.
- ✓ Key parallel processing algorithms have already been developed to support execution of programs on different, but co-located processors.
- ✓ High-performance computing system deployment, contrary to popular belief, is not limited to academic or research institutions.
- ✓ In fact, more than half of supercomputers deployed in the world today are in use at various corporations.

32) Explain about cluster computing. (Nov 10)

- ✓ Cluster computing came about as a response to the high prices of supercomputers, which made those systems out of reach for many research projects.
- ✓ Clusters are high-performance, massively parallel computers built primarily out of commodity hardware components, running a free-software operating system such as Linux or FreeBSD, and interconnected by a private high-speed network.
- ✓ It consists of a cluster of PCs, or workstations, dedicated to running high-performance computing tasks.
- ✓ The nodes in the cluster do not sit on users' desks, but are dedicated to running cluster jobs. A cluster is usually connected to the outside world through only a single node.

33) Explain Peer-to-Peer Computing.

- ✓ Peer-to-Peer (P2P) networks and file sharing into the public eye, methods for transferring files and information between computers have been, in fact, around almost as long as computing itself.
- ✓ Until recently, however, systems for sharing files and information between computers were exceedingly limited. They were largely confined to Local Area Networks (LANs) and the exchange of files with known individuals over the Internet.
- ✓ LAN transfers were executed mostly via a built-in system or network software while Internet file exchanges were mostly executed over an FTP (File Transfer Protocol) connection.
- ✓ The reach of this Peer-to-Peer sharing was limited to the circle of computer users an individual knew and agreed to share files with.
- ✓ Users who wanted to communicate with new or unknown users could transfer files using IRC (Internet Relay Chat) or other similar bulletin boards dedicated to specific subjects, but these methods never gained mainstream popularity because they were somewhat difficult to use.

34) What is Internet Computing?

- ✓ The explosion of the Internet and the increasing power of the home computer prompted computer scientists and engineers to apply techniques learned in high-performance and cluster-based distributed computing to utilize the vast processing cycles available at users' desktops. This has come to be known as Internet computing.

35) Explain Grid Applications Service Providers.

- ✓ The Grid Applications Service Provider (GASP) provides end-to-end Grid Computing services to the user of a particular application or applications.
- ✓ The customer in this case will purchase "application time" from the provider, and will provide the data or parameters to the GASP through an application portal and in the future through published Web service specifications.
- ✓ The GASP may choose to purchase services from the GReP or may choose to build the infrastructure organically

36)What are Desktop Grids? (Apr 11)

- ✓ These are grids that leverage the compute resources of desktop computers.

Because of the true (but unfortunate) ubiquity of Microsoft® Windows® operating system in corporations, desktop grids are assumed to apply to the Windows environment.

The Mac OS™ environment is supported by a limited number of vendors.

37)What are Server Grids?

- ✓ Some corporations, while adopting Grid Computing , keep it limited to server resources that are within the purview of the IT department.

- ✓ Special servers, in some cases, are bought solely for the purpose of creating an internal "utility grid" with resources made available to various departments.
- ✓ No desktops are included in server grids. These usually run some flavor of the Unix/Linux operating system.

38)What are Data Grids?

- ✓ Grid deployments that require access to, and processing of, data are called data grids.
- ✓ They are optimized for data-oriented operations. Although they may consume a lot of storage capacity, these grids are not to be confused with storage service providers.

39)What are Utility Grids?

- ✓ utility grids as being commercial compute resources that are maintained and managed by a service provider. Customers that have the need to augment their existing, internal computational resources may purchase "cycles" from a utility grid.
- ✓ In addition to overflow applications, customers may choose to use utility grids for business continuity and disaster recovery purposes.
- ✓ Utility grid providers are also called Grid Resource Providers (GReP). Along with computing resources, some utility grids also offer key business applications that can be purchased "by the minute."

40) Explain the key components of grid computing

- ✓ **Resource management:** a grid must be aware of what resources are available for different tasks
- ✓ **Security management:** the grid needs to take care that only authorized users can access and use the available resources
- ✓ **Data management:** data must be transported, cleansed, parceled and processed
- ✓ **Services management:** users and applications must be able to query the grid in an effective and efficient manner

41) What is Cyber-Physical Systems(CPS)

- ✓ A cyber-physical system (CPS) is the result of interaction between computational processes and the physical world. A CPS integrates "cyber" (heterogeneous, asynchronous) with "physical" (concurrent and information-dense) objects.
- ✓ A CPS merges the "3C" technologies of **computation, communication, and control** into an intelligent closed feedback system between the physical world and the information world, a concept which is actively explored in the United States.

42) List out the technologies for Network-Based Systems .

a) Multicore CPUs and Multithreading Technologies
b) GPU Computing to Exascale and Beyond
c) Memory, Storage, and Wide-Area Networking
d) Virtual Machines and Virtualization Middleware
e) Data Center Virtualization for Cloud Computing

43) What is Virtual Infrastructures

- ✓ Physical resources for compute, storage, and networking at the bottom are mapped to the needy applications embedded in various VMs at the top. Hardware and software are then separated.

- ✓ Virtual infrastructure is what connects resources to distributed applications. It is a dynamic mapping of system resources to specific applications.
- ✓ Depicts the cloud landscape and major cloud players, based on three cloud service models.
    - Infrastructure as a Service (IaaS)
    - Platform as a Service (PaaS)
    - Software as a Service (SaaS)

44) What is Infrastructure as a Service (IaaS)

- ✓ This model puts together infrastructures demanded by users—namely servers, storage, networks, and the data center fabric.
- ✓ The user can deploy and run on multiple VMs running guest OSes on specific applications.
- ✓ The user does not manage or control the underlying cloud infrastructure, but can specify when to request and release the needed resources.

45) What is Platform as a Service (PaaS)

- ✓ This model enables the user to deploy user-built applications onto a virtualized cloud platform. PaaS includes middleware, databases, development tools, and some runtime support such as Web 2.0 and Java.
- ✓ The platform includes both hardware and software integrated with specific programming interfaces. The provider supplies the API and software tools (e.g., Java, Python, Web 2.0, .NET). The user is freed from managing the cloud infrastructure.

46) What is Software as a Service (SaaS)

- ✓ This refers to browser-initiated application software over thousands of paid cloud customers. The SaaS model applies to business processes, industry applications, consumer relationship management (CRM), enterprise resources planning (ERP), human resources (HR), and collaborative applications.
- ✓ On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications.

47) List the highlights eight reasons to adapt the cloud for upgraded Internet applications and web services:

1. Desired location in areas with protected space and higher energy efficiency
2. Sharing of peak-load capacity among a large pool of users, improving overall utilization
3. Separation of infrastructure maintenance duties from domain-specific application development
4. Significant reduction in cloud computing cost, compared with traditional computing paradigms
5. Cloud computing programming and application development
6. Service and data discovery and content/service distribution
7. Privacy, security, copyright, and reliability issues
8. Service agreements, business models, and pricing policies

48) Give the component layers of the architecture with specific capabilities at each layer. Each layer shares the behavior of the component layers described in the next discussion.

- ✓ (i) Fabric Layer: Interface to Local Resources
- ✓ (ii) The Connectivity layer

- ✓ (iii) Resource Layer: Sharing of a Single Resource
- ✓ (iv) The Collective Layer: Coordinating Multiple Resources
- ✓ (v) Application Layer: User-Defined Grid Applications

PART-B

1. Suggest ideas and defend your argument with some plausible solutions that describe Scalable computing over the internet.

- ✓ The Age of Internet Computing. .
- ✓ Scalable Computing Trends and New Paradigms.
- ✓ The Internet of Things and Cyber-Physical Systems

The Age of Internet Computing

- ✓ Billions of people use the Internet every day. As a result, supercomputer sites and large data centers must provide high-performance computing services to huge numbers of Internet users concurrently.
- ✓ Because of this high demand, the Linpack Benchmark for high-performance computing **(HPC)**applications is no longer optimal for measuring system performance. The emergence of computing clouds instead demands high-throughput computing **(HTC)** systems built with parallel and distributed computing technologies .
- ✓ We have to upgrade data centers using fast servers, storage systems, and high-bandwidth networks. The purpose is to advance network-based computing and web services with the emerging new technologies.

- ✓ The Platform Evolution
- ✓ High-Performance Computing
- ✓ High-Throughput Computing
- ✓ Three New Computing Paradigms
- ✓ Computing Paradigm Distinctions
- ✓ Distributed System Families

1. The Platform Evolution

- ✓ Computer technology has gone through five generations of development, with each generation lasting from 10 to 20 years. Successive generations are overlapped in about 10 years.
- ✓ For instance, from 1950 to 1970, a handful of mainframes, including the IBM 360 and CDC 6400, were built to satisfy the demands of large businesses and government organizations.
- ✓ From 1960 to 1980, lower-cost minicomputers such as the DEC PDP 11 and VAX Series became popular among small businesses and on college campuses.
- ✓ From 1970 to 1990, we saw widespread use of personal computers built with VLSI microprocessors.
- ✓ From 1980 to 2000, massive numbers of portable computers and pervasive devices appeared in both wired and wireless applications.
- ✓ Since 1990, the use of both HPC and HTC systems hidden in clusters, grids, or Internet clouds has proliferated. These systems are employed by both consumers and high-end web-scale computing and information services.
- ✓ The general computing trend is to leverage shared web resources and massive amounts of data over the Internet.

On the HPC side, supercomputers (massively parallel processors or MPPs) are gradually replaced by clusters of cooperative computers out of a desire to share computing resources. The cluster is often a collection of homogeneous compute nodes that are physically connected in close range to one another.

On the HTC side, peer-to-peer (P2P) networks are formed for distributed file sharing and content delivery applications. A P2P system is built over many client machines.

Peer machines are globally distributed in nature. P2P, cloud computing, and web service platforms are more focused on HTC applications than on HPC applications.

Clustering and P2P technologies lead to the development of computational grids or data grids.

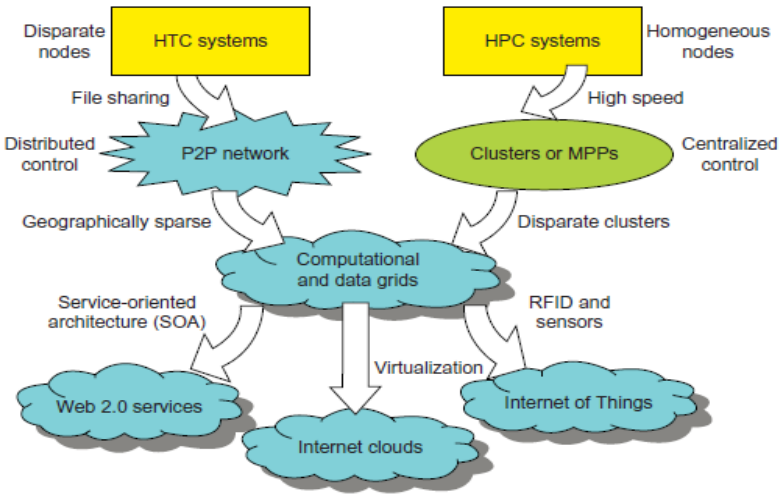


FIGURE 1.1

Evolutionary trend toward parallel, distributed, and cloud computing with clusters, MPPs, P2P networks, grids, clouds, web services, and the Internet of Things.

2. High-Performance Computing(HPC)

- ✓ For many years, HPC systems emphasize the raw speed performance. The speed of HPC systems has increased from Gflops in the early 1990s to now Pflops in 2010. This improvement was driven mainly by the demands from scientific, engineering, and manufacturing communities.

For example, the Top 500 most powerful computer systems in the world are measured by floating-point speed in Linpack benchmark results.

However, the number of supercomputer users is limited to less than 10% of all computer users. Today, the majority of computer users are using desktop computers or large servers when they conduct Internet searches and market-driven computing tasks.

3. High-Throughput Computing(HTC)

- ✓ The development of market-oriented high-end computing systems is undergoing a strategic change from an HPC paradigm to an HTC paradigm. This HTC paradigm pays more attention to high-flux computing.
- ✓ The main application for high-flux computing is in Internet searches and web services by millions or more users simultaneously. The performance goal thus shifts to measure high throughput or the number of tasks completed per unit of time.
- ✓ HTC technology needs to not only improve in terms of batch processing speed, but also address the acute problems of cost, energy savings, security, and reliability at many data and enterprise computing centers.

4. Three New Computing Paradigms

- ✓ Advances in virtualization make it possible to see the growth of Internet clouds as a new computing paradigm are as follows

- The maturity of radio-frequency identification **(RFID)**
- Global Positioning System **(GPS)**
- sensor technologies has triggered the development of the Internet of Things **(IoT)**

5 Computing Paradigm Distinctions

- ✓ The high-technology community has argued for many years about the precise definitions of centralized computing, parallel computing, distributed computing, and cloud computing.
- ✓ In general, distributed computing is the opposite of centralized computing. The field of parallel computing overlaps with distributed computing to a great extent, and cloud computing overlaps with distributed, centralized, and parallel computing.

Centralized computing

- ✓ **T**his is a computing paradigm by which all computer resources are centralized in one physical system. All resources (processors, memory, and storage) are fully shared and tightly coupled within one integrated OS. Many data centers and supercomputers are centralized systems, but they are used in parallel, distributed, and cloud computing applications

Parallel computing

- ✓ In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory.
- ✓ Interprocessor communication is accomplished through shared memory or via message passing. A computer system capable of parallel computing is commonly known as a parallel computer . Programs running in a parallel computer are called parallel programs. The process of writing parallel programs is often referred to as parallel programming .

Distributed computing

- ✓ This is a field of computer science/engineering that studies distributed systems. A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network.
- ✓ Information exchange in a distributed system is accomplished through message passing. A computer program that runs in a distributed system is known as a distributed program.
- ✓ The process of writing distributed programs is referred to as distributed programming.

Cloud computing

- ✓ An Internet cloud of resources can be either a centralized or a distributed computing system. The cloud applies parallel or distributed computing, or both. Clouds can be built with physical or virtualized resources over large data centers that are centralized or distributed.
- ✓ As an alternative to the preceding terms, some in the high-tech community prefer the term concurrent computing or concurrent programming. These terms typically refer to the union of parallel computing and distributing computing, although biased practitioners may interpret them differently.
- ✓ Ubiquitous computing refers to computing with pervasive devices at any place and time using wired or wireless communication. The Internet of Things (IoT) is a networked connection of everyday objects including computers, sensors, humans, etc.
- ✓ The IoT is supported by Internet clouds to achieve ubiquitous computing with any object at any place and time. Finally, the term Internet computing is even broader and covers all computing paradigms over the Internet.

6. Distributed System Families

- ✓ In the future, both HPC and HTC systems will demand multicore or many-core processors that can handle large numbers of computing threads per core. Both HPC and HTC systems emphasize parallelism and distributed computing.
- ✓ Future HPC and HTC systems must be able to satisfy this huge demand in computing power in terms of throughput, efficiency, scalability, and reliability.
- ✓ The system efficiency is decided by speed, programming, and energy factors (i.e., throughput per watt of energy consumed). Meeting these goals requires to yield the following design objectives:

• **Efficiency measures** the utilization rate of resources in an execution model by exploiting massive parallelism in HPC. For HTC, efficiency is more closely related to job throughput, dataaccess, storage, and power efficiency.

• **Dependability measures** the reliability and self-management from the chip to the system and application levels. The purpose is to provide high-throughput service with Quality of Service (QoS) assurance, even under failure conditions.

• **Adaptation in the programming model measures** the ability to support billions of job requests over massive data sets and virtualized cloud resources under various workload and service models.

• **Flexibility in application deployment measures** the ability of distributed systems to run well in both HPC (science and engineering) and HTC (business) applications.

2. Briefly define the following basic techniques and technologies that represent the Scalable Computing Trends and New Paradigms

Several predictable trends in technology are known to drive computing applications. In fact, designers and programmers want to predict the technological capabilities of future systems.

- ✓ Degrees of Parallelism
- ✓ Innovative Applications
- ✓ The Trend toward Utility Computing
- ✓ The Hype Cycle of New Technologies

Let's review the degrees of parallelism before we discuss the special requirements for distributed computing.

1. Degrees of Parallelism

- ✓ Fifty years ago, when hardware was bulky and expensive, most computers were designed in a bit-serial fashion. In this scenario, **bit-level parallelism (BLP)** converts bit-serial processing to word-level processing gradually.
- ✓ Over the years, users graduated from 4-bit microprocessors to 8-,16-, 32-, and 64-bit CPUs. This led us to the next wave of improvement, known as **instruction-level parallelism (ILP)**, in which the processor executes multiple instructions simultaneously rather than only one instruction at a time.
- ✓ For the past 30 years, we have practiced ILP through **pipelining, superscalar computing, VLIW (very long instruction word) architectures, and multithreading**. ILP requires branch prediction, dynamic scheduling, speculation, and compiler support to work efficiently.

- ✓ **Data-level parallelism (DLP)** was made popular through SIMD (single instruction, multiple data) and vector machines using vector or array types of instructions. DLP requires even more hardware support and compiler assistance to work properly.
- ✓ Ever since the introduction of multicore processors and chip multiprocessors (CMPs), we have been exploring **task-level parallelism (TLP)**.
- ✓ A modern processor explores all of the aforementioned parallelism types. In fact, BLP, ILP, and DLP are well supported by advances in hardware and compilers. However, **TLP is far from being very successful** due to difficulty in programming and compilation of code for efficient execution on multicore CMPs.
- ✓ As we move from parallel *processing* to distributed processing, we will see an increase in computing granularity to job-level parallelism (JLP). It is fair to say that coarse-grain parallelism is built on top of fine-grain parallelism.

2. Innovative Applications

- ✓ Both HPC and HTC systems desire transparency in many application aspects. For example, data access, resource allocation, process location, concurrency in execution, job replication, and failure recovery should be made transparent to both users and system management.
- ✓ Table 1.1 highlights a few key applications that have driven the development of parallel and distributed systems over the years.
- ✓ These applications spread across many important domains in **science, engineering, business, education, health care, traffic control, Internet and web services, military, and government applications**.
- ✓ Almost all applications demand computing economics, web-scale data collection, system reliability, and scalable performance.
- ✓ For example, distributed transaction processing is often practiced in the banking and finance industry. Transactions represent 90 percent of the existing market for reliable banking systems. Users must deal with multiple database servers in distributed transactions.
- ✓ Maintaining the consistency of replicated transaction records is crucial in real-time banking services.Other complications include lack of software support, network saturation, and security threats in these applications.

Table 1.1 Applications of High-Performance and High-Throughput Systems

Domain	Specific Applications
Science and engineering	Scientific simulations, genomic analysis, etc.
	Earthquake prediction, global warming, weather forecasting, etc.
Business, education, services industry, and health care	Telecommunication, content delivery, e-commerce, etc.
	Banking, stock exchanges, transaction processing, etc.
	Air traffic control, electric power grids, distance education, etc.
	Health care, hospital automation, telemedicine, etc.
Internet and web services, and government applications	Internet search, data centers, decision-making systems, etc.
	Traffic monitoring, worm containment, cyber security, etc.
	Digital government, online tax return processing, social networking, etc.
Mission-critical applications	Military command and control, intelligent systems, crisis management, etc.

3 .The Trend toward Utility Computing

- ✓ Figure 1.2 identifies major computing paradigms to facilitate the study of distributed systems and their applications. These paradigms share some common characteristics.
- ✓ First, they are all ubiquitous in daily life. Reliability and scalability are two major design objectives in these computing models.
- ✓ Second, they are aimed at autonomic operations that can be self-organized to support dynamic discovery.
- ✓ Finally, these paradigms are composable with QoS and SLAs (service-level agreements).
- ✓ These paradigms and their attributes realize the computer utility vision.
  Utility computing focuses on a business model in which customers receive computing
  Resources from a paid service provider.
- ✓ All grid/cloud platforms are regarded as utility service providers.However, cloud computing offers a broader concept than utility computing. Distributed cloud applications run on any available servers in some edge networks.
- ✓ Major technological challenges include all aspects of computer science and engineering.
- ✓ For example, users demand new network efficient processors, scalable memory and storage schemes, distributed OSes, middleware for machine virtualization, new programming models, effective resource management, and application program development.
- ✓ These hardware and software supports are necessary to build distributed systems that explore massive parallelism at all processing levels.

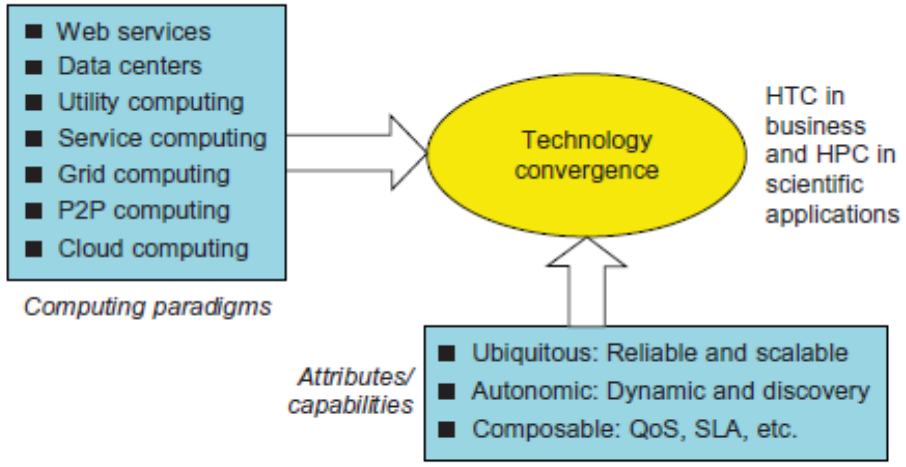


FIGURE 1.2
The vision of computer utilities in modern distributed computing systems.

4. The Hype Cycle of New Technologies

- ✓ Any new and emerging computing and information technology may go through a hype cycle, as illustrated in Figure 1.3. This cycle shows the expectations for the technology at five different stages. The expectations rise sharply from the trigger period to a high peak of inflated expectations.

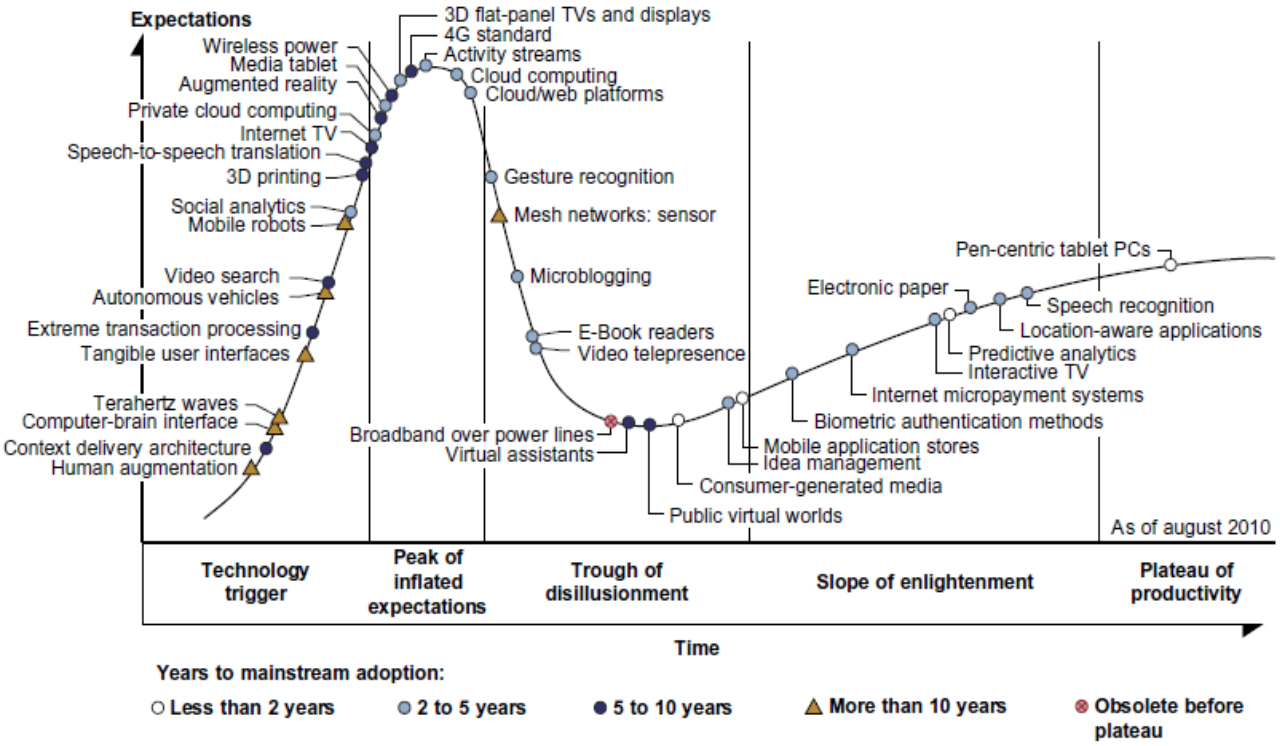


FIGURE 1.3

Hype cycle for Emerging Technologies, 2010.

- ✓ Through a short period of disillusionment, the expectation may drop to a valley and then increase steadily over a long enlightenment period to a plateau of productivity. The number of years for an emerging technology to reach a certain stage is marked by special symbols.
- ✓ The hollow circles indicate technologies that will reach mainstream adoption in two years. The gray circles represent technologies that will reach mainstream adoption in two to five years. The solid circles represent those that require five to 10 years to reach mainstream adoption, and the triangles denote those that require more than 10 years. The crossed circles represent technologies that will become obsolete before they reach the plateau.
- ✓ The hype cycle in Figure 1.3 shows the technology status as of August 2010. For example, at that time consumer-generated media was at the disillusionment stage, and it was predicted to take less than two years to reach its plateau of adoption.
- ✓ Internet micropayment systems were forecast to take two to five years to move from the enlightenment stage to maturity.
- ✓ It was believed that 3D printing would take five to 10 years to move from the rising expectation stage to mainstream adoption, and mesh network sensors were expected to take more than 10 years to move from the inflated expectation stage to a plateau of mainstream adoption.
- ✓ Also as shown in Figure 1.3, the cloud technology had just crossed the peak of the expectation stage in 2010, and it was expected to take two to five more years to reach the productivity stage.
- ✓ However, broadband over power line technology was expected to become obsolete before leaving the valley of disillusionment stage in 2010.
- ✓ Many additional technologies (denoted by dark circles in Figure 1.3) were at their peak expectation stage in August 2010, and they were expected to take five to 10 years to reach their plateau of success.
- ✓ Once a technology begins to climb the slope of enlightenment, it may reach the productivity plateau within two to five years.

- ✓ Among these promising technologies are the clouds, biometric authentication, interactive TV, speech recognition, predictive analytics, and media tablets.

3. Explain in detail about The Internet of Things and Cyber-Physical Systems

In this section, we will discuss two Internet development trends:

i. The Internet of Things
ii. Cyber-Physical Systems

(i) The Internet of Things

- ✓ The traditional Internet connects machines to machines or web pages to web pages. The IoT refers to the networked interconnection of everyday objects, tools, devices, or computers.
- ✓ One can view the IoT as a wireless network of sensors that interconnect all things in our daily life. These things can be large or small and they vary with respect to time and place.
- ✓ The idea is to tag every object using RFID or a related sensor or electronic technology such as GPS. With the introduction of the IPv6 protocol, 2128 IP addresses are available to distinguish all the objects on Earth, including all computers and pervasive devices.
- ✓ The IoT researchers have estimated that every human being will be surrounded by 1,000 to 5,000 objects. The IoT needs to be designed to track 100 trillion static or moving objects simultaneously.
- ✓ The IoT demands universal addressability of all of the objects or things. To reduce the complexity of identification, search, and storage, one can set the threshold to filter out fine-grain objects. The IoT obviously extends the Internet and is more heavily developed in Asia and European countries.
- ✓ In the IoT era, all objects and devices are instrumented, interconnected, and interacted with each other intelligently. This communication can be made between people and things or among the things themselves.

  Three communication patterns co-exist: namely
  - H2H (human-to-human),
  - H2T (human-tothing),
  - T2T (thing-to-thing).

- ✓ Here things include machines such as PCs and mobile phones. The idea here is to connect things (including human and machine objects) at any time and any place intelligently with low cost. **Any place connections** include at the PC, indoor (away from PC), outdoors, and on the move. **Any time connections** include daytime, night, outdoors and indoors, and on the move as well.
- ✓ The dynamic connections will grow exponentially into a new dynamic network of networks, called the Internet of Things **(IoT).**
- ✓ The IoT is still in its infancy stage of development. Many prototype IoTs with restricted areas of coverage are under experimentation at the time of this writing.
- ✓ Cloud computing researchers expect to use the cloud and future Internet technologies to support fast, efficient, and intelligent interactions among humans, machines, and any objects on Earth.
- ✓ A smart Earth should have intelligent cities, clean water, efficient power, convenient transportation, good food supplies, responsible banks, fast telecommunications, green IT, better schools, good health care, abundant resources, and so on. This dream living environment may take some time to reach fruition at different parts of the world.

2 Cyber-Physical Systems(CPS)

- ✓ A cyber-physical system (CPS) is the result of interaction between computational processes and the physical world. A CPS integrates "cyber" (heterogeneous, asynchronous) with "physical" (concurrent and information-dense) objects.
- ✓ A CPS merges the "3C" technologies of computation, communication, and control into an intelligent closed feedback system between the physical world and the information world, a concept which is actively explored in the United States.
- ✓ The IoT emphasizes various networking connections among physical objects, while the CPS emphasizes exploration of virtual reality (VR) applications in the physical world.
- ✓ We may transform how we interact with the physical world just like the Internet transformed how we interact with the virtual world.

5. Briefly explain the Technologies for Network-Based Systems for Scalable computing over the Internet.

- ✓ With the concept of scalable computing under our belt, it's time to explore hardware, software, and network technologies for distributed computing system design and applications. In particular, we will focus on viable approaches to building distributed operating systems for handling massive parallelism in a distributed environment.

The Technologies for Network-Based Systems are as follows,

f) Multicore CPUs and Multithreading Technologies
g) GPU Computing to Exascale and Beyond
h) Memory, Storage, and Wide-Area Networking
i) Virtual Machines and Virtualization Middleware
j) Data Center Virtualization for Cloud Computing

a) Multicore CPUs and Multithreading Technologies

- Advances in CPU Processors
- Multicore CPU and Many-Core GPU Architectures
- Multithreading Technology

- ✓ Consider the growth of component and network technologies over the past 30 years. They are crucial to the development of HPC and HTC systems.
- ✓ In Figure 1.4, processor speed is measured in millions of instructions per second (MIPS) and network bandwidth is measured in megabits per second (Mbps) or gigabits per second (Gbps). The unit GE refers to 1 Gbps Ethernet bandwidth.

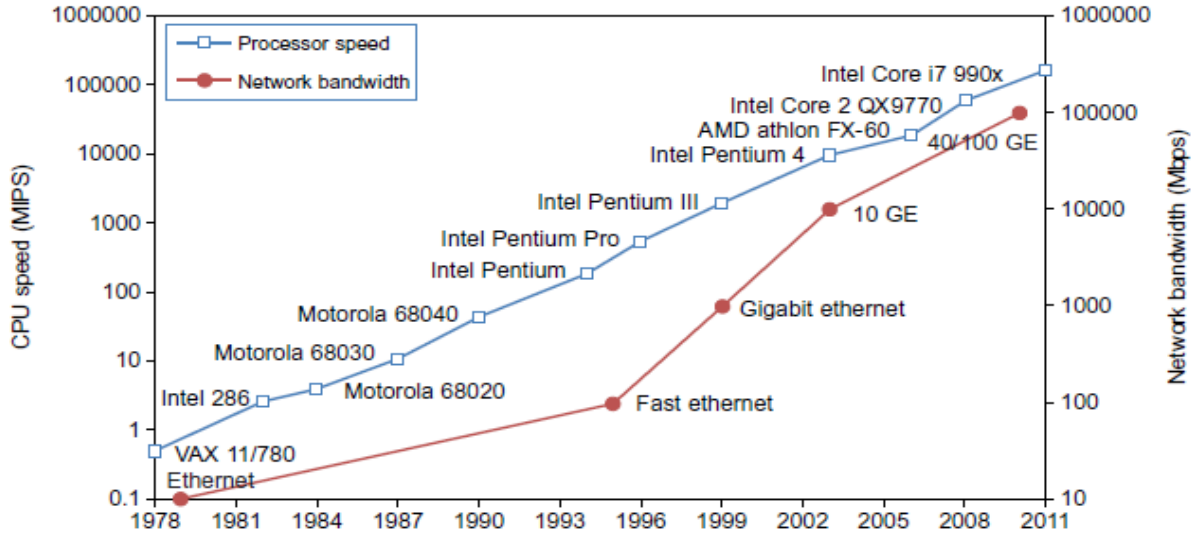


FIGURE 1.4

Improvement in processor and network technologies over 33 years.

(*Courtesy of Xiaosong Lou and Lizhong Chen of University of Southern California, 2011*)

Advances in CPU Processors

- ✓ Today, advanced CPUs or microprocessor chips assume a multicore architecture with dual, quad, six, or more processing cores. These processors exploit parallelism at ILP and TLP levels. Processor speed growth is plotted in the upper curve in Figure 1.4 across generations of microprocessors or CMPs.
- ✓ We see growth from 1 MIPS for the VAX 780 in 1978 to 1,800 MIPS for the Intel Pentium 4 in 2002, up to a 22,000 MIPS peak for the Sun Niagara 2 in 2008. As the figure shows, Moore's law has proven to be pretty accurate in this case. The clock rate for these processors increased from 10 MHz for the Intel 286 to 4 GHz for the Pentium 4 in 30 years.
- ✓ However, the clock rate reached its limit on CMOS-based chips due to power limitations. At the time of this writing, very few CPU chips run with a clock rate exceeding 5 GHz.
- ✓ In other words, clock rate will not continue to improve unless chip technology matures. This limitation is attributed primarily to excessive heat generation with high frequency or high voltages. The ILP is highly exploited in modern CPU processors.
- ✓ ILP mechanisms include multiple-issue superscalar architecture, dynamic branch prediction, and speculative execution, among others. These ILP techniques demand hardware and compiler support.
- ✓ In addition, DLP and TLP are highly explored in graphics processing units (GPUs) that adopt a many-core architecture with hundreds to thousands of simple cores.

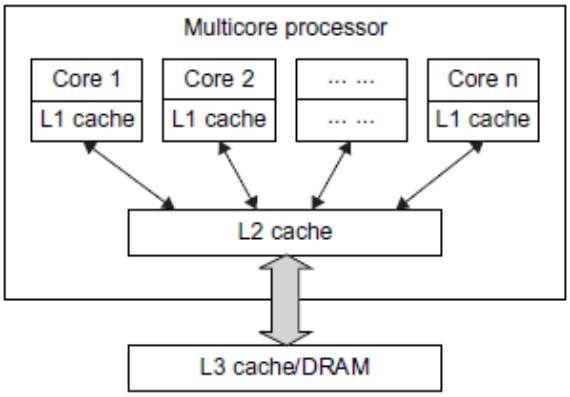


FIGURE 1.5

Schematic of a modern multicore CPU chip using a hierarchy of caches, where L1 cache is private to each core, on-chip L2 cache is shared and L3 cache or DRAM Is off the chip.

- ✓ Both multi-core CPU and many-core GPU processors can handle multiple instruction threads at different magnitudes today. Figure 1.5 shows the architecture of a typical multicore processor.
- ✓ Each core is essentially a processor with its own private cache (L1 cache). Multiple cores are housed in the same chip with an L2 cache that is shared by all cores.
- ✓ In the future, multiple CMPs could be built on the same CPU chip with even the L3 cache on the chip. Multicore and multithreaded CPUs are equipped with many high-end processors, including the Intel i7, Xeon, AMD Opteron, Sun Niagara, IBM Power 6, and X cell processors. Each core could be also multithreaded.

For example, the Niagara II is built with eight cores with eight threads handled by each core. This implies that the maximum ILP and TLP that can be exploited in Niagara is 64 (8 × 8 = 64). In 2011, the Intel Core i7 990x has reported 159,000 MIPS execution rate as shown in the uppermost square in Figure 1.4.

Multicore CPU and Many-Core GPU Architectures

- ✓ Multicore CPUs may increase from the tens of cores to hundreds or more in the future. But the CPU has reached its limit in terms of exploiting massive DLP due to the aforementioned memory wall problem.
- ✓ This has triggered the development of many-core GPUs with hundreds or more thin cores. Both IA-32 and IA-64 instruction set architectures are built into commercial CPUs. Now, x-86 processors have been extended to serve HPC and HTC systems in some high-end server processors.
- ✓ Many RISC processors have been replaced with multicore x-86 processors and many-core GPUs in the Top 500 systems. This trend indicates that x-86 upgrades will dominate in data centers and supercomputers.
- ✓ The GPU also has been applied in large clusters to build supercomputers in MPPs. In the future, the processor industry is also keen to develop asymmetric or heterogeneous chip multiprocessors that can house both fat CPU cores and thin GPU cores on the same chip.

Multithreading Technology

- ✓ Consider in Figure 1.6 the dispatch of five independent threads of instructions to four pipelined data paths (functional units) in each of the following five processor categories, from left to right: a four-issue superscalar processor, a fine-grain multithreaded processor, a coarse-grain multithreaded processor, a two-core CMP, and a simultaneous multithreaded (SMT) processor. The superscalar processor is single-threaded with four functional units.

- ✓ Each of the three multithreaded processors is four-way multithreaded over four functional data paths. In the dual-core processor,assume two processing cores, each a single-threaded two-way superscalar processor.
- ✓ Instructions from different threads are distinguished by specific shading patterns for instructions from five independent threads. Typical instruction scheduling patterns are shown here.
- ✓ Only instructions from the same thread are executed in a superscalar processor. Fine-grain multithreading switches the execution of instructions from different threads per cycle. Course-grain multithreading executes many instructions from the same thread for quite a few cycles before switching to another thread.
- ✓ The multicore CMP executes instructions from different threads completely.The SMT allows simultaneous scheduling of instructions from different threads in the same cycle. These execution patterns closely mimic an ordinary program. The blank squares correspond to no available instructions for an instruction data path at a particular processor cycle.
- ✓ More blank cells imply lower scheduling efficiency. The maximum ILP or maximum TLP is difficult to achieve at each processor cycle. The point here is to demonstrate your understanding of typical instruction scheduling patterns in these five different micro-architectures in modern processors.

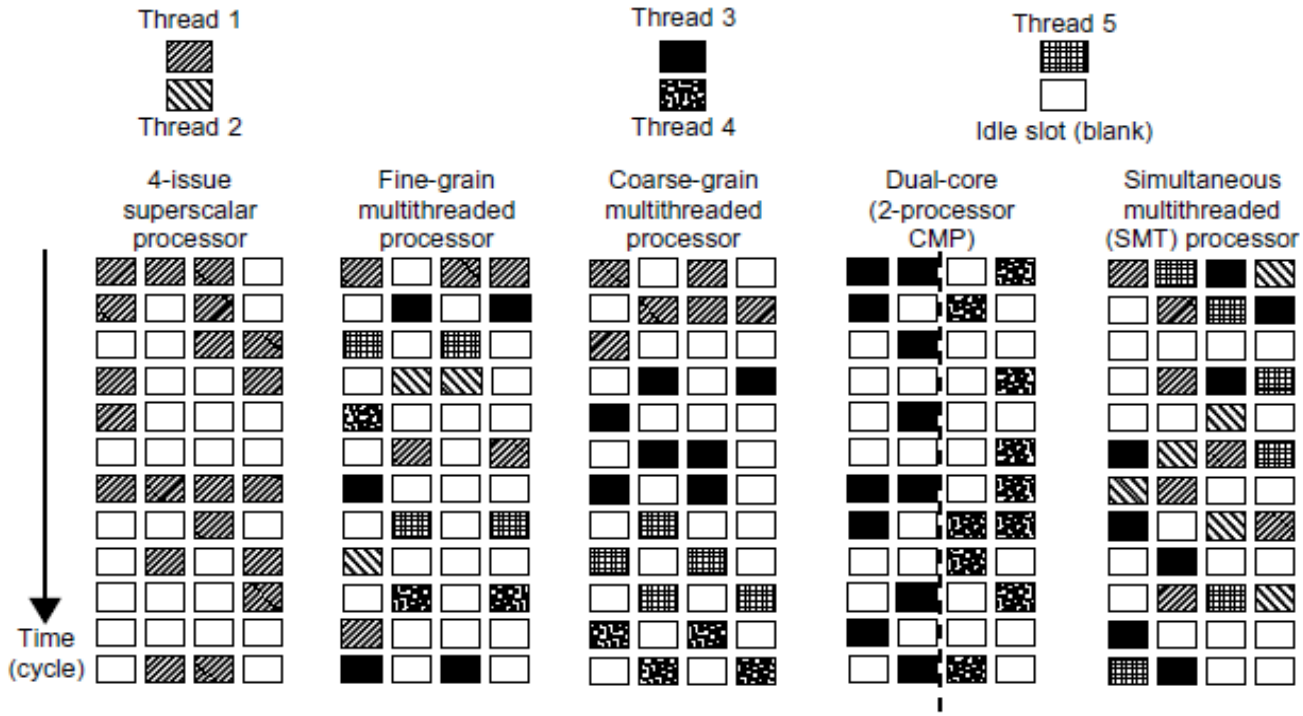


FIGURE 1.6

Five micro-architectures in modern CPU processors, that exploit ILP and TLP supported by multicore and multithreading technologies.

b) GPU Computing to Exascale and Beyond

- How GPUs Work
- GPU Programming Model
- Power Efficiency of the GPU

How GPUs Work

- ✓ A GPU is a graphics coprocessor or accelerator mounted on a computer's graphics card or video card. A GPU offloads the CPU from tedious graphics tasks in video editing applications. The world's first GPU, the GeForce 256, was marketed by NVIDIA in 1999.
- ✓ These GPU chips can process a minimum of 10 million polygons per second, and are used in nearly every computer on the market today. Some GPU features were also integrated into certain CPUs. Traditional CPUs are structured with only a few cores.
- ✓ For example, the Xeon X5670 CPU has six cores. However, a modern GPU chip can be built with hundreds of processing cores.
- ✓ Unlike CPUs, GPUs have a throughput architecture that exploits massive parallelism by executing many concurrent threads slowly, instead of executing a single long thread in a conventional microprocessor very quickly.
- ✓ Lately, parallel GPUs or GPU clusters have been garnering a lot of attention against the use of CPUs with limited parallelism. General-purpose computing on GPUs, known as GPGPUs, have appeared in the HPC field. NVIDIA's CUDA model was for HPC using GPGPUs.

- ✓ Early GPUs functioned as coprocessors attached to the CPU. Today, the NVIDIA GPU has been upgraded to 128 cores on a single chip. Furthermore, each core on a GPU can handle eight threads of instructions.
- ✓ This translates to having up to 1,024 threads executed concurrently on a single GPU. This is true massive parallelism, compared to only a few threads that can be handled by a conventional CPU. The CPU is optimized for latency caches, while the GPU is optimized to deliver much higher throughput with explicit management of on-chip memory.
- ✓ Modern GPUs are not restricted to accelerated graphics or video coding. They are used in HPC systems to power supercomputers with massive parallelism at multicore and multithreading levels.
- ✓ GPUs are designed to handle large numbers of floating-point operations in parallel. In a way, the GPU offloads the CPU from all data-intensive calculations, not just those that are related to video processing.
- ✓ Conventional GPUs are widely used in mobile phones, game consoles, embedded systems, PCs, and servers. The NVIDIA CUDA Tesla or Fermi is used in GPU clusters or in HPC systems for parallel processing of massive floating-pointing data.

GPU Programming Model

- ✓ Figure 1.7 shows the interaction between a CPU and GPU in performing parallel execution of floating-point operations concurrently. The CPU is the conventional multicore processor with limited parallelism to exploit.
- ✓ The GPU has a many-core architecture that has hundreds of simple processing cores organized as multiprocessors. Each core can have one or more threads. Essentially, the CPU's floating-point kernel computation role is largely offloaded to the many-core GPU. The CPU instructs the GPU to perform massive data processing.
- ✓ The bandwidth must be matched between the on-board main memory and the on-chip GPU memory.This process is carried out in NVIDIA's CUDA programming using the GeForce 8800 or Tesla and Fermi GPUs.

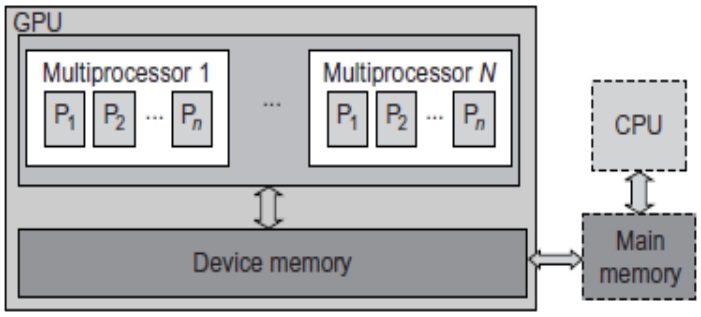


FIGURE 1.7

The use of a GPU along with a CPU for massively parallel execution in hundreds or thousands of processing cores.

Power Efficiency of the GPU

- ✓ Bill Dally of Stanford University considers power and massive parallelism as the major benefits of GPUs over CPUs for the future. By extrapolating current technology and computer architecture, it was estimated that 60 Gflops/watt per core is needed to run an exaflops system (see Figure 1.10).

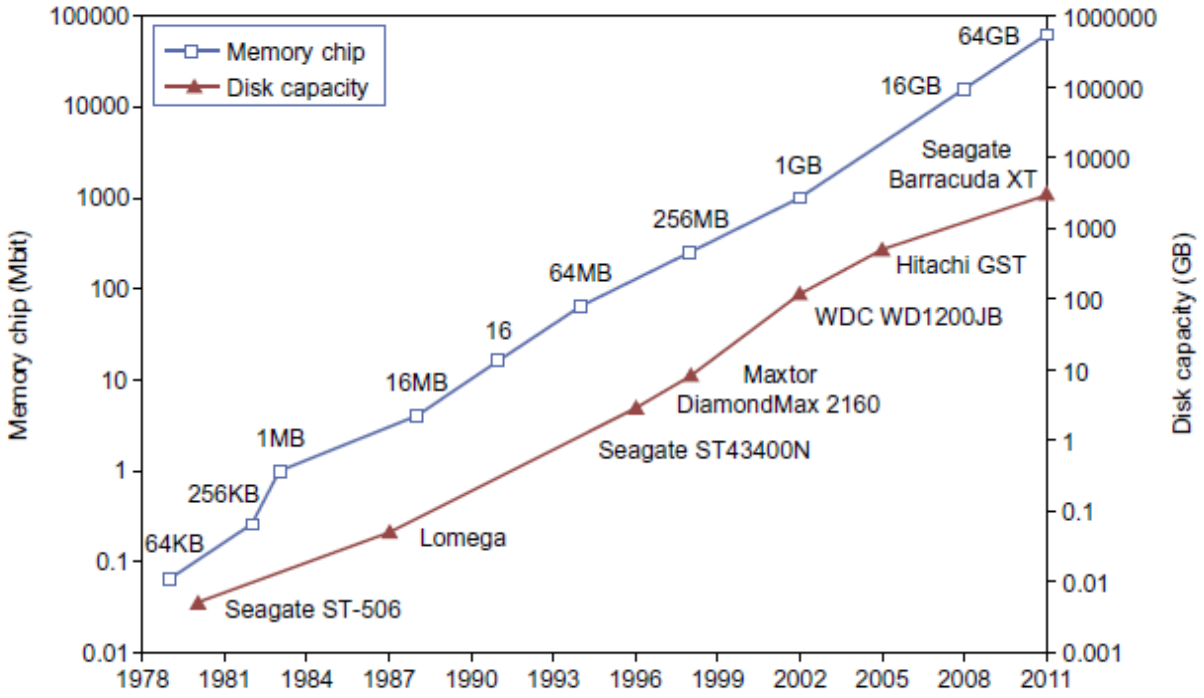


FIGURE 1.10

Improvement in memory and disk technologies over 33 years. The Seagate Barracuda XT disk has a capacity of 3 TB in 2011

- ✓ Power constrains what we can put in a CPU or GPU chip. Dally has estimated that the CPU chip consumes about 2 nJ/instruction, while the GPU chip requires 200 pJ/instruction, which is 1/10 less than that of the CPU.
- ✓ The CPU is optimized for latency in caches and memory, while the GPU is optimized for throughput with explicit management of on-chip memory.

- ✓ Figure 1.9 compares the CPU and GPU in their performance/power ratio measured in Gflops/watt per core. In 2010, the GPU had a value of 5 Gflops/watt at the core level, compared with less than 1 Gflop/watt per CPU core.
- ✓ This may limit the scaling of future supercomputers. However, the GPUs may close the gap with the CPUs. Data movement dominates power consumption. One needs to optimize the storage hierarchy and tailor the memory to the applications.
- ✓ We need to promote self-aware OS and runtime support and build locality-aware compilers and auto-tuners for GPU based MPPs. This implies that both power and software are the real challenges in future parallel and distributed computing systems.

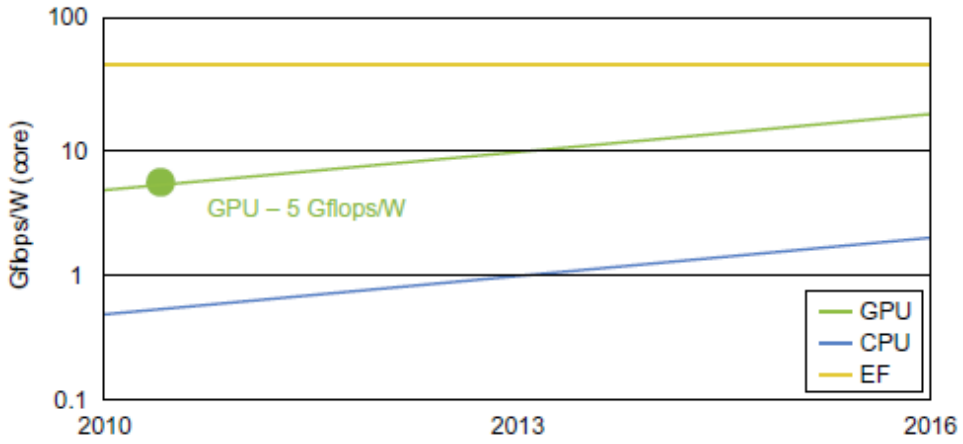


FIGURE 1.9

The GPU performance (middle line, measured 5 Gflops/W/core in 2011), compared with the lower CPU performance (lower line measured 0.8 Gflops/W/core in 2011) and the estimated 60 Gflops/W/core performance in 2011 for the Exascale (EF in upper curve) in the future.

c) Memory, Storage, and Wide-Area Networking

- Memory Technology
- Disks and Storage Technology
- System-Area Interconnects
- Wide-Area Networking

Memory Technology

- ✓ The upper curve in Figure 1.10 plots the growth of DRAM chip capacity from 16 KB in 1976 to 64 GB in 2011. This shows that memory chips have experienced a 4x increase in capacity every three years.
- ✓ Memory access time did not improve much in the past. In fact, the memory wall problem is getting worse as the processor gets faster. For hard drives, capacity increased from 260 MB in 1981 to 250 GB in 2004.
- ✓ The Seagate Barracuda XT hard drive reached 3 TB in 2011. This represents an approximately 10x increase in capacity every eight years. The capacity increase of disk arrays will be even greater in the years to come.
- ✓ Faster processor speed and larger memory capacity result in a wider gap between processors and memory. The memory wall may become even worse a problem limiting the CPU performance in the future.

Disks and Storage Technology

- ✓ Beyond 2011, disks or disk arrays have exceeded 3 TB in capacity. The lower curve in Figure 1.10shows the disk storage growth in 7 orders of magnitude in 33 years. The rapid growth of flash memory and solid-state drives (SSDs) also impacts the future of HPC and HTC systems. The mortality rate of SSD is not bad at all.
- ✓ A typical SSD can handle 300,000 to 1 million write cycles per block. So the SSD can last for several years, even under conditions of heavy write usage. Flash and SSD will demonstrate impressive speedups in many applications.

System-Area Interconnects

- ✓ The nodes in small clusters are mostly interconnected by an Ethernet switch or a local area network(LAN). As Figure 1.11 shows, a LAN typically is used to connect client hosts to big servers.
- ✓ A storage area network (SAN) connects servers to network storage such as disk arrays. Network attached storage (NAS) connects client hosts directly to the disk arrays. All three types of networks often appear in a large cluster built with commercial network components.
- ✓ If no large distributed storage is shared, a small cluster could be built with a multiport Gigabit Ethernet switch plus copper cables to link the end machines. All three types of networks are commercially available.

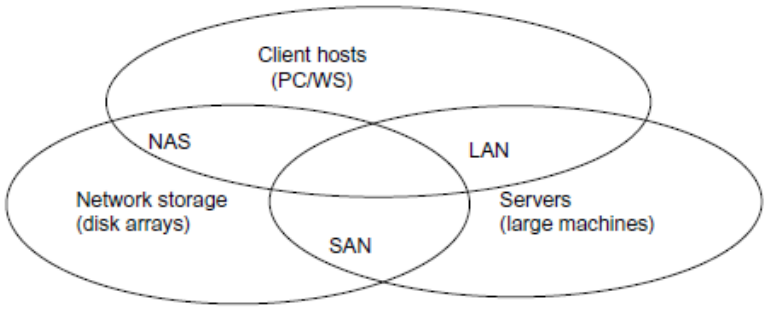


FIGURE 1.11

Three interconnection networks for connecting servers, client hosts, and storage devices; the LAN connects client hosts and servers, the SAN connects servers with disk arrays, and the NAS connects clients with large storage systems in the network environment.

Wide-Area Networking

- ✓ The lower curve in Figure 1.10 plots the rapid growth of Ethernet bandwidth from 10 Mbps in 1979 to 1 Gbps in 1999, and 40 ~ 100 GE in 2011. It has been speculated that 1 Tbps network links will become available by 2013.
- ✓ According to Berman, Fox, and Hey , network links with 1,000, 1,000, 100, 10, and 1 Gbps bandwidths were reported, respectively, for international, national, organization, optical desktop, and copper desktop connections in 2006.
- ✓ An increase factor of two per year on network performance was reported, which is faster than Moore's law on CPU speed doubling every 18 months. The implication is that more computers will be used concurrently in the future.
- ✓ High-bandwidth networking increases the capability of building massively distributed systems. The IDC 2010 report predicted that both InfiniBand and Ethernet will be the two major interconnect choices in the HPC arena. Most data centers are using Gigabit Ethernet as the interconnect in their server clusters.

d) Virtual Machines and Virtualization Middleware

- Virtual Machines
- VM Primitive Operations
- Virtual Infrastructures

✓ A conventional computer has a single OS image. This offers a rigid architecture that tightly couples application software to a specific hardware platform. Some software running well on one machine may not be executable on another platform with a different instruction set under a fixed OS.

✓ Virtual machines (VMs) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines.

✓ Today, to build large clusters, grids, and clouds, we need to access large amounts of computing, storage, and networking resources in a virtualized manner.

✓ We need to aggregate those resources,and hopefully, offer a single system image. In particular, a cloud of provisioned resources must rely on virtualization of processors, memory, and I/O facilities dynamically. Figure 1.12 illustrates the architectures of three VM configurations.

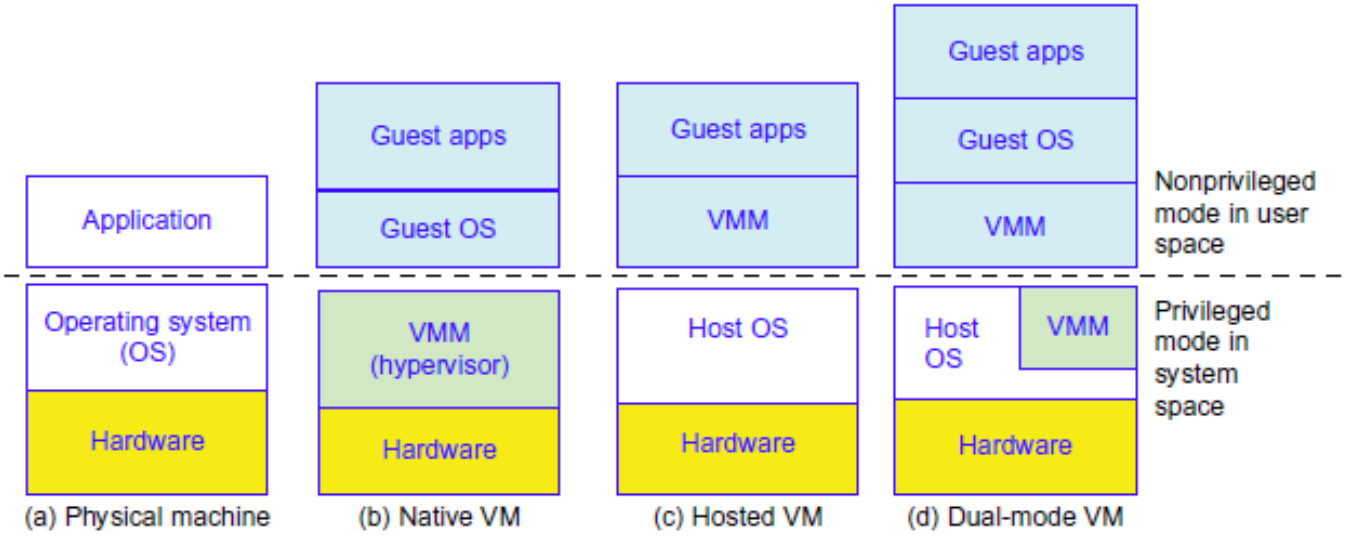


FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

Virtual Machines

✓ In Figure 1.12, the host machine is equipped with the physical hardware, as shown at the bottom of the figure. An example is an x-86 architecture desktop running its installed Windows OS, as shown in part (a) of the figure.

✓ The VM can be provisioned for any hardware system. The VM is built with virtual resources managed by a guest OS to run a specific application. Between the VMs and the host platform, one needs to deploy a middleware layer called a virtual machine monitor (VMM).

✓ Figure 1.12(b) shows a native VM installed with the use of a VMM called a hypervisor in privileged mode. For example, the hardware has x-86 architecture running the Windows system.

✓ The guest OS could be a Linux system and the hypervisor is the XEN system developed at Cambridge University. This hypervisor approach is also called bare-metal VM, because the hypervisor handles the bare hardware (CPU, memory, and I/O) directly.

✓ Another architecture is the host VM shown in Figure 1.12(c). Here the VMM runs in nonprivileged mode.

- ✓ The host OS need not be modified. The VM can also be implemented with a dual mode, as shown in Figure 1.12(d). Part of the VMM runs at the user level and another part runs at the supervisor level. In this case, the host OS may have to be modified to some extent.
- ✓ Multiple VMs can be ported to a given hardware system to support the virtualization process. The VM approach offers hardware independence of the OS and applications. The user application running on its dedicated OS could be bundled together as a virtual appliance that can be ported to any hardware platform. The VM could run on an OS different from that of the host computer.

VM Primitive Operations

- ✓ The VMM provides the VM abstraction to the guest OS. With full virtualization, the VMM exports a VM abstraction identical to the physical machine so that a standard OS such as Windows 2000 or Linux can run just as it would on the physical hardware.

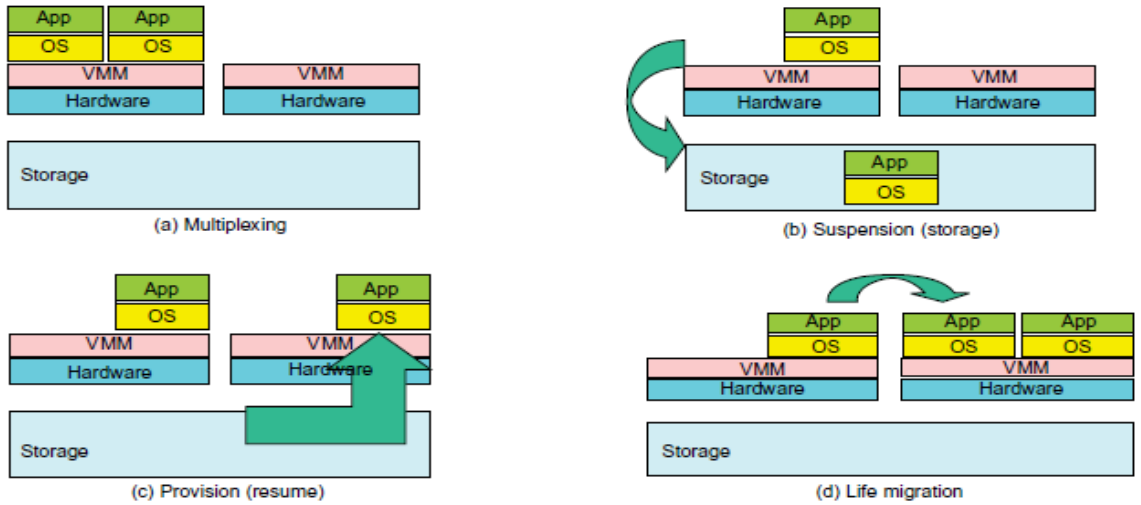


FIGURE 1.13

VM multiplexing, suspension, provision, and migration in a distributed computing environment.

- ✓ First, the VMs can be multiplexed between hardware machines, as shown in Figure 1.13(a).
- ✓ Second, a VM can be suspended and stored in stable storage, as shown in Figure 1.13(b).
- ✓ Third, a suspended VM can be resumed or provisioned to a new hardware platform, as shown in Figure 1.13(c).
- ✓ Finally, a VM can be migrated from one hardware platform to another, as shown in Figure 1.13(d).
- ✓ These VM operations enable a VM to be provisioned to any available hardware platform. They also enable flexibility in porting distributed application executions.

Furthermore, the VM approach will significantly enhance the utilization of server resources. Multiple server functions can be consolidated on the same hardware platform to achieve higher system efficiency.

This will eliminate server sprawl via deployment of systems as VMs, which move transparency to the shared hardware.With this approach, VMware claimed that server utilization could be increased from its current 5–15 percent to 60–80 percent.

Virtual Infrastructures

- ✓ Physical resources for compute, storage, and networking at the bottom of Figure 1.14 are mapped to the needy applications embedded in various VMs at the top. Hardware and software are then separated.
- ✓ Virtual infrastructure is what connects resources to distributed applications. It is a dynamic mapping of system resources to specific applications.

- ✓ The result is decreased costs and increased efficiency and responsiveness. Virtualization for server consolidation and containment is a goodexample of this.

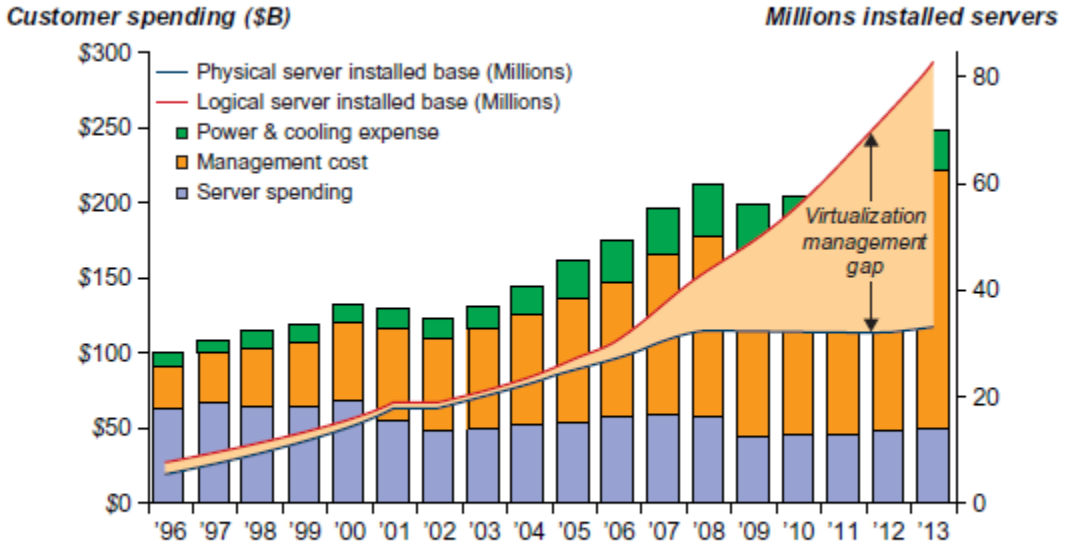


FIGURE 1.14

Growth and cost breakdown of data centers over the years.

e) Data Center Virtualization for Cloud Computing

- Data Center Growth and Cost Breakdown
- Low-Cost Design Philosophy
- Convergence of Technologies

- ✓ In this section, we discuss basic architecture and design considerations of data centers. Cloud architecture is built with commodity hardware and network devices. Almost all cloud platforms choose the popular x86 processors.
- ✓ Low-cost terabyte disks and Gigabit Ethernet are used to build data centers. Data center design emphasizes the performance/price ratio over speed performance alone. In other words, storage and energy efficiency are more important than shear speed performance.
- ✓ Figure 1.13 shows the server growth and cost breakdown of data centers over the past 15 years. Worldwide, about 43 million servers are in use as of 2010. The cost of utilities exceeds the cost of hardware after three years.

Data Center Growth and Cost Breakdown

- ✓ A large data center may be built with thousands of servers. Smaller data centers are typically built with hundreds of servers. The cost to build and maintain data center servers has increased over the years.
- ✓ According to a 2009 IDC report (see Figure 1.14), typically only 30 percent of data center costs goes toward purchasing IT equipment (such as servers and disks), 33 percent is attributed to the chiller, 18 percent to the uninterruptible power supply (UPS), 9 percent to computer room air conditioning (CRAC), and the remaining 7 percent to power distribution, lighting, and transformer costs. Thus, about 60 percent of the cost to run a data center is allocated to management and maintenance.
- ✓ The server purchase cost did not increase much with time. The cost of electricity and cooling did increase from 5 percent to 14 percent in 15 years.

Low-Cost Design Philosophy

- ✓ High-end switches or routers may be too cost-prohibitive for building data centers. Thus, using high-bandwidth networks may not fit the economics of cloud computing. Given a fixed budget, commodity switches and networks are more desirable in data centers.
- ✓ Similarly, using commodity x86 servers is more desired over expensive mainframes. The software layer handles network traffic balancing, fault tolerance, and expandability. Currently, nearly all cloud computing data centers use Ethernet as their fundamental network technology.

Convergence of Technologies

Essentially, cloud computing is enabled by the convergence of technologies in four areas:
(1) Hardware virtualization and multi-core chips,
(2) utility and grid computing,
(3) SOA, Web 2.0, and WS mashups,
(4) Autonomic computing and data center automation.

- ✓ Hardware virtualization and multicore chips enable the existence of dynamic configurations in the cloud. Utility and grid computing technologies lay the necessary foundation for computing clouds.
- ✓ Recent advances in SOA, Web 2.0, and mashups of platforms are pushing the cloud another step forward. Finally, achievements in autonomic computing and automated data center operations contribute to the rise of cloud computing.
- ✓ By linking computer science and technologies with scientists, a spectrum of e-science or e-research applications in biology, chemistry, physics, the social sciences, and the humanities has generated new insights from interdisciplinary activities.
- ✓ **Cloud computing is a transformative approach** as it promises much more than a data center model. It fundamentally changes how we interact with information. The cloud provides services on demand at the infrastructure, platform, or software level.
- ✓ At the platform level, MapReduce offers a new programming model that transparently handles data parallelism with natural fault tolerance capability.
- ✓ Iterative MapReduce extends MapReduce to support a broader range of data mining algorithms commonly used in scientific applications. The cloud runs on an extremely large cluster of commodity computers.
- ✓ Internal to each cluster node, multithreading is practiced with a large number of cores in many-core GPU clusters. Data-intensive science, cloud computing, and multicore computing are converging and revolutionizing the next generation of computing in architectural design and programming challenges. They enable the pipeline: Data becomes information and knowledge, and in turn becomes machine wisdom as desired in SOA.

6. Explain in detail about the Clusters of Cooperative Computers

- ✓ A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource. In the past, clustered computer systems have demonstrated impressive results in handling heavy workloads with large data sets.
    - Cluster Architecture
    - Single-System Image
    - Hardware, Software, and Middleware Support

Cluster Architecture

- ✓ Figure 1.15 shows the architecture of a typical server cluster built around a low-latency, highbandwidth interconnection network. This network can be as simple as a SAN (e.g., Myrinet) or a LAN (e.g., Ethernet).

- ✓ To build a larger cluster with more nodes, the interconnection network can be built with multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches. Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes.
- ✓ The cluster is connected to the Internet via a virtual private network (VPN) gateway. The gateway IP address locates the cluster. The system image of a computer is decided by the way the OS manages the shared cluster resources. Most clusters have loosely coupled node computers. All resources of a server node are managed by their own OS.
- ✓ Thus, most clusters have multiple system images as a result of having many autonomous nodes under different OS control.

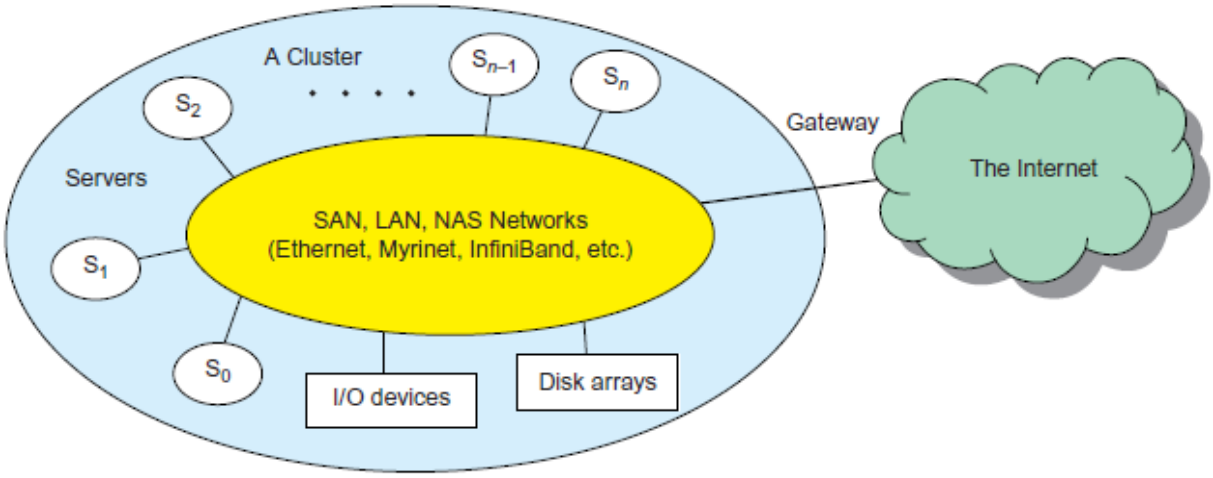


FIGURE 1.15

A cluster of servers interconnected by a high-bandwidth SAN or LAN with shared I/O devices and disk arrays; the cluster acts as a single computer attached to the Internet.

Single-System Image

- ✓ Greg Pfister has indicated that an ideal cluster should merge multiple system images into a single-system image (SSI). Cluster designers desire a cluster operating system or some middleware to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes.
- ✓ An SSI is an illusion created by software or hardware that presents a collection of resources as one integrated, powerful resource. SSI makes the cluster appear like a single machine to the user. A cluster with multiple system images is nothing but a collection of independent computers.

Hardware, Software, and Middleware Support

- ✓ Clusters exploring massive parallelism are commonly known as MPPs. Almost all HPC clusters in the Top 500 list are also MPPs. The building blocks are computer nodes (PCs, workstations, servers, or SMP), special communication software such as PVM or MPI, and a network interface card in each computer node.
- ✓ Most clusters run under the Linux OS. The computer nodes are interconnected by a high-bandwidth network (such as Gigabit Ethernet, Myrinet, InfiniBand, etc.). Special cluster middleware supports are needed to create SSI or high availability (HA).
- ✓ Both sequential and parallel applications can run on the cluster, and special parallel environments are needed to facilitate use of the cluster resources. For example, distributed memory has multiple images.

- ✓ Users may want all distributed memory to be shared by all servers by forming distributed shared memory (DSM). Many SSI features are expensive or difficult to achieve at various cluster operational levels.
- ✓ Instead of achieving SSI, many clusters are loosely coupled machines.Using virtualization, one can build many virtual clusters dynamically, upon user demand.
- ✓ Middleware or OS extensions were developed at the user space to achieve SSI at selected functional levels. Without this middleware, cluster nodes cannot work together effectively to achieve cooperative computing. The software environments and applications must rely on the middleware to achieve high performance.

7. Explian in detail about the Grid Computing Infrastructures

- ✓ Internet services such as the Telnet command enables a local computer to connect to a remote computer.
- ✓ A web service such as HTTP enables remote access of remote web pages.
- ✓ Grid computing is envisioned to allow close interaction among applications running on distant computers simultaneously.
- ✓ Forbes Magazine has projected the global growth of the IT-based economy from $1 trillion in 2001 to $20 trillion by 2015. The evolution from Internet to web and grid services is certainly playing a major role in this growth.

a) Computational Grids
b) Grid Families

a) Computational Grids

- ✓ Like an electric utility power grid, a computing grid offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together. The grid is often constructed across LAN, WAN, or Internet backbone networks at a regional, national, or global scale.
- ✓ Enterprises or organizations present grids as integrated computing resources. They can also be viewed as virtual platforms to support virtual organizations. The computers used in a grid are primarily workstations, servers, clusters, and supercomputers. Personal computers, laptops, and PDAs can be used as access devices to a grid system.
- ✓ Figure 1.16 shows an example computational grid built over multiple resource sites owned by different organizations. The resource sites offer complementary computing resources, including workstations, large servers, a mesh of processors, and Linux clusters to satisfy a chain of computational needs.
- ✓ The grid is built across various IP broadband networks including LANs and WANs already used by enterprises or organizations over the Internet. The grid is presented to users as an integrated resource pool as shown in the upper half of the figure.
- ✓ Special instruments may be involved such as using the radio telescope in SETI@ Home search of life in the galaxy and the austrophysics@Swineburne for pulsars.
- ✓ At the server end, the grid is a network. At the client end, we see wired or wireless terminal devices.
- ✓ The grid integrates the computing, communication, contents, and transactions as rented services. Enterprises and consumers form the user base, which then defines the usage trends and service characteristics.

b)Grid Families

- ✓ Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures. National grid projects are followed by industrial grid platform development by IBM, Microsoft, Sun, HP, Dell, Cisco, EMC, Platform Computing, and others.

- ✓ New grid service providers (GSPs) and new grid applications have emerged rapidly, similar to the growth of Internet and web services in the past two decades.
- ✓ In Table 1.4, grid systems are classified in essentially two categories:
  - computational or data grids
  - P2P grids
- ✓ Computing or data grids are built primarily at the national level.

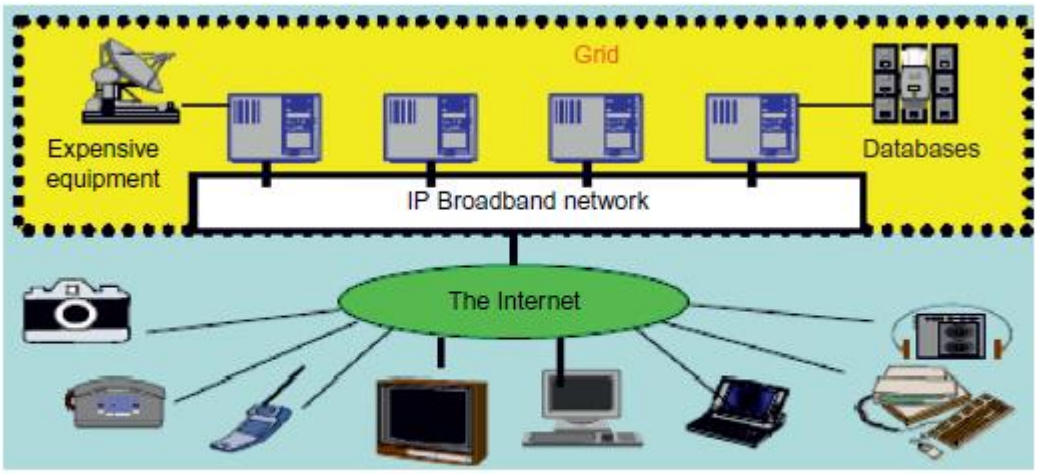


FIGURE 1.16

Computational grid or data grid providing computing utility, data, and information services through resource sharing and cooperation among participating organizations.

Table 1.4 Two Grid Computing Infrastructures and Representative Systems

Design Issues	Computational and Data Grids	P2P Grids
Grid Applications Reported	Distributed supercomputing, National Grid initiatives, etc.	Open grid with P2P flexibility, all resources from client machines
Representative Systems	TeraGrid built in US, ChinaGrid in China, and the e-Science grid built in UK	JXTA, FightAid@home, SETI@home
Development Lessons Learned	Restricted user groups, middleware bugs, protocols to acquire resources	Unreliable user-contributed resources, limited to a few apps

8. Explain in detail about the Cloud computing over the Internet

- ✓ Cloud computing has been defined differently by many users and designers. For example, IBM, a major player in cloud computing, has defined it as follows: **"A cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads, including batch-style backend jobs and interactive and user-facing applications." Based on this definition, a cloud allows workloads to be deployed and scaled out quickly through rapid provisioning of virtual or physical machines.**
- ✓ The cloud supports redundant, self-recovering, highly scalable programming models that allow workloads to recover from many unavoidable hardware/software failures. Finally, the cloud system should be able to monitor resource use in real time to enable rebalancing of allocations when needed.

a) Internet Clouds
b) The Cloud Landscape
c) Eight reasons to adapt cloud for upgraded Internet applications and web services

a)Internet Clouds

- ✓ Cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and data sets dynamically (see Figure 1.18). The idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers.
- ✓ Cloud computing leverages its low cost and simplicity to benefit both users and providers. Machine virtualization has enabled such cost-effectiveness. Cloud computing intends to satisfy many user applications simultaneously.
- ✓ The cloud ecosystem must be designed to be secure, trustworthy, and dependable. Some computer users think of the cloud as a centralized resource pool. Others consider the cloud to be a server cluster which practices distributed computing over all the servers used.

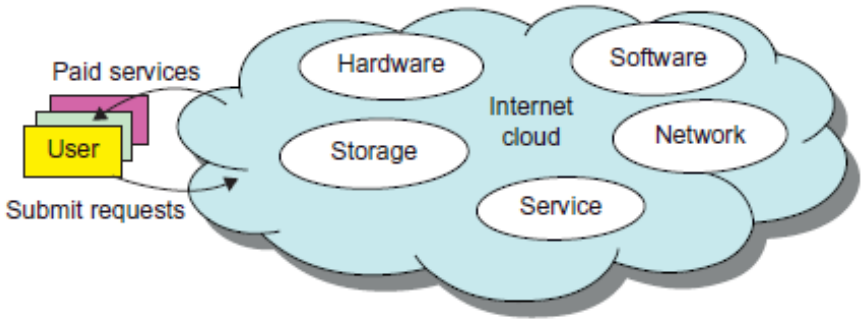


FIGURE 1.18

Virtualized resources from data centers to form an Internet cloud, provisioned with hardware, software, storage, network, and services for paid users to run their applications.

b)The Cloud Landscape

- ✓ Traditionally, a distributed computing system tends to be owned and operated by an autonomous administrative domain (e.g., a research laboratory or company) for on-premises computing needs.
- ✓ However, these traditional systems have encountered several performance bottlenecks: constant system maintenance, poor utilization, and increasing costs associated with hardware/software upgrades. Cloud computing as an on-demand computing paradigm resolves or relieves us from these problems.
- ✓ Figure 1.19 depicts the cloud landscape and major cloud players, based on three cloud service models.
    - Infrastructure as a Service (IaaS)
    - Platform as a Service (PaaS)
    - Software as a Service (SaaS)

Infrastructure as a Service (IaaS)

- ✓ This model puts together infrastructures demanded by users—namely servers, storage, networks, and the data center fabric.
- ✓ The user can deploy and run on multiple VMs running guest OSes on specific applications.
- ✓ The user does not manage or control the underlying cloud infrastructure, but can specify when to request and release the needed resources.

Platform as a Service (PaaS)

- ✓ This model enables the user to deploy user-built applications onto a virtualized cloud platform. PaaS includes middleware, databases, development tools, and some runtime support such as Web 2.0 and Java.
- ✓ The platform includes both hardware and software integrated with specific programming interfaces. The provider supplies the API and software tools (e.g., Java, Python, Web 2.0, .NET). The user is freed from managing the cloud infrastructure.

Software as a Service (SaaS)

- ✓ This refers to browser-initiated application software over thousands of paid cloud customers. The SaaS model applies to business processes, industry applications, consumer relationship management (CRM), enterprise resources planning (ERP), human resources (HR), and collaborative applications.
- ✓ On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications.

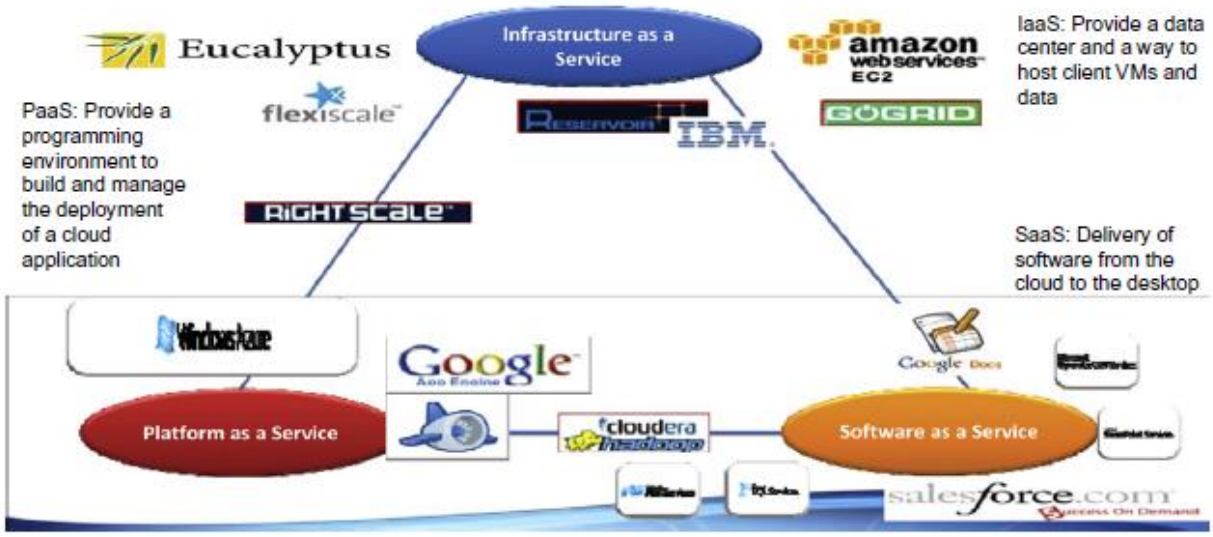


FIGURE 1.19

Three cloud service models in a cloud landscape of major providers.

- ✓ Internet clouds offer four deployment modes: private, public, managed, and hybrid [11]. These modes demand different levels of security implications. The different SLAs imply that the security responsibility is shared among all the cloud providers, the cloud resource consumers, and the thirdparty cloud-enabled software providers.
- ✓ Advantages of cloud computing have been advocated by many IT experts, industry leaders, and computer science researchers.

c) The following list highlights eight reasons to adapt the cloud for upgraded Internet applications and web services:

1. Desired location in areas with protected space and higher energy efficiency
2. Sharing of peak-load capacity among a large pool of users, improving overall utilization
3. Separation of infrastructure maintenance duties from domain-specific application development
4. Significant reduction in cloud computing cost, compared with traditional computing

paradigms
5. Cloud computing programming and application development
6. Service and data discovery and content/service distribution
7. Privacy, security, copyright, and reliability issues
8. Service agreements, business models, and pricing policies

8. Explain in detail about the Service-Oriented Architecture (SOA)

- ✓ In grids/web services, Java, and CORBA, an entity is, respectively, a service, a Java object, and a CORBA distributed object in a variety of languages. These architectures build on the traditional seven Open Systems Interconnection (OSI) layers that provide the base networking abstractions.
- ✓ On top of this we have a base software environment, which would be .NET or Apache Axis for web services, the Java Virtual Machine for Java, and a broker network for CORBA.
- ✓ On top of this base environment one would build a higher level environment reflecting the special features of the distributed computing environment.
- ✓ This starts with entity interfaces and inter-entity communication, which rebuild the top four OSI layers but at the entity and not the bit level. Figure 1.20 shows the layered architecture for distributed entities used in web services and grid systems.
    a) Layered Architecture for Web Services and Grids
    b) Web Services and Tools
    c) The Evolution of SOA
    d) Grids versus Clouds

a) Layered Architecture for Web Services and Grids

- ✓ The entity interfaces correspond to the Web Services Description Language (WSDL), Java method, and CORBA interface definition language (IDL) specifications in these example distributed systems.
- ✓ These interfaces are linked with customized, high-level communication systems: SOAP, RMI, and IIOP in the three examples. These communication systems support features including particular message patterns (such as Remote Procedure Call or RPC), fault recovery, and specialized routing.
- ✓ Often, these communication systems are built on message-oriented middleware (enterprise bus) infrastructure such as Web- Sphere MQ or Java Message Service (JMS) which provide rich functionality and support virtualization of routing, senders, and recipients.
- ✓ In the case of fault tolerance, the features in the Web Services Reliable Messaging (WSRM) framework mimic the OSI layer capability (as in TCP fault tolerance) modified to match the different abstractions (such as messages versus packets, virtualized addressing) at the entity levels.
- ✓ Security is a critical capability that either uses or reimplements the capabilities seen in concepts such as Internet Protocol Security (IPsec) and secure sockets in the OSI layers.

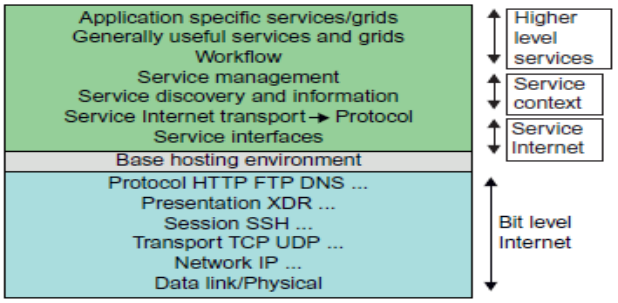


FIGURE 1.20
Layered achitecture for web services and the grids.

- ✓ The above language or interface terms form a collection of entity-level capabilities.The latter can have performance advantages and offers a "shared memory" model allowing more convenient exchange of information.
- ✓ However, the distributed model has two critical advantages:namely
  - o Higher performance (from multiple CPUs when communication is unimportant)
  - o Cleaner separation of software functions with clear software reuse and maintenance advantages.
- ✓ The distributed model is expected to gain popularity as the default approach to software systems. In the earlier years, CORBA and Java approaches were used in distributed systems rather than today's SOAP, XML, or REST (Representational State Transfer).

b) Web Services and Tools

- ✓ Loose coupling and support of heterogeneous implementations make services more attractive thandistributed objects.

  Figure 1.20 corresponds to two choices of service architecture:
  - o Web services
  - o REST systems
- ✓ Both web services and REST systemshave very distinct approaches to building reliable interoperable systems.
- ✓ In web services,one aims to fully specify all aspects of the service and its environment. This specification is carried with communicated messages using Simple Object Access Protocol (SOAP).
- ✓ The hosting environment then becomes a universal distributed operating system with fully distributed capability carried by SOAP messages. This approach has mixed success as it has been hard to agree on key parts of the protocol and even harder to efficiently implement the protocol by software such as Apache Axis.
- ✓ In the REST approach, one adopts simplicity as the universal principle and delegates most of the difficult problems to application (implementation-specific) software. In a web services language, REST has minimal information in the header, and the message body (that is opaque to generic message processing) carries all the needed information.
- ✓ REST architectures are clearly more appropriate for rapid technology environments. However, the ideas in web services are important and probably will be required in mature systems at a different level in the stack (as part of the application).
- ✓ Note that REST can use XML schemas but not those that are part of SOAP; "XML over HTTP" is a popular design choice in this regard. Above the communication and management layers, we have the ability to compose new entities or distributed programs by integrating several entities together.
- ✓ In CORBA and Java, the distributed entities are linked with RPCs, and the simplest way to build composite applications is to view the entities as objects and use the traditional ways of linking them together. For Java, this could be as simple as writing a Java program with method calls replaced by Remote Method Invocation (RMI), while CORBA supports a similar model with a syntax reflecting the C++ style of its entity (object) interfaces.
- ✓ Allowing the term "grid" to refer to a single service or to represent a collection of services, here sensors represent entities that output data (as messages), and grids and clouds represent collections of services that have multiple message-based inputs and outputs.

c) The Evolution of SOA

- ✓ As shown in Figure 1.21, service-oriented architecture (SOA) has evolved over the years. SOA applies to building grids, clouds, grids of clouds, clouds of grids, clouds of clouds (also known as interclouds), and systems of systems in general.

- ✓ A large number of sensors provide data-collection services, denoted in the figure as SS (sensor service). A sensor can be a ZigBee device, a Bluetooth device, a WiFi access point, a personal computer, a GPA, or a wireless phone, among other things.
- ✓ Raw data is collected by sensor services. All the SS devices interact with large or small computers,many forms of grids, databases, the compute cloud, the storage cloud, the filter cloud, the discovery cloud, and so on. Filter services (fs in the figure) are used to eliminate unwanted raw data, in order to respond to specific requests from the web, the grid, or web services.

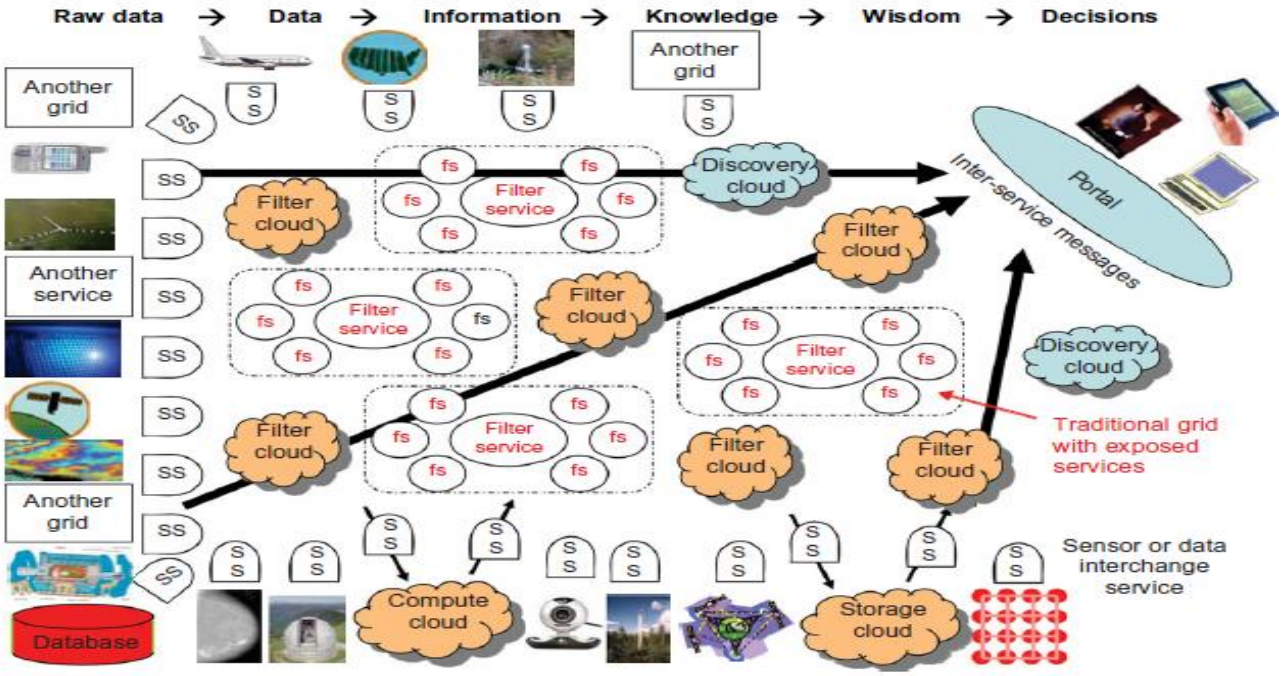


FIGURE 1.21

The evolution of SOA: grids of clouds and grids, where "SS" refers to a sensor service and "fs" to a filter or transforming service.

d) Grids versus Clouds

- ✓ The boundary between grids and clouds are getting blurred in recent years. For web services, workflow technologies are used to coordinate or orchestrate services with certain specifications used to define critical business process models such as two-phase transactions.
- ✓ In general, a grid system applies static resources, while a cloud emphasizes elastic resources. For some researchers, the differences between grids and clouds are limited only in dynamic resource allocation based on virtualization and autonomic computing. One can build a grid out of multiple clouds.
- ✓ This type of grid can do a better job than a pure cloud, because it can explicitly support negotiated resource allocation.
- ✓ Thus one may end up building with a system of systems: such as a cloud of clouds, a grid of clouds, or a cloud of grids, or inter-clouds as a basic SOA architecture.

9. Explain in detail about the Grid Architecture and standards or Overview of Grid Architecture

A new architecture model and technology was developed for the establishment, management, and cross-organizational resource sharing within a virtual organization.

This new architecture, called grid architecture, identifies the basic components of a grid system, defines the purpose and functions of such components and indicates how each of these components interacts with one another (Foster, Kesselman, & Tuecke).

Figure 3.1. This illustrates a layered grid architecture and its relationship to the Internet protocol architecture (Foster, Kesselman, & Tuecke).

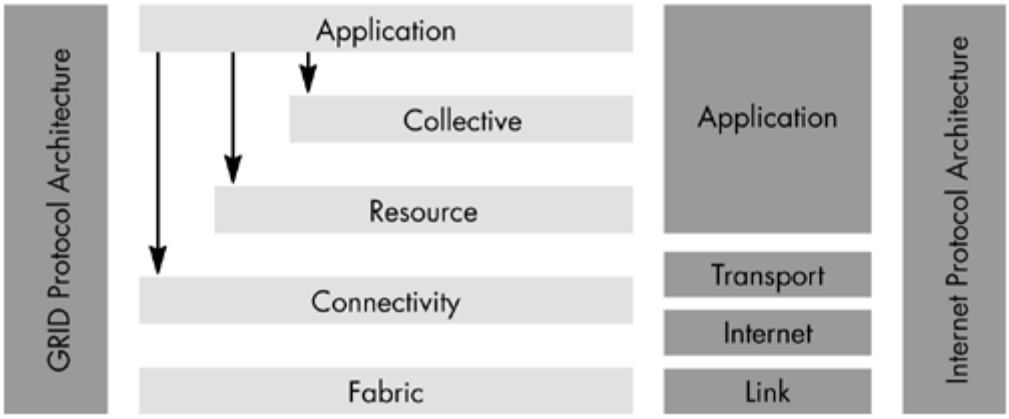


Figure 3.1 illustrates the component layers of the architecture with specific capabilities at each layer. Each layer shares the behavior of the component layers described in the next discussion.

(i)Fabric Layer: Interface to Local Resources

(ii)The Connectivity layer

(iii)Resource Layer: Sharing of a Single Resource

(iv)The Collective Layer: Coordinating Multiple Resources

(v)Application Layer: User-Defined Grid Applications

Now let us explore each of these layers in more detail.

(i)Fabric Layer: Interface to Local Resources

The Fabric layer defines the resources that can be shared. This could include computational resources, data storage, networks, catalogs, and other system resources. These resources can be physical resources or logical resources by nature.

These basic capabilities should be considered as "best practices" toward Grid Computing disciplines. These best practices are as follows:

1. Provide an "inquiry" mechanism whereby it allows for the discovery against its own resource capabilities, structure, and state of operations. These are value-added features for resource discovery and monitoring.
2. Provide appropriate "resource management" capabilities to control the QoS the grid solution promises, or has been contracted to deliver. Connectivity Layer: Manages Communications

(ii)The Connectivity layer

Defines the core communication and authentication protocols required for grid-specific networking services transactions.

- ✓ Communications protocols, which include aspects of networking transport, routing, and naming, assist in the exchange of data between fabric layers of respective resources. The authentication protocol builds on top of the networking communication.
- ✓ The communication protocol can work with any of the networking layer protocols Single sign-on Each resource and hosting has specific security requirements and security solutions that match the local environment.
- ✓ This may include (for example) Kerberos security methods, Windows security methods, Linux security methods, and UNIX security methods.

User-based trust relationships

- ✓ In Grid Computing, establishing an absolute trust relationship between users and multiple service providers is very critical. This accomplishes the environmental factor to which there is then no need of interaction among the providers to access the resources that each of them provide.

Data security

- ✓ The data security topic is important in order to provide data integrity and confidentiality. The data passing through the Grid Computing solution, no matter what complications may exist, should be made secure using various cryptographic and data encryption mechanisms

(iii)Resource Layer: Sharing of a Single Resource

- ✓ The Resource layer utilizes the communication and security protocols defined by the networking communications layer, to control the secure negotiation, initiation, monitoring, metering, accounting, and payment involving the sharing of operations across individual resources. ponsibility of the Collective layer.
- ✓ There are two primary classes of resource layer protocols. These protocols are key to the operations and integrity of any single resource. These protocols are as follows:

Information Protocols

- ✓ These protocols are used to get information about the structure and the operational state of a single resource, including configuration, usage policies, service-level agreements, and the state of the resource. In most situations, this information is used to monitor the resource capabilities and availability constraints.

Management Protocols

The important functionalities provided by the management protocols are:

- Negotiating access to a shared resource is paramount. These negotiations can include the requirements on quality of service, advanced reservation, scheduling, and other key operational factors.
- Performing operation(s) on the resource, such as process creation or data access, is also a very important operational factor.
- Acting as the service/resource policy enforcement point for policy validation between a user and resource is critical to the integrity of the operations.
- Providing accounting and payment management functions on resource sharing is mandatory.
- Monitoring the status of an operation, controlling the operation including terminating the operation, and providing asynchronous notifications on operation status, is extremely critical to the operational state of integrity.

It is recommended that these resource-level protocols should be minimal from a functional overhead point of view and they should focus on the functionality each provides from a utility aspect.

(iv) The Collective Layer: Coordinating Multiple Resources

- ✓ While the Resource layer manages an individual resource, the Collective layer is responsible for all global resource management and interaction with a collection of resources. This layer of protocol implements a wide variety of sharing behaviors (protocols) utilizing a small number of Resource layer and Connectivity layer protocols.

Directory services

- ✓ This enables the virtual organization participants to discover the existence and/or properties of that specific available virtual organization's resources.

Co allocation, Scheduling, and Brokering Services

- ✓ These services allow virtual organization participants to request the allocation of one or more resources for a specific task, during a specific period of time, and to schedule those tasks on the appropriate resources.

Monitoring and Diagnostic Services

- ✓ These services afford the virtual organizations resource failure recovery capabilities, monitoring of the networking and device services, and diagnostic services that include common event logging and intrusion detection.

Data Replication Services

- ✓ These services support the management aspects of the virtual organization's storage resources in order to maximize data access performance with respect to response time, reliability, and costs.

Grid-Enabled Programming Systems

- ✓ These systems allow familiar programming models to be utilized in the Grid Computing environments, while sustaining various Grid Computing networking services.

Workload Management Systems and Collaborative Frameworks

- ✓ This provides multistep, asynchronous, multicomponent workflow management. This is a complex topic across several dimensions, yet a fundamental area of concern for enabling optimal performance and functional integrity.

Software Discovery Services

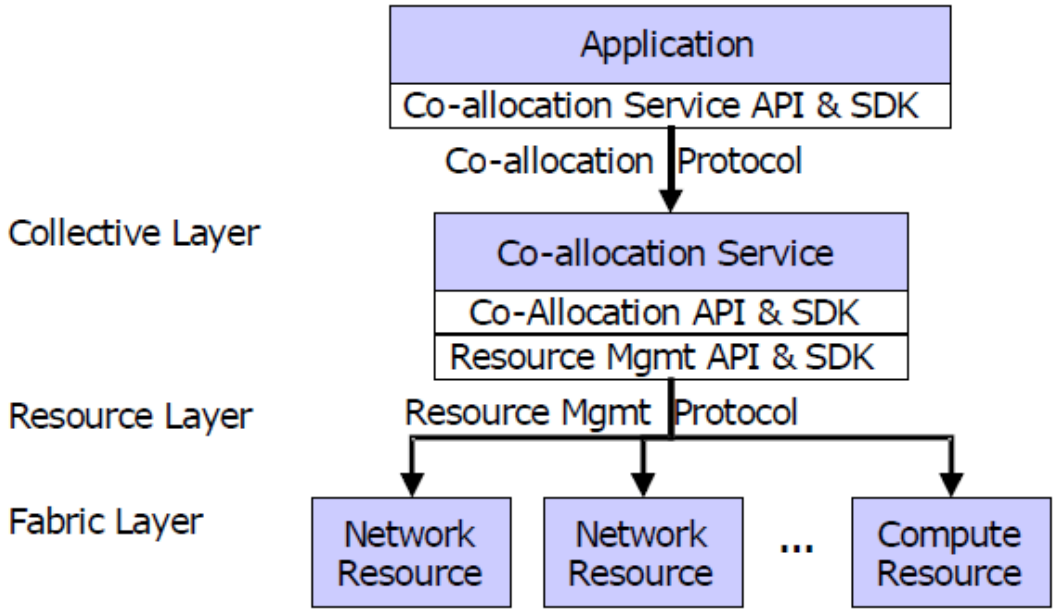
- ✓ This provides the mechanisms to discover and select the best software implementation(s) available in the grid environment, and those available to the platform based on the problem being solved.

Community Authorization Servers

- ✓ These servers control resource access by enforcing community utilization policies and providing these respective access capabilities by acting as policy enforcement agents.

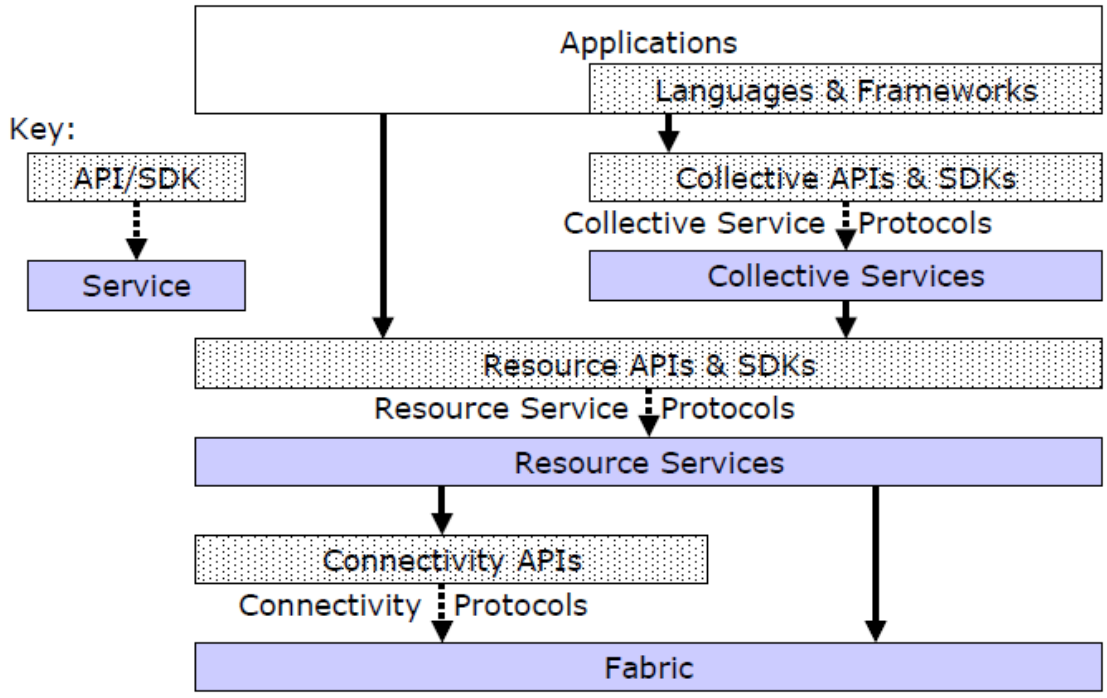
Community Accounting and Payment Services

- ✓ These services provide resource utilization metrics, while at the same time generating payment requirements for members of any community.



(v)Application Layer: User-Defined Grid Applications

- ✓ These are user applications, which are constructed by utilizing the services defined at each lower layer. Such an application can directly access the resource, or can access the resource through the Collective Service interface APIs (Application Provider Interface).



Standards that are involved in areas related to grid computing include:

- ✓ Global Grid Forum (GGF)
- ✓ Organization for the Advancement of Structured Information Standards (OASIS)
- ✓ World Wide Web Consortium (W3C)
- ✓ Distributed Management Task Force (DMTF)
- ✓ Web Services Interoperability Organization (WS-I)

OGSA

- ✓ The Global Grid Forum has published the Open Grid Service Architecture (OGSA). To address the requirements of grid computing in an open and standard way, requires a framework for distributed systems that support integration, virtualization, and management.
- ✓ Such a framework requires a core set of interfaces, expected behaviors, resource models, and bindings.

- ✓ OGSA defines requirements for these core capabilities and thus provides a general reference architecture for grid computing environments. It identifies the components and functions that are useful if not required for a grid environment.
- ✓ Though it does not go to the level of detail such as defining programmatic interfaces or other aspects that would guarantee interoperabilty between implementations, it can be used to identify the functions that should be included based on the requirements of the specific target environment.

OGSI

- ✓ As grid computing has evolved it has become clear that a service-oriented architecture could provide many benefits in the implementation of a grid infrastructure.
- ✓ The Global Grid Forum extended the concepts defined in OGSA to define specific interfaces to various services that would implement the functions defined by OGSA.
- ✓ More specifically, the **Open Grid Services Interface (OGSI)** defines mechanisms for creating, managing, and exchanging information among Grid services.
- ✓ A Grid service is a Web service that conforms to a set of interfaces and behaviors that define how a client interacts with a Grid service.
- ✓ These interfaces and behaviors, along with other OGSI mechanisms associated with Grid service creation and discovery, provide the basis for a robust grid environment.
- ✓ OGSI provides the Web Service Definition Language (WSDL) definitions for these key interfaces. Globus Toolkit 3 included several of its core functions as Grid services conforming to OGSI.

OGSA-DAI

- ✓ The OGSA-DAI (data access and integration) project is concerned with constructing middleware to assist with access and integration of data from separate data sources via the grid.
- ✓ The project was conceived by the UK Database Task Force and is working closely with the Global Grid Forum DAIS-WG and the Globus team.

GridFTP

- ✓ GridFTP is a secure and reliable data transfer protocol providing high performance and optimized for wide-area networks that have high bandwidth.
- ✓ As one might guess from its name, it is based upon the Internet FTP protocol and includes extensions that make it a desirable tool in a grid environment.
- ✓ The GridFTP protocol specification is a proposed recommendation document in the Global Grid Forum (GFD-R-P.020).
- ✓ GridFTP uses basic Grid security on both control (command) and data channels. Features include multiple data channels for parallel transfers, partial file transfers, third-party transfers, and more.
- ✓ GridFTP can be used to move files (especially large files) across a network efficiently and reliably. These files may include the executables required for an application or data to be consumed or returned by an application.
- ✓ Higher level services, such as data replication services, could be built on top of GridFTP.

WSRF

- ✓ **Web Services Resource Framework (WSRF)** WSRF is being promoted and developed through work from a variety of companies, including IBM, and has been submitted to OASIS technical committees.
- ✓ Basically, WSRF defines a set of specifications for defining the relationship between Web services (that are normally stateless) and stateful resources.
- ✓ WSRF is a general term that encompasses several related proposed standards that cover: Resources Resource lifetime Resource properties Service groups (collections of resources) Faults Notifications Topics As the concept of Grid services evolves, the WSRF suite of evolving standards holds great

promise for the merging of Web services standards with the stateful resource management requirements of grid computing.

Web services

- ✓ Web services related standards Because Grid services are so closely related to Web services, the plethora of standards associated with Web services also apply to Grid services.
- ✓ We do not describe all of these standards in this document, but rather recommend that the reader become familiar with standards commonly associate with Web services,
- ✓ such as: XML WSDL SOAP UDDI In addition, there are many evolving standards related to Web Services Interoperabilty (WS-I) that also can be applied to and bring value to grid environments, standards, and proposed standards.

10. Explain in detail about Elements of Grid Computing

- ✓ Grid computing combines elements such as distributed computing, high-performance computing and disposable computing depending on the application of the technology and the scale of operation . Grids can create a virtual supercomputer out of the existing servers, workstations and personal computers.

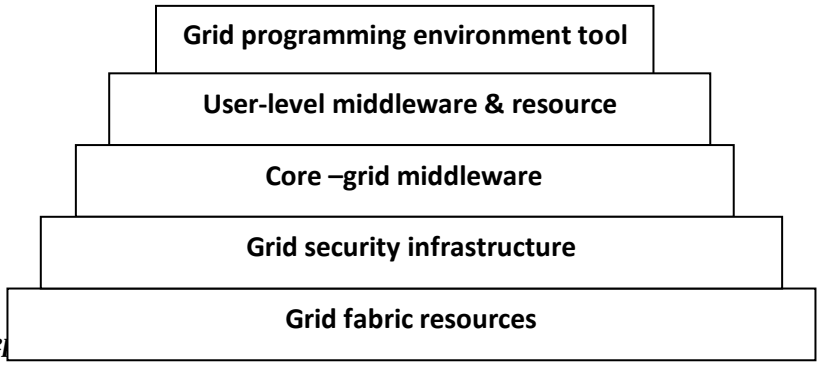
Present-day grids encompass the following types

- ✓ **Computational grids,** in which machines will set aside resources to "number crunch" data or provide coverage for other intensive workloads
- ✓ **Scavenging grids,** commonly used to find and harvest machine cycles from idle servers and desktop computers for use in resource-intensive tasks (scavenging is usually implemented in a way that is unobtrusive to the owner/user of the processor)
- ✓ **Data grids,** which provide a unified interface for all data repositories in an organization, and through which data can be queried, managed and secured.
- ✓ **Market-oriented grid ,** which deal with price setting and negotiation, grid economy management and utility driven scheduling and resource allocation.

specifically, grid computing environment can be viewed as a computing setup constituted by a number of logical hierarchical layers. **Figure 1 represents these layers**.

They include

- ✓ Grid fabric resources,
- ✓ Grid security infrastructure,
- ✓ Core grid middleware,
- ✓ User level middleware and resource aggregators,
- ✓ Grid programming environment and tools and
- ✓ Grid applications.



Overview of grid computing layer

Basic constituent elements—a functional view

- ✓ A resource is an entity that is to be shared; this includes computers, storage, data, and software. A resource does not have to be a physical entity. A resource is defined in terms of interfaces, not devices; for example, schedulers such as Platform's LSF and PBS define a compute resource. Open/close/read/write define access to a distributed file system, for example, NFS, AFS, DFS .

In this section, we look at following grid components:

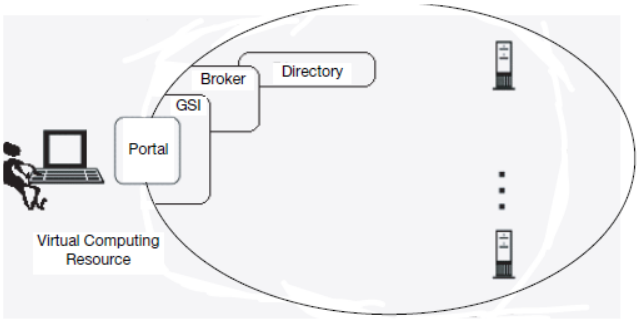
a) **Grid portal**
b) **Security (grid security infrastructure)**
c) **Broker (along with directory)**
d) **Scheduler**
e) **Data management Job and resource management**
f) **Resources**
g)

Figures 3.5 and 3.6 provide a pictorial depiction of the concepts discussed in the subsections that follow. Table 3.5 provides a more detailed listing of functions and capabilities to be supported .

1

2

3



4

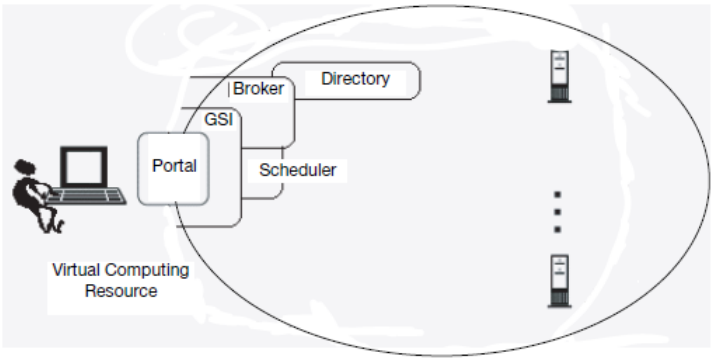


Figure 3.5 Basic grid elements—a user's view.

5

6

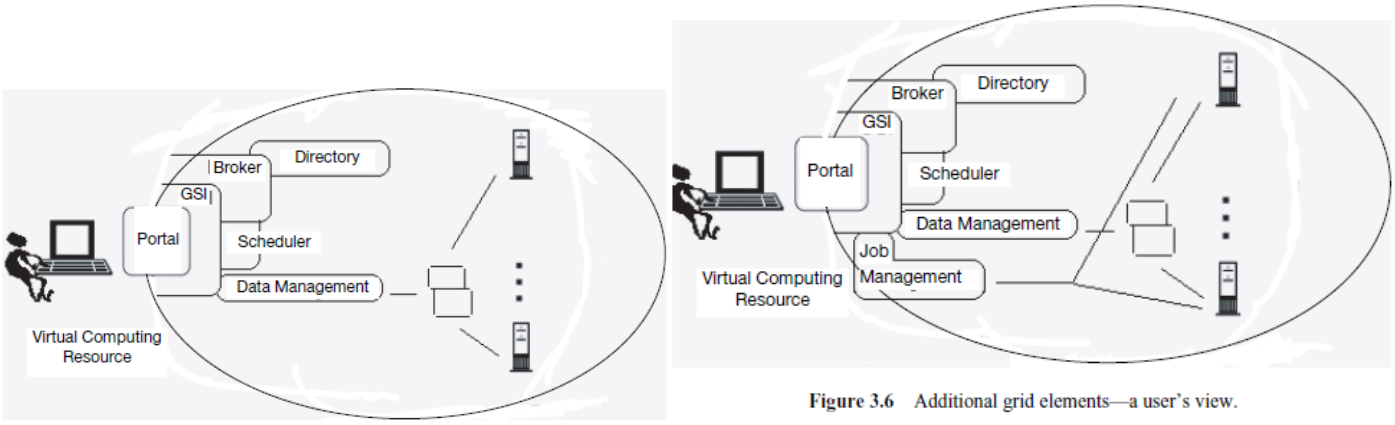


Figure 3.6 Additional grid elements—a user's view.

Table 3.5 Grid functionality to be supported

(Co-)reservation, workflow	Monitoring
Accounting and payment	Performance guarantees
Adaptation	Remote data access
Authorization and policy	Resource allocation
Distributed algorithms	Resource characterization
Fault management	Resource discovery
High-speed data transfer	Resource management
Identity and authentication	System evolution
Intrusion detection	

- ✓ As a supplement, interested readers may also wish to consult the document, "*Anatomy of the Grid*" by Ian Foster, Carl Kesselman, and Steven Tuecke; the paper contains a description of a grid's constituent parts and what they do, with a focus is on grid architecture [3].
- ✓ In the discussion below, we use the term "functional block." This is a generic architectural construct (applicable to any architecture).
- ✓ A functional block is a logical aggregation of functions and capabilities that have an affinity, similarity, close relationship, or related purpose.

a) Portal/User Interface Function/Functional Block

- ✓ A portal/user interface functional block usually exists in the grid environment. The user interaction mechanism (specifically, the interface) can take a number of forms.
- ✓ The interaction mechanism typically is application specific. In the simplest grid environment, the user access may be via a portal (see Figure 3.5, top).
- ✓ Such a portal provides the user with an interface to launch applications. The applications make transparent the use of resources and/or services provided by the grid. With this arrangement, the user perceives the grid as a virtual computing resource.

b)The Grid Security Infrastructure: User Security Function/Functional Block

- ✓ A user security functional block usually exists in the grid environment and, as noted above, a key requirement for grid computing is security.
- ✓ In a grid environment,there is a need for mechanisms to provide authentication, authorization, data confidentiality,data integrity, and availability, particularly from a user's point of view;see Figure 3.5, center.
- ✓ When a user's job executes, typically it requires confidentialmessage-passing services. There may be on-the-fly relationships. But also, the user of the grid infrastructure software (such as a specialized

scheduler) may need to set up a long-lived service; administrators may require that only certain users are allowed to access the service.

- ✓ In each of these cases, the application must anticipate and be designed to provide this required security functionality.
- ✓ The invoker of these applications must have an understanding of how to check if these security services are available and how they can be invoked [72].
- ✓ In grids (particularly intergrids), there is a requirement to support security across organizational boundaries. This makes a centrally managed security system impractical; administrators want to support "single sign-on" for users of the grid, including delegation of credentials for computations that involve multiple resources and/or sites.
- ✓ The grid security infrastructure provides a single-sign-on, run-anywhere authentication service, with support for local control over access rights and mapping from global to local user identities [167].
- ✓ The grid security infrastructure supports uniform authentication, authorization, and message-protection mechanisms in multiinstitutional settings. Specifically, the grid security infrastructure provides, among other services, single sign-on, delegation, and identity mapping using public key technology (X.509 certificates)

Node Security Function/Functional Block

- ✓ A node security functional block usually exists in the grid environment. Authentication and authorization is a "two-way street"; not only does the user need to be authenticated,but also the computing resource.
- ✓ There is the need for secure (authenticated and, in most instances, also confidential) communication between internal elements of a computational grid. This is because a grid is comprised of a collection of hardware and software resources whose origins may not be obvious to a grid user.
- ✓ When a user wants to run on a particular processor, the user needs assurances that the processor has not been compromised, making his or her proprietary application, or data, subject to undesired exposure.
- ✓ If a processor enrolls in a dynamic-rather than preadministered manner, then an identification and authentication validation must be performed before the processor can actually participate in the grid's work, as we discussed earlier.
- ✓ A certificate authority (CA) can be utilized to establish the identity of the "donor" processor, as well as the users and the grid itself. Some grid systems provide their own log-in to the grid, whereas other grid systems depend on the native operating systems for user authentication.

c) Broker Function/Functional Block and Directory

- ✓ A broker functional block usually exists in the grid environment. After the user is authenticated by the user security functional block, the user is allowed to launch an application.
- ✓ At this juncture, the grid system needs to identify appropriate and available resources that can/should be used within the grid, based on the application and application-related parameters provided by the user of the application.
- ✓ This task is carried out by a broker function. The broker functionality provides information about the available resources on the grid and the working status of these resources.
- ✓ Specifically, grid systems have a capability to define (and monitor) a grid's topology in order to share resources and support collaboration; this is typi-cally accomplished via a directory mechanism (e.g., LDAP and/or DNS)

d) Scheduler Function/Functional Block

- ✓ A scheduler functional block usually exists in the grid environment. If a set of stand-alone jobs without any interdependencies needs to execute, then a scheduler is not necessarily required.

- ✓ In the situation where the user wishes to reserve a specific resource or to ensure that different jobs within the application run concurrently,then a scheduler is needed to coordinate the execution of the jobs.
- ✓ In a "trivial" environment, the user may select a processor suitable for running the job and then execute a grid instruction that routes the job to the selected processor.In "nontrivial" environments, a grid-based system is responsible for routing a job to a properly selected processor so that the job can execute.
- ✓ Here, the scheduling software identifies a processor on which to run a specific grid job that has been submitted by a user; see Figure 3.6, top. After available resources have been identified, the follow-on step is to schedule the individual jobs to run on these resources.
- ✓ Schedulers are designed to dynamically react to grid load. They accomplish this by utilizing measurement information relating to the current utilization of processors to determine which ones are available before submitting a job.
- ✓ In an entry-level case, the scheduler could assign jobs in a round-robin fashion to the "next" processor matching the resource requirements. More commonly, the scheduler automatically finds the most appropriate processor on which to run a given job.
- ✓ Some schedulers implement a job priority (queue) mechanism. In this environment,as grid processors become available to execute jobs, the jobs are selected from the highest-priority queues first. Policies of various kinds can be implemented via the scheduler; for example, there could be a policy that restricts grid jobs from executing at certain time windows.
- ✓ In some more complex environments, there could be different levels of schedulers organized in a hierarchy. For example, a metascheduler may submit a job to a cluster scheduler or other lower-level scheduler rather than to an individual target processor.
- ✓ As another example, a cluster could be represented as a single resource; here, the cluster could have its own scheduler to manage the internal cluster nodes, while a higher-level scheduler could be employed to schedule work to be supported by the cluster in question as an ensemble.
- ✓ Advanced schedulers monitor the progress of active jobs, managing the overall workflow. If the jobs were to become lost due to system or network outages, a highend scheduler would automatically resubmit the job elsewhere.
- ✓ However, if a job appears to be in an infinite loop and reaches a maximum timeout, then such jobs will not be rescheduled. Typically, jobs have different kinds of completion codes.
- ✓ This code determines if the job is suitable for resubmission or not [147]).Some grids also have a "reservation system." These systems allow one to reserve resources on the grid. These are a calendar-based mechanism for reserving resources for specific time periods, and preventing others from reserving the same resource at the same time.
- ✓ In a "scavenging" grid environment, any processor that becomes idle reports its idle status to the grid management node. The management node in turn assigns to this processor the next job that is satisfied by the processor's resources.
- ✓ If the processor becomes busy with local nongrid work, the grid job is usually suspended or delayed. This situation creates somewhat unpredictable completion times for grid jobs, although it is not disruptive to the processors donating resources to the grid.
- ✓ To create more predictable behavior, grid processors are often "dedicated" to the grid and are not preempted by external work; this enables schedulers to compute the approximate completion time for a set of jobs when their running characteristics are known.

e) Data Management Function/Functional Block

- ✓ A data management functional block usually exists in a grid environment. There typically needs to be a reliable (and secure) method for moving files and data to various nodes within the grid.
- ✓ This functionality is supported by the data management functional block. Figure 3.6, middle, depicts a data management function needed to support this data management function.

f) Resources

- ✓ A grid would be of no value if it did not contribute resources to the ultimate user and/or application. As noted, resources include processors, data storage, scientific equipment, etc. Besides "physical presence" on the grid (by way of an interconnecting network), there has to be "logical presence." "Logical presence" is achieved by installing grid-support software on the participating processors.
- ✓ After loading and activating the software that manages the grid's use of its affiliated resources, each processor contributing itself or contributing ancillary resources to the grid needs to properly enroll as a member of the grid.
- ✓ As discussed in the previous subsections, a user accessing the grid to accomplish a task submits a job for execution on the grid. The grid management software communicates with the grid donor software of the resource(s) logically present to forward the job to an appropriate processor.
- ✓ The grid-support software on the processor accepts an executable job from the grid management system and executes it.The grid software on the "donor" processor must be able to receive the executablefile (in some cases the executable copies preinstalled on the processor.)
- ✓ The software is run and the output is sent back to the requester. More advanced implementations can dynamically adjust the priority of a running job, suspend a job and resume it later, or checkpoint a job with the possibility of resuming its execution on adifferent processor.
- ✓ The grid system sends information about any available resources on that processor to the resource management functional block described in the previous subsection. The participating donor processor typically has a selfmonitoring capability that determines or measures how busy the processor is.
- ✓ This information is "distributed" to the management software of the grid and it is utilized to schedule the appropriate use of the resources. For example, in a scavenging system, this utilization information informs the grid management software when the processor is idle and available to accept work.

g) Job Management and Resource Management Function/Functional Block

- ✓ A job management and resource management functional block usually exists in a grid environment. This functionality is also known as the grid resource allocation manager (GRAM). The job management and resource management function (see Figure 3.6, bottom) provides the services to actually launch a job on a particular resource,to check the job's status, and to retrieve the results when the job is complete.
- ✓ Typically, the management component keeps track of the resources available to the grid and which users are members of the grid. This information is used by the scheduler to decide where grid jobs should be assigned.
- ✓ Also, typically, there are measurement mechanisms that determine both the capacities of the nodes on the grid and their current utilization levels at any given point in time; this information is used to schedule jobs in the grid, to monitor the health of the grid (e.g., outages, congestion, overbooking/over commitment), and to support administrative tasks (e.g., determine overall usage patterns and statistics, log and account for usage of grid resources, etc.)
- ✓ Furthermore, advanced grid management software can automatically manage recovery from a number of grid failures and/or outages (e.g., specifically identify alternatives processors or setups to get the workload processed) .
- ✓ With grid computing, administrators can "virtualize," or pool, IT resources (computers, storage, and applications) into a single virtual system whose resources can be managed from a single administration console and can be allocated dynamically, based on demand.
- ✓ The job management and resource management functional block supports this simplified view of the enterprise-wide resources.
- ✓ The work involved in managing the grid may be distributed hierarchically, in order to increase the scalability of the grid. For example, a central job scheduler maynot schedule a submitted job directly,

but instead, the job request is sent to a secondary scheduler that handles a specified set of processors (e.g., a cluster); the secondary scheduler handles the assignment to the specific processor.

- ✓ Hence, in this instance, the grid operation, the resource data, and the job scheduling are distributed to match the topology of the grid. On the resource management side of this function, mechanisms usually exist to handle observation, management, measurement, and correlation.
- ✓ It was noted above that schedulers need to react to instantaneous loads on the grid. The donor software typically includes "load sensors" that measure the instantaneous load and activity on resources or processors. Such measurement information is useful not only for instantaneous scheduling of tasks and work, but also for assessing (administratively)overall grid usage patterns.
- ✓ Observation, management, and measurement data can be used, in aggregate, to support capacity planning and initiate deployment of additional hardware.
- ✓ Furthermore, measurement information about specific jobs can be collected and used to forecast the resource requirements of that job the next time it executes. Some grid systems provide the means for implementing custom load sensors for more than just processor or storage resources.

h) User/Application Submission Function/Functional Block

- ✓ A user/application submission functional block usually exists. Typically, any member of a grid can submit jobs to the grid and perform grid queries, but in some grid systems, this function is implemented as a separate component installed on "submission nodes or clients".

i) Protocols

- ✓ After identifying the functional blocks, a generic architecture description proceeds by defining the protocols to be employed between (specifically, on the active interfaces of the) functional blocks.
- ✓ To interconnect these functional blocks, we need protocols, especially standardized protocols. Protocols are formal descriptions of message formats and a set of rules for message exchange. The rules may define sequence of message exchanges. Protocols are generally layered. Figure 3.7 depicts two examples of protocol stacks and network-enabled services.
- ✓ The grid dénouement (call it vision) requires protocols that are not only open and general purpose but also are vendor-independent and widely adopted *standards.*

Standards allows the grid to establish resource-sharing arrangements dynamically with *any* interested party and thus to create something more than a plethora of balkanized, incompatible, noninteroperable distributed systems; standards are also important as a means of enabling general-purpose services and tools.

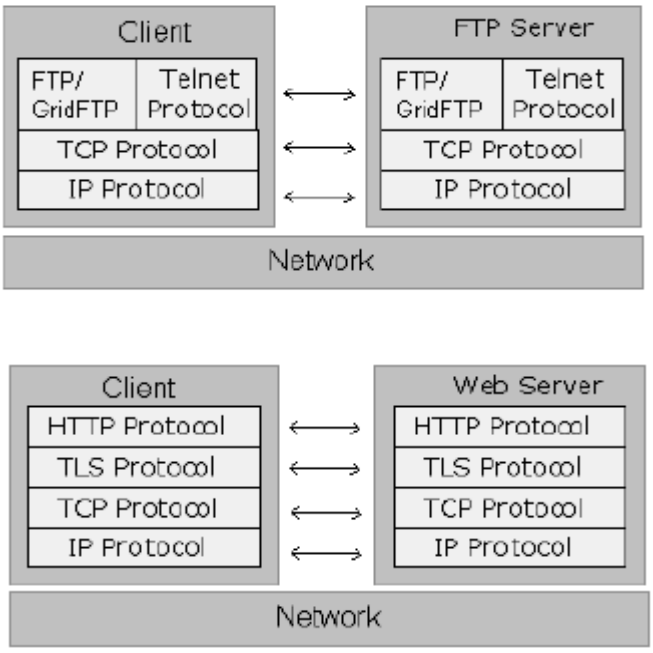


Figure 3.7 Example of protocol stacks and network-enabled services.

- ✓ Both open source and commercial products can, then, interoperate effectively in this heterogeneous,multivendor grid world, thus providing the pervasive infrastructure that will enable successful grid applications.
- ✓ At this juncture, the Global Grid Forum is in the process of developing consensus standards for grid environments. On the commercial side, nearly a decade of experience and refinement have resulted in a widely used de-facto standard in the form of the open source Globus Toolkit.
- ✓ The Global Grid Forum has a major effort underway to define the Open Grid Services Architecture (OGSA), which modernizes and extends Globus Toolkit protocols to address new requirements, while also embracing Web services.

IMPORTANT QUESTIONS
PART-A

1. Define Evolution of Distributed computing
2. What is Scalable computing over the Internet
3. Give the Technologies for network based systems
4. Define clusters of cooperative computers
5. What is Grid computing Infrastructures
6. Define cloud computing
7. Define service oriented architecture
8. Define standards in grid
9. List the Elements of Grid.
10. Write the short notes about utility computing
11. Write the short notes about parallel computing
12. What is cluster?
13. What is grid computing? (Apr 11)(Nov 10)
14. What are the business benefits in Grid Computing?
15. What are the areas are difficult to implement in Grid Computing Infrastructure?

16. Explain the key components of grid computing

PART-B

1. Explain in detail about the Evolution of Distributed computing: Scalable computing over the Internet .
2. List out Technologies for network based systems with brief explanation.
3. What is clusters of cooperative computers and give its components.
4. Explain what is Grid computing Infrastructures with neat description.
5. What are cloud computing and give short notes.
6. Explain in detail about the service oriented architecture with neat sketch.
7. Give the Introduction to Grid Architecture and list out the standards in grid.
8. Briefly explain about the Elements of Grid .
9. Give the Overview of Grid Architecture.

UNIT IV
PROGRAMMING MODEL

Open source grid middleware packages – Globus Toolkit (GT4) Architecture , Configuration – Usageof Globus – Main components and Programming model - Introduction to Hadoop Framework -Mapreduce, Input splitting, map and reduce functions, specifying input and output parameters,configuring and running a job – Design of Hadoop file system, HDFS concepts, command line and java interface, dataflow of File read & File write.

PART-A

1. **Define Globus Toolkit: Grid Computing Middleware**
   - ✓ Globus is open source grid software that addresses the most challenging problmes in distributed resources sharing.
   - ✓ The Globus Toolkit includes software services and libraries for distributed security, resource management, monitoring and discovery, and data management.

2. **What Is the Globus Toolkit?**
   - ✓ The Globus Toolkit is a *collection of solutions* to problems that frequently come up when trying to build collaborative distributed applications.
   - ✓ *Not* turnkey solutions, but *building blocks* and *tools* for application developers and system integrators.
   - ✓ Some components (e.g., file transfer) go farther than others (e.g., remote job submission) toward end-user relevance.
   - ✓ To date (v1.0 - v4.0), the Toolkit has focused on *simplifying heterogeneity* for application developers.
   - ✓ The goal has been to *capitalize on and encourage use of existing standards* (IETF, W3C, OASIS, GGF).
   - ✓ The Toolkit also includes reference implementations of new/proposed standards in these organizations.

3. **Give Three main components of Globus .**
   Three main components of Globus are:
   1. Security
   2. Information Infrastructure
   3. Resource Management

4.List the resources of The Grid Resource Information Service (GRIS)

- ✓ The Grid Resource Information Service (GRIS) provides a uniform means of querying resources on a computational grid for their current status. Such resources include:

  a) computational nodes
  b) data storage systems
  c) scientific instruments
  d) network links
  e) databases

5. Give An overview of the software components (GridFTP, Replica Catalog, Replica Management)

GridFTP: A high-performance, secure, robust data transfer mechanism

Globus Replica Catalog: A mechanism for maintaining a catalog of dataset replicas.

Globus Replica Management: A mechanism that ties together the Replica Catalog and GridFTP technologies, allowing applications to create and manage replicas of large datasets.

6. What is a Gatekeeper ?

- ✓ Gatekeeper is the process which exists on the remote computer before any request is submitted.It runs as a root and handles the job requests.
- ✓ When the gatekeeper receives an allocation request from a client, it mutually authenticates with the client,maps the requestor to a local user, starts a job manager on the local host as the local user,and passes the allocation arguments to the newly created job manager.

7. What is a Job Manager ?

- ✓ A single job manager is created by the gatekeeper to fulfill every request submitted to the gatekeeper. It starts the job on the local system, and handles all further communication with the client.

It is made up of two components:

a) Common Component - translates messages received from the gatekeeper and client into an internal API that is implemented by the machine specific component. It also translates callback requests from the machine specific components through the internal API into messages to the application manager.

b) Machine-Specific Component - implements the internal API in the local environment. This includes calls to the local system, messages to the resource monitor, and inquiries to the MDS.

8.What is Globus Resource Allocation Manager (GRAM)

- ✓ The Globus Resource Allocation Manager (GRAM) processes the requests for resources for remote application execution, allocates the required resources, and manages the active jobs. It also returns updated information regarding the capabilities and availability of the computing resources to the Metacomputing Directory Service (MDS).
- ✓ GRAM provides an API for submitting and canceling a job request, as well as checking the status of a submitted job.The specifications are written by the user in the Resource Specification Language (RSL), and is processed by GRAM as part of the job request.
- ✓ The Globus Resource Allocation Manager (GRAM) is the lowest level of Globus resource management architecture.

GRAM allows you to run jobs remotely, providing an API for submitting, monitoring, and terminating your job. When a job is submitted,the request is sent to the gatekeeper of the remote computer. The gatekeeper handles the request and creates a job manager for the job.

9. Give Input Splitting concept in Hadoop Framework

- ✓ For the framework to be able to distribute pieces of the job to multiple machines, it needs to fragment the input into individual pieces, which can in turn be provided as input to the individual distributed tasks.
- ✓ Each fragment of input is called an *input split*. The default rules for how input splits are constructed from the actual input files are a combination of configuration parameters and the capabilities of the class that actually reads the input records.
- ✓ An input split will normally be a contiguous group of records from a single input file, and in this case, there will be at least *N* input splits, where *N* is the number of input files. If the number of requested map tasks is larger than this number, or the individual files are larger than the suggested fragment size, there may be multiple input splits constructed of each input file.

10. Give the Specifying Input Formats

- ✓ The Hadoop framework provides a large variety of input formats. The major distinctions are between textual input formats and binary input formats.

- ✓ The following are the available formats:
  - **KeyValueTextInputFormat**: Key/value pairs, one per line.
  - **TextInputFormant:** The key is the line number, and the value is the line.
  - **NLineInputFormat:** Similar to KeyValueTextInputFormat, but the splits are based on *N* lines of input rather than *Y* bytes of input.
  - **MultiFileInputFormat:** An abstract class that lets the user implement an input format that aggregates multiple files into one split.
  - **SequenceFIleInputFormat:** The input file is a Hadoop sequence file, containing serialized key/value pairs.
- ✓ KeyValueTextInputFormat and SequenceFileInputFormat are the most commonly used input formats. The examples in this chapter use KeyValueTextInputFormat, as the input files are human-readable.

11. To configure the reduce phase, the user must supply the framework with five pieces of information what are they.

i. The number of reduce tasks; if zero, no reduce phase is run
ii. The class supplying the reduce method
iii. The input key and value types for the reduce task; by default, the same as the reduce output
iv. The output key and value types for the reduce task
v. The output file type for the reduce task output

12. Give the HDFS Concepts

i. Blocks
ii. Namenodes and Datanodes
iii. HDFS Federation
iv. HDFS High-Availability

13. Define Blocks in HDFS

- ✓ A disk has a block size, which is the minimum amount of data that it can read or write. Filesystems for a single disk build on this by dealing with data in blocks, which are an integral multiple of the disk block size. Filesystem blocks are typically a few kilobytes in size, while disk blocks are normally 512 bytes. This is generally transparent to the filesystem user who is simply reading or writing a file—of whatever length.
- ✓ However, there are tools to perform filesystem maintenance, such as *df* and *fsck*, that operate on the filesystem block level. HDFS, too, has the concept of a block, but it is a much larger unit—64 MB by default.

14. Define Namenodes and Datanodes

- ✓ An HDFS cluster has two types of node operating in a master-worker pattern:
  - ➢ a ***namenode*** (the master) and
  - ➢ a number of ***datanodes*** (workers).
- ✓ The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. This information is stored persistently on the local disk in the form of two files: the namespace image and the edit log.
- ✓ The namenode also knows the datanodes on which all the blocks for a given file are located, however, it does not store block locations persistently, since this information is reconstructed from datanodes when the system starts.
- ✓ A ***client*** **accesses** the filesystem on behalf of the user by communicating with the namenode and datanodes. The client presents a POSIX-like filesystem interface, so the user code does not need to know about the namenode and datanode to function.

- ✓ Datanodes are the workhorses of the filesystem. They store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing.
- ✓ Without the namenode, the filesystem cannot be used. In fact, if the machine running the namenode were obliterated, all the files on the filesystem would be lost since there would be no way of knowing how to reconstruct the files from the blocks on the datanodes.

15. Define HDFS Federation

- ✓ The namenode keeps a reference to every file and block in the filesystem in memory, which means that on very large clusters with many files, memory becomes the limiting factor for scaling HDFS Federation, introduced in the 0.23 release series, allows a cluster to scale by adding namenodes, each of which manages a portion of the filesystem namespace.
- ✓ For example, **one namenode** might manage all the files rooted under */user*, say, and a **second namenode** might handle files under */share*. Under federation, each namenode manages a *namespace volume*, which is made up of the metadata for the namespace, and a *block pool* containing all the blocks for the files in the namespace.
- ✓ Namespace volumes are independent of each other, which means namenodes do not communicate with one another, and furthermore the failure of one namenode does not affect the availability of the namespaces managed by other namenodes.

16. Define Directories in file system.

FileSystem provides a method to create a directory:

public boolean mkdirs(Path f) throws IOException

- ✓ This method creates all of the necessary parent directories if they don't already exist, just like the java.io.File's mkdirs() method. It returns true if the directory (and all parent directories) was (were) successfully created.

17 . Define File patterns

- ✓ It is a common requirement to process sets of files in a single operation. For example, a MapReduce job for log processing might analyze a month's worth of files contained in a number of directories. Rather than having to enumerate each file and directory to specify the input, it is convenient to use wildcard characters to match multiple files with a single expression, an operation that is known as *globbing*.
- ✓ Hadoop provides two FileSystem method for processing globs:

public FileStatus[] globStatus(Path pathPattern) throws IOException
public FileStatus[] globStatus(Path pathPattern, PathFilter filter) throws IOException

- ✓ The globStatus() method returns an array of FileStatus objects whose paths match the supplied pattern, sorted by path. An optional PathFilter can be specified to restrict the matches further.

18. List out the GT Components are grouped into five main functional areas:

- **Common runtime** components provide low-level infrastructure for communication and threading, as well as libraries and tools for service and state management. The C, Java, and Python subsystems provide support for services written in C, Java, and Python, respectively. This tutorial is designed to teach you how to build a service using GT's Java runtime; C and Python runtime components are outside the scope of this tutorial.
- **Data services** address the requirements for locating, replicating, transferring and accessing data stored in either files or databases. A specific focus of components in this area is the management of very large scale datasets both in terms of number of data objects (i.e. files) and the size of those individual data objects. The data components are outside the scope of this tutorial.

- **Security** components address cross-cutting issues of authentication and authorization of both users and services. With GT4, the security implementation has moved from primarily a collection of libraries to a more service-oriented structure, as illustrated by the introduction of standalone delegation and authorization services. This tutorial includes an introduction to authentication and authorization.
- **Execution** components provide for the deployment, execution and management of applications. Previous versions of the toolkit focused primarily on the execution of user-provided programs. However, newer services have generalized these functions to address deployment and management of community services as well as providing enhanced capability with respect to configuring and managing the execution environment in which applications and service run. Execution components are outside the scope of this tutorial.
- **Information** components are concerned with the monitoring and discovery of both GT and non-GT services and capabilities. Starting with GT3 and continuing in GT4, many of the basic functions of state publication and management have been incorporated into the common runtime component of the system. The information components build on this by providing an extensible infrastructure for aggregating service state information, providing interfaces for publishing information from non-GT services, and notifying users when events of interest occur. The GT4 Index service and WebMDS are used in this tutorial.

19.Give Globus Toolkit 4 provides components in the following five categories:

1. Common runtime components
2. Security
3. Data management
4. Information services
5. Execution management

20. Define Security components

- ✓ Because security is one of the most important issues in grid environments, Globus Toolkit 4 includes various types of security components.
  - ✓ WS authentication and authorization
  - ✓ Pre-WS authentication and authorization
  - ✓ Community Authorization Service (CAS)
  - ✓ Delegation service
  - ✓ SimpleCA
  - ✓ MyProxy
  - ✓ GSI-OpenSSH

21.Define Community Authorization Service (CAS)

- ✓ CAS provides access control to virtual organizations. The CAS server grants fine-grained permissions on subsets of resources to members of the community. CAS authorization is currently not available for Web services, but it supports the GridFTP server.

22.Define Delegation service

- ✓ The Delegation service enables delegation of credentials between various services in one host. The Delegation service allows a single delegated credential to be used by many services. Also, this service has a credential renewal interface, and this service is capable of extending the valid date of credentials.

23.Define Data management components

Globus Toolkit 4 provides various tools that enable data management in a grid environment.

- ✓ GridFTP
- ✓ Reliable File Transfer (RFT)
- ✓ Replica Location Service (RLS)

- ✓ OGSA-DAI
- ✓ Data Replication Service (DRS)

24. Define Reliable File Transfer (RFT)

- ✓ Reliable File Transfer provides a Web service interface for transfer and deletion of files. RFT receives requests via SOAP messages over HTTP and utilizes GridFTP. RFT also uses a database to store the list of file transfers and their states, and is capable of recovering a transfer request that was interrupted. Figure 10-3 shows how RFT and GridFTP work.

The following are some of the key features of an Index service:

- ➢ Index services can be configured in hierarchies, but there is no single global index that provides information about every resource on the Grid.
- ➢ The presence of a resource in an Index service makes no guarantee about the availability of the resource for users of that Index.
- ➢ Information published with MDS is recent but not the absolute latest.
- ➢ Each registration into an Index service has a lifetime and requires periodic renewal of registrations to indicate the continued existence of a resource or a service.

25. What is Execution management

- ✓ Globus Toolkit 4 provides various tools that enable execution management in a grid environment.
  - ➢ WS GRAM
  - ➢ Community Scheduler Framework 4 (CSF4)
  - ➢ Globus Teleoperations Control Protocol (GTCP)
  - ➢ Workspace Management Service (WMS)

PART-B

1. Give detail explanation about the Open Source Grid Middleware Packages

We first introduce some grid standards and popular APIs. Then we present the desired software support and middleware developed for grid computing. Table 7.6 summarizes four grid middleware packages.

(i) Grid Standards and APIs Grid standards have been developed over the years.

- ✓ The Open Grid Forum (formally Global Grid Forum) and Object Management Group are two well-formed organizations behind those standards. We have already introduced the OGSA (Open Grid Services Architecture)
- ✓ The grid standards have guided the development of several middleware libraries and API tools for grid computing. They are applied in both research grids and production grids today. Research grids tested include the EGEE, France Grilles, D-Grid (German), CNGrid (China),TeraGrid (USA), etc.
- ✓ Production grids built with the standards include the EGEE, INFN grid (Italian), NorduGrid, Sun Grid, Techila, and Xgrid . We review next the software environments and middleware implementations based on these standards.

(ii) Software Support and Middleware

- ✓ Grid middleware is specifically designed a layer between hardware and the software. The middleware products enable the sharing of heterogeneous resources and managing virtual organizations created around the grid. Middleware glues the allocated resources with specific user applications.
- ✓ Popular grid middleware tools include the Globus Toolkits (USA), gLight, UNICORE (German), BOINC (Berkeley), CGSP (China), Condor-G, and Sun Grid Engine, etc. Table 7.6 summarizes the grid software support and middleware packages developed for grid systems since 1995.

Table 7.6 Grid Software Support and Middleware Packages

Package	Brief Description, References and Coverage in This Book
BOINC	Berkeley Open Infrastructure for Network Computing.
UNICORE	Middleware developed by the German grid computing community.
Globus (GT4)	A middleware library jointly developed by Argonne National Lab., Univ. of Chicago, and USC Information Science Institute, funded by DARPA, NSF, and NIH. (Sections 7.4.2, 7.4.3, and 7.5.5).
CGSP in ChinaGrid	The CGSP (ChinaGrid Support Platform) is a middleware library developed by 20 top universities in China as part of the ChinaGrid Project (Section 7.4.4).
Condor-G	Originally developed at the Univ. of Wisconsin for general distributed computing, and later extended to Condor-G for grid job management. (Example 7.9).
Sun Grid Engine (SGE)	Developed by Sun Microsystems for business grid applications. Applied to private grids and local clusters within enterprises or campuses. (Example 7.10).

2. Explain in detail about The Globus Toolkit Architecture (GT4)

- ✓ The Globus Toolkit, started in 1995 with funding from DARPA, is an open middleware library for the grid computing communities. These open source software libraries support many operational grids and their applications on an international basis.
- ✓ The toolkit addresses common problems and issues related to grid resource discovery, management, communication, security, fault detection, and portability.
- ✓ The software itself provides a variety of components and capabilities. The library includes a rich set of service implementations.
- ✓ The implemented software supports grid infrastructure management, provides tools for building new web services in Java, C, and Python, builds a powerful standard-based security infrastructure and client APIs (in different languages), and offers comprehensive command-line programs for accessing various grid services.
- ✓ The Globus Toolkit was initially motivated by a desire to remove obstacles that prevent seamless collaboration, and thus sharing of resources and services, in scientific and engineering applications. The shared resources can be computers, storage, data, services, networks, science instruments (e.g., sensors), and so on. The Globus library version GT4, is conceptually shown in Figure 7.18.

(i) The GT4 Library

- ✓ GT4 offers the middle-level core services in grid applications. The high-level services and tools, such as MPI, Condor-G, and Nirod/G, are developed by third parties for general-purpose distributed computing applications.
- ✓ The local services, such as LSF, TCP, Linux, and Condor, are at the bottom level and are fundamental tools supplied by other developers.
- ✓ Table 7.7 summarizes GT4's core grid services by module name. Essentially, these functional modules help users to discover available resources, move data between sites, manage user credentials, and so on.
- ✓ As a de facto standard in grid middleware, GT4 is based on industry-standard web service technologies.
- ✓ Nexus is used for collective communications and HBM for heartbeat monitoring of resource nodes. GridFTP is for speeding up internode file transfers. The module GASS is used for global access of secondary storage.

Table 7.7 Functional Modules in Globus GT4 Library

Service Functionality	Module Name	Functional Description
Global Resource Allocation Manager	GRAM	Grid Resource Access and Management (HTTP-based)
Communication	Nexus	Unicast and multicast communication
Grid Security Infrastructure	GSI	Authentication and related security services
Monitory and Discovery Service	MDS	Distributed access to structure and state information
Health and Status	HBM	Heartbeat monitoring of system components
Global Access of Secondary Storage	GASS	Grid access of data in remote secondary storage
Grid File Transfer	GridFTP	Inter-node fast file transfer

Courtesy of Ian Foster [17].

(ii) Globus Job Workflow

- ✓ Figure 7.19 shows the typical job workflow when using the Globus tools. A typical job execution sequence proceeds as follows: The user delegates his credentials to a delegation service.
- ✓ The user submits a job request to GRAM with the delegation identifier as a parameter. GRAM parses the request, retrieves the user proxy certificate from the delegation service, and then acts on behalf of the user.
- ✓ GRAM sends a transfer request to the RFT (Reliable File Transfer), which applies GridFTP to bring in the necessary files. GRAM invokes a local scheduler via a GRAM adapter and the SEG (Scheduler Event Generator) initiates a set of user jobs.
- ✓ The local scheduler reports the job state to the SEG. Once the job is complete, GRAM uses RFT and GridFTP to stage out the resultant files. The grid monitors the progress of these operations and sends the user a notification when they succeed, fail, or are delayed.

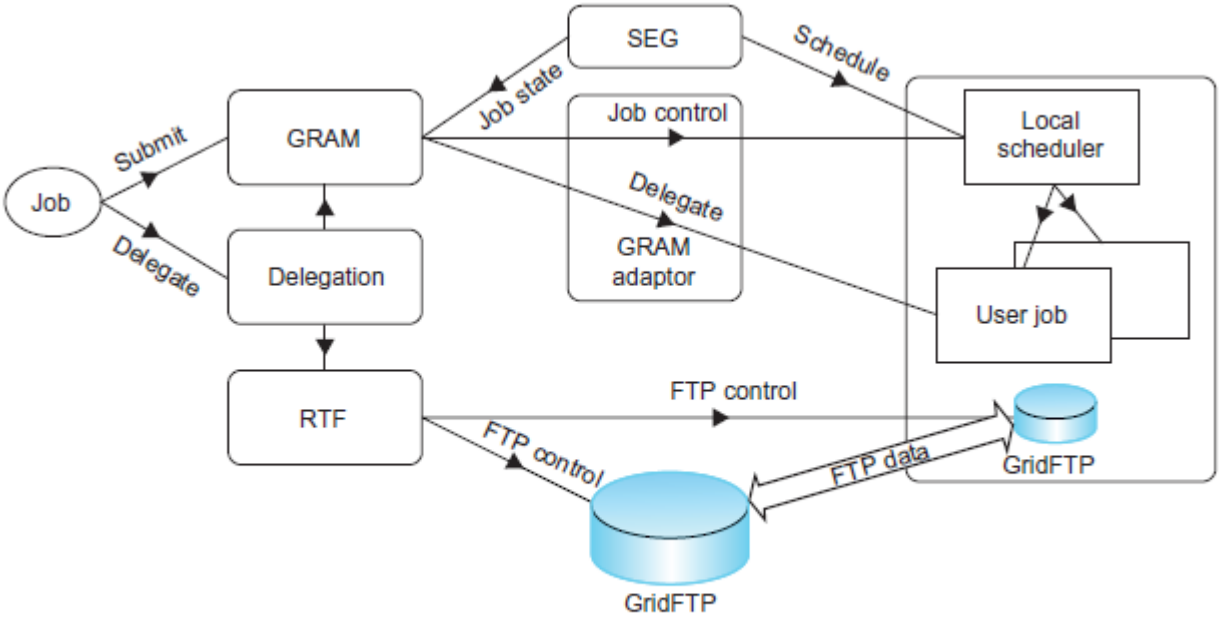


FIGURE 7.19

Globus job workflow among interactive functional modules.

(iii) Client-Globus Interactions

- ✓ GT4 service programs are designed to support user applications as illustrated in Figure 7.20. There are strong interactions between provider programs and user code.
- ✓ GT4 makes heavy use of industry-standard web service protocols and mechanisms in service description, discovery, access, authentication, authorization, and the like.
- ✓ GT4 makes extensive use of Java, C, and Python to write user code. Web service mechanisms define specific interfaces for grid computing. Web services provide flexible, extensible, and widely adopted XML-based interfaces.

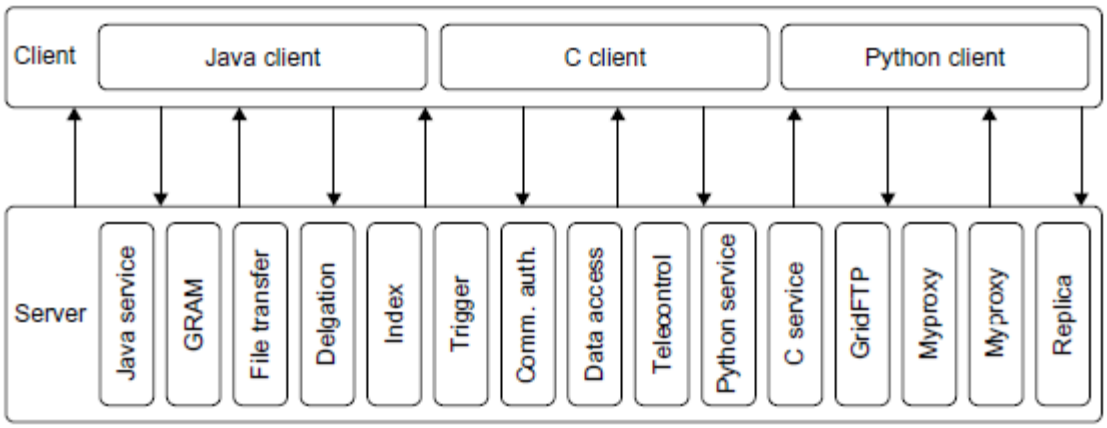


FIGURE 7.20

Client and GT4 server interactions; vertical boxes correspond to service programs and horizontal boxes represent the user codes.

- ✓ GT4 components do not, in general, address end-user needs directly. Instead, GT4 provides a set of infrastructure services for accessing, monitoring, managing, and controlling access to infrastructure elements.
- ✓ The server code in the vertical boxes in Figure 7.22 corresponds to 15 grid services that are in heavy use in the GT4 library. These demand computational, communication, data, and storage resources. We must enable a range of end-user tools that provide the higher-level capabilities needed in specific user applications.
- ✓ Wherever possible, GT4 implements standards to facilitate construction of operable and reusable user code. Developers can use these services and libraries to build simple and complex systems quickly.
- ✓ A high-security subsystem addresses message protection, authentication, delegation, and authorization. Comprising both a set of service implementations (server programs at the bottom of Figure 7.21) and associated client libraries at the top, GT4 provides both web services and non-WS applications. The horizontal boxes in the client domain denote custom applications and or third-party tools that access GT4 services.
- ✓ The toolkit programs provide a set of useful infrastructure services.
  Three containers are used to host user-developed services written in Java, Python, and C, respectively.
- ✓ These containers provide implementations of security, management, discovery, state management, and other mechanisms frequently required when building services.
- ✓ They extend open source service hosting environments with support for a range of useful web service specifications, including WSRF, WS-Notification, and WS-Security.
- ✓ A set of client libraries allow client programs in Java, C, and Python to invoke operations on both GT4 and user-developed services. In many cases, multiple interfaces provide different levels of control:

- ✓ For example, in the case of GridFTP, there is not only a simple command-line client (globusurl- copy) but also control and data channel libraries for use in programs—and the XIO library allowing for the integration of alternative transports.
- ✓ The use of uniform abstractions and mechanisms means clients can interact with different services in similar ways, which facilitates construction of complex, interoperable systems and encourages code

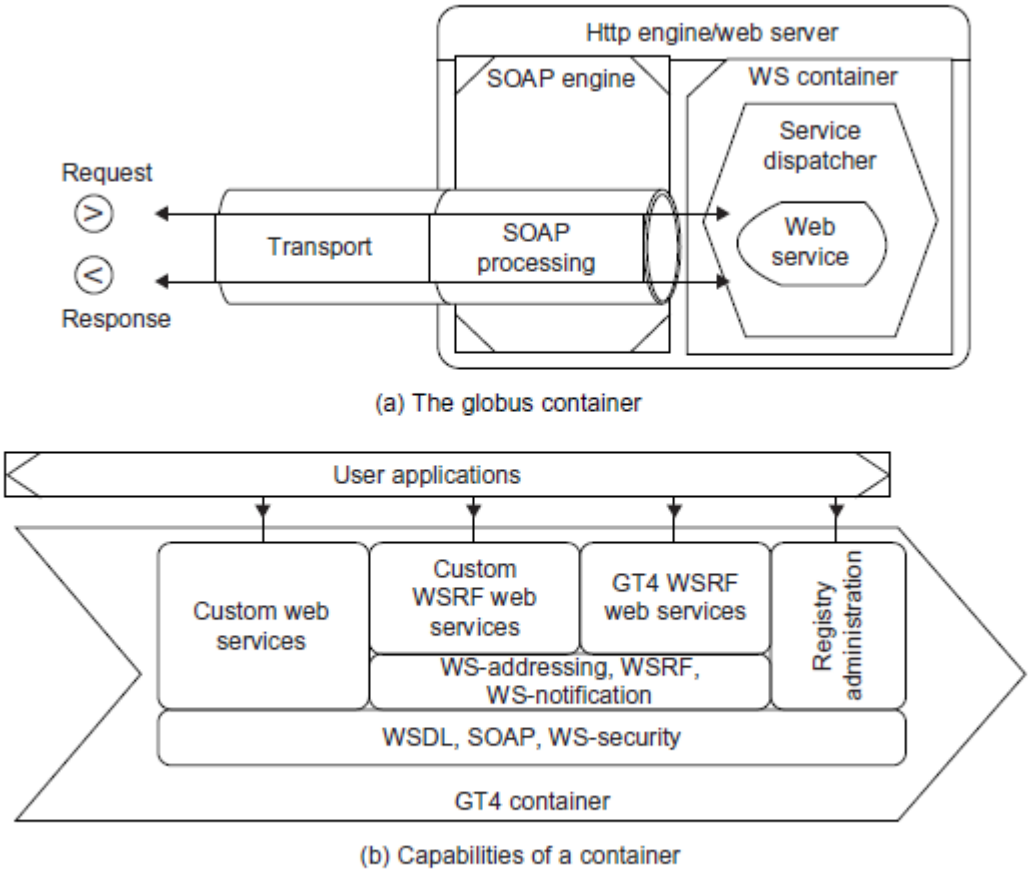


FIGURE 7.21

Globus container serving as a runtime environment for implementing web services in a grid platform.

reuse.

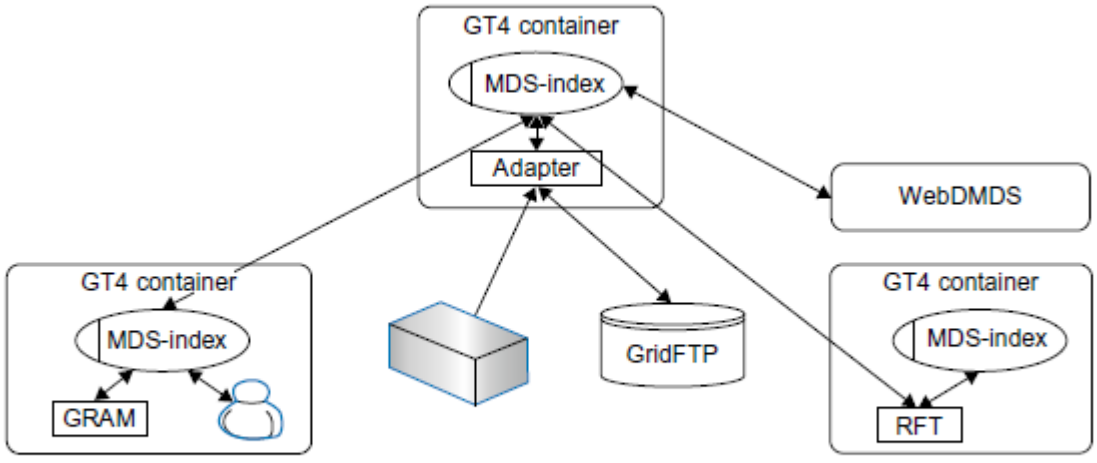


FIGURE 7.22

GT4 system monitoring and resource discovery infrastructure.

3. Explain in detail about the Usage of globus procedure

What is Globus?

- ✓ The Globus Toolkit is an open architecture, open source software toolkit. It facilitates the creation of computational Grids. It provides software tools enabling coupling of people, computers, databases and instruments. With Globus, you can run job on two or more high-performance machines at the same time, even though the machines might be located far apart and owned by different organizations.
- ✓ Globus software helps scientists deal with very large datasets and complex remote collaborations.Globus software is used for large distributed computational jobs, remote instrumentation and remote data transfer.

Three main components of Globus are:

4. Security
5. Information Infrastructure
6. Resource Management

i.Security

- ✓ The Globus Toolkit uses the Grid Security Infrastructure (GSI) for secure communication over an open network. It provides a number of useful services for Grids including mutual authentication and single sign-on. GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol.
- ✓ Extensions to these standards have been added for single sign-on and delegation. The Globus Toolkit's implementation of the GSI adheres to the Generic Security Service API (GSS-API), which is a standard API for security systems promoted by the Internet Engineering Task Force (IETF).
- ✓ A central concept in GSI authentication is the certificate. Every user and service on the Grid is identified via a certificate, which contains information vital to identifying and authenticating the user or service.

A GSI certificate includes four primary pieces of information:

- A subject name, which identifies the person or object that the certificate represents.
- The public key belonging to the subject.
- The identity of a Certificate Authority (CA) that has signed the certificate to certify that the public key and the identity both belong to the subject.
- The digital signature of the named CA.

GSI certificates are encoded in the X.509 certificate format, a standard data format for certificates established by the Internet Engineering Task Force (IETF).

Signing Onto the Grid: Creating a Proxy Certificate

- ✓ Proxies are certificates signed by the user, or by another proxy, that do not require a password to submit a job. They are intended for short-term use, when the user is submitting many jobs and cannot be troubled to repeat his password for every job.
- ✓ The subject of a proxy certificate is the same as the subject of the certificate that signed it, with /CN=proxy added to the name. (The gatekeeper will accept any job requests submitted by the user, as well as any proxies he has created.)
- ✓ Proxies provide a convenient alternative to constantly entering passwords, but are also less secure than the user's normal security credential. Therefore, they should always be user-readable only, and should be deleted after they are no longer needed (or after they expire).

To create a proxy with the default expiration (12 hours), run the grid-proxy-init program.
For example:

% grid-proxy-init

The grid-proxy-init program can also take arguments to specify the expiration and proxy key length.
For example:

% grid-proxy-init -hours 8 -bits 512

To delete a proxy that was previously created with grid-proxy-init, run:

% grid-proxy-destroy

ii. Information Infrastructure

- ✓ Information is a critical resource in computational grids. The Globus Monitoring and Discovery Service (MDS) provides the necessary tools to build an LDAP(LightWeight Directory Access Protocol) -based information infrastructure for computational grids.
- ✓ MDS uses the LDAP protocol as a uniform means of querying system information from a rich variety of system components, and for optionally constructing a uniform namespace for resource information across a system that may involve many organizations.

The Grid Resource Information Service (GRIS) provides a uniform means of querying resources on a computational grid for their current status. Such resources include:

a) computational nodes
b) data storage systems
c) scientific instruments
d) network links
e) databases

- ✓ The Grid Index Information Service (GIIS) provides a means of knitting together arbitrary GRIS services to provide a coherent system image that can be explored or searched by grid applications. For example, a GIIS could list all of the computational resources available within a confederation of laboratories,or all of the distributed data storage systems owned by a particular agency.
- ✓ A GIIS could pool information about all of the grid resources (computation, data, networks, instruments) in a particular research consortium, thus providing a coherent system image of that consortium's computational grid.

An overview of the software components (GridFTP, Replica Catalog, Replica Management)

GridFTP: A high-performance, secure, robust data transfer mechanism

Globus Replica Catalog: A mechanism for maintaining a catalog of dataset replicas.

Globus Replica Management: A mechanism that ties together the Replica Catalog and GridFTP technologies, allowing applications to create and manage replicas of large datasets.

iii. Resource Management:

What is a Gatekeeper ?

- ✓ Gatekeeper is the process which exists on the remote computer before any request is submitted.It runs as a root and handles the job requests. When the gatekeeper receives an allocation request from a client, it mutually authenticates with the client,maps the requestor to a local user, starts a job manager on the local host as the local user,and passes the allocation arguments to the newly created job manager.

What is a Job Manager ?

- ✓ A single job manager is created by the gatekeeper to fulfill every request submitted to the gatekeeper. It starts the job on the local system, and handles all further communication with the client.

It is made up of two components:

a) Common Component - translates messages received from the gatekeeper and client into an internal API that is implemented by the machine specific component. It also translates callback requests from the machine specific components through the internal API into messages to the application manager.

b) Machine-Specific Component - implements the internal API in the local environment. This includes calls to the local system, messages to the resource monitor, and inquiries to the MDS.

Globus Resource Allocation Manager (GRAM)

- ✓ The Globus Resource Allocation Manager (GRAM) processes the requests for resources for remote application execution, allocates the required resources, and manages the active jobs. It also returns updated information regarding the capabilities and availability of the computing resources to the Metacomputing Directory Service (MDS).
- ✓ GRAM provides an API for submitting and canceling a job request, as well as checking the status of a submitted job.The specifications are written by the user in the Resource Specification Language (RSL), and is processed by GRAM as part of the job request.
- ✓ The Globus Resource Allocation Manager (GRAM) is the lowest level of Globus resource management architecture.
- ✓ GRAM allows you to run jobs remotely, providing an API for submitting, monitoring, and terminating your job. When a job is submitted,the request is sent to the gatekeeper of the remote computer. The gatekeeper handles the request and creates a job manager for the job.
- ✓ The job manager starts and monitors the remote program, communicating state changes back to the user on the local machine.
- ✓ When the remote application terminates, normally or by failing, the job manager terminates as well. GRAM is responsible for Parsing and processing the Resource Specification Language (RSL) specifications that outline job requests. The request specifies resource selection, job process creation, and job control.
- ✓ This is accomplished by either denying the request or creating one or more processes (jobs) to satisfy the request. Enabling remote monitoring and managing of jobs already created. The Resource Specification Language (RSL) is a structured language by which resource requirements and parameters can be outlined by a user.
- ✓ To run a job remotely, a GRAM gatekeeper (server) must be running on a remote computer, listening at a port; and the application needs to be compiled on that remote machine. The execution begins when a GRAM user application runs on the local machine, sending a job request to the remote computer.
- ✓ The executable, stdin and stdout, as well as the name and port of the remote computer, are specified as part of the job request. The job request is handled by the gatekeeper, which creates a job manager for the new job. The job manager handles the execution of the job, as well as any communication with the user. Monitoring and Discovery Service (MDS) is the information services component of the Globus Toolkit.

Commands to Run a Job

a) globus-job-run

- ✓ globus-job-run is the basic command for running Globus jobs. globus-job-run runs in the foreground and defaults to sending output to your terminal. It is equivalent to rsh, but has more functionality for running complex jobs on the Grid.

b) globus-job-submit

- ✓ globus-job-submit is for submitting jobs to a remote batch job scheduler such as Condor or the Portable Batch System (PBS) or LSF. With globus-job-submit, you can submit a job, log out, and log back in later to collect the output. That is, globus-job-submit runs in the background and defaults to sending output to the machine running the command. Retrieve output with globus-job-get-output, and then clean up with globus-job-clean.

c) globusrun

- ✓ The globusrun command runs specifications written in the Globus Resource Specification Language (RSL). globusrun can run jobs either in the foreground or background, and can send output to your terminal or to the machine running the command. The trend in Globus software development is toward considering globusrun as middleware, which can be used by application specific shell scripts to manage

job submission. In fact, globus-job-run and globus-job-submit are simply shell scripts that use globusrun for job submission, but present a simpler interface to users.

d) mpirun

- ✓ MPICH-G2, an implementation of the Message Passing Interface (MPI) standard based on Globus, has an mpirun command that can be used to submit MPI parallel jobs to multiple computers in a Grid.

User's Guide to Globus:

For using globus you should have a valid account and X.509 certificate in your .globus directory in your account.

1) If you already have a valid account and certificate and proxy running the command

% globusrun -a -r <hostname>

Example: globusrun -a -r gibhead.ccs.uky.edu
should give you "GRAM Authentication test successful message."

2) Now you can try running simple job on the local cluster using globus-job-run command

% globus-job-run <hostname> </path/executable> <arguments>

Example:
% globus-job-run gibhead.ccs.uky.edu /bin/echo hello world
Prints the Output
hello world

3) Running job on the remote machine gib01.bu.edu from your local machine

% globus-job-run <hostname> </path/executable> <arguments>

Example:
% globus-job-run gib01.bu.edu /bin/echo hello world
Prints the Output
hello world

4) To run a script, open a editor and try the below script
Example:

```
#! /bin/csh -f
echo "Hello World from "; $GLOBUS_LOCATION/bin/globus-hostname
echo arg 1 = $1
echo arg 2 = $2
echo -n "sum is "
echo "$1+$2" | /usr/bin/bc -l
```

and then save the script with hw as the filename. To run the script locally change the permissions on the file with the below command
% chmod +x hw

Then use globus-job-run to run your file. If you are trying to run the script on the local cluster use the below command
Syntax:
% globus-job-run <hostname> <path/executable> <arguments>
For example, if your hw file is located on gibhead.ccs.uky.edu
% globus-job-run gibhead.ccs.uky.edu ./hw 5 6
Prints:
Hello World from

gibhead.ccs.uky.edu
arg 1 = 5

arg 2 = 6
sum is 11

5) To run your local file on a remote machine there is a concept of staging a file.

Staging Files
If your executable file is on your home machine (the machine from which you are issuing Globus commands) and you wish to run that executable on a remote machine, you will need to stage your executable. To stage your executable file hw, which resides on your home machine, over to a remote machine, execute it, and automatically remove the staged copy after the program has finished, use the -s (-stage) option that is available with globus-job-run:
Syntax:
% globus-job-run <remote host> -stage <executable > <arg arg>
For example:
% globus-job-run gib01.bu.edu -stage ./hw 4 6
prints something like:
Hello Globus World from gib01.bu.edu
sum is 10

6) Subjobs and Multiple Commands
Now run your executable "hw" in several locations from one command. If your hw file happens to be on gibhead.ccs.uky.edu:
% globus-job-run -args 3 5 \
-: gibhead.ccs.uky.edu ./hw \
-: gib01.bu.edu -np 2 -stage ./hw 8 9
Produces output
Hello World from
gibhead.ccs.uky.edu
arg 1 = 3
arg 2 = 5
sum is 8
Hello World from
gib01.bu.edu
arg 1 = 8
arg 2 = 9
The above command with a delimiter '-:' allows you to run the same command in several locations.

4. Explain in detail about Overview of Globus Toolkit 4

- ✓ Globus Toolkit 4 is a collection of open-source components. Many of these are based on existsing standards, while others are based on (and in some cases driving) evolving standards. Version 4 of the toolkit is the first version to support Web service based implementations of many of its components. (Version 3 had included an OGSI implementation of some components, and Version 2 was not service based at all.)
- ✓ Though many components have Web service based implementations, some do not, and for compatibility and migration reasons, some have both implementations.

Globus Toolkit 4 provides components in the following five categories:

6. Common runtime components
7. Security

8. Data management
9. Information services
10. Execution management

Table 10-1 shows a list of components in Globus Toolkit 4, and identifies those that are Web service based and those that are not. In the sections that follow, we describe each of these in more detail.

Table 10-1 List of components in Globus Toolkit 4

	Web service based components				**Non Web service based components**	
Common runtime components	Java WS Core	C WS Core	Python WS Core		C Common Libraries	eXtensible IO (XIO)
Security components	WS authentication and authorization	Community Authorization Service (CAS)	Delegation service		Pre-WS authentication and authorization	Credential Management
Data management components	Reliable File Transfer (RFT)	OGSA-DAI	Data Replication Service (DRS)		GridFTP	Replica Location Service (RLS)
Monitoring and Discovery Services	Index service	Trigger service	Aggregator Framework	WebMDS	MDS2	

	Web service based components				**Non Web service based components**	
Execution management	WS GRAM	Community Scheduler Framework 4 (CSF4)	Globus Teleoperations Control Protocol (GTCP)	Workspace Management Service (WMS)	Pre WS GRAM	

1. Common runtime components

✓ Globus Toolkit 4 includes common runtime components. Common runtime components consist of libraries and tools needed by both types of implementations and used by most of the other components.

- Java WS Core
- C WS Core
- Python WS Core

Java WS Core

✓ Java WS Core consists of APIs and tools that implement WSRF and WS-Notification standards implemented in Java. These components act as the base components for various default services that Globus Toolkit 4 supplies.

✓ Also, Java WS Core provides the development base libraries and tools for custom WS-RF based services. Figure 10-1 on page 144 shows the relation between Java WS Core and other services.

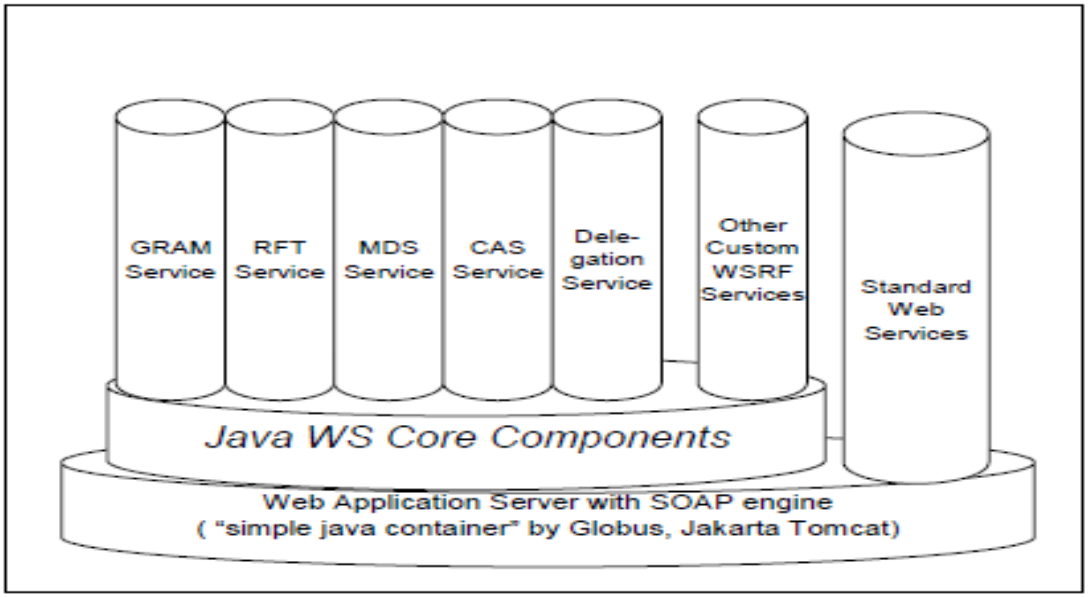


Figure 10-1 Relation between Java WS Core and Globus Toolkit 4 supplied services

C WS Core

- ✓ C WS Core consists of APIs and tools that implement WS-RF and WS-Notification standards using C.

Python WS Core

- ✓ Python WS Core consists of APIs and tools that implement WS-RF and WS-Notification standards with Python. This component is also known as pyGridWare, contributed by Lawrence Berkeley National Laboratory.

2. Security components

- ✓ Because security is one of the most important issues in grid environments, Globus Toolkit 4 includes various types of security components.
  - ✓ WS authentication and authorization
  - ✓ Pre-WS authentication and authorization
  - ✓ Community Authorization Service (CAS)
  - ✓ Delegation service
  - ✓ SimpleCA
  - ✓ MyProxy
  - ✓ GSI-OpenSSH

WS authentication and authorization

- ✓ Globus Toolkit 4 enables message-level security and transport-level security for SOAP communication of Web services. Also, it provides an Authorization Framework for container-level authorization.

Pre-WS authentication and authorization

- ✓ Pre-WS authentication and authorization consists of APIs and tools for authentication, authorization, and certificate management.

Community Authorization Service (CAS)

- ✓ CAS provides access control to virtual organizations. The CAS server grants fine-grained permissions on subsets of resources to members of the community. CAS authorization is currently not available for Web services, but it supports the GridFTP server.

Delegation service

- ✓ The Delegation service enables delegation of credentials between various services in one host. The Delegation service allows a single delegated credential to be used by many services. Also, this service has a credential renewal interface, and this service is capable of extending the valid date of credentials.

SimpleCA

- ✓ SimpleCA is a simplified Certificate Authority. This package has fully functioning CA features for a PKI environment. In Chapter 11, "Globus Toolkit 4 installation and configuration" on page 155, we use SimpleCA as a Certificate Authority for our grid environment.

MyProxy

- ✓ MyProxy is responsible for storing X.509 proxy credentials, protecting them by pass phrase, and enabling an interface for retrieving the proxy credential. MyProxy acts as a repository of credentials, and is often used by Web portalapplications.

GSI-OpenSSH

- ✓ GSI-OpenSSH is a modified version of the OpenSSH client and server that adds support for GSI authentication. GSI-OpenSSH can be used to remotely create a shell on a remote system to run shell scripts or to interactively issue shell commands, and it also permits the transfer of files between systems without being prompted for a password and a user ID. Nevertheless, a valid proxy must be created by using the **grid-proxy-init** command.

3. Data management components

Globus Toolkit 4 provides various tools that enable data management in a grid environment.

- ✓ GridFTP
- ✓ Reliable File Transfer (RFT)
- ✓ Replica Location Service (RLS)
- ✓ OGSA-DAI
- ✓ Data Replication Service (DRS)

GridFTP

- ✓ The GridFTP facility provides secure and reliable data transfer between grid hosts. Its protocol extends the well-known FTP standard to provide additional features, including support for authentication through GSI. One of the major features of GridFTP is that it enables third-party transfer. Third-party transfer is suitable for an environment where there is a large file in remote storage and the client wants to copy it to another remote server, as illustrated in Figure 10-2.

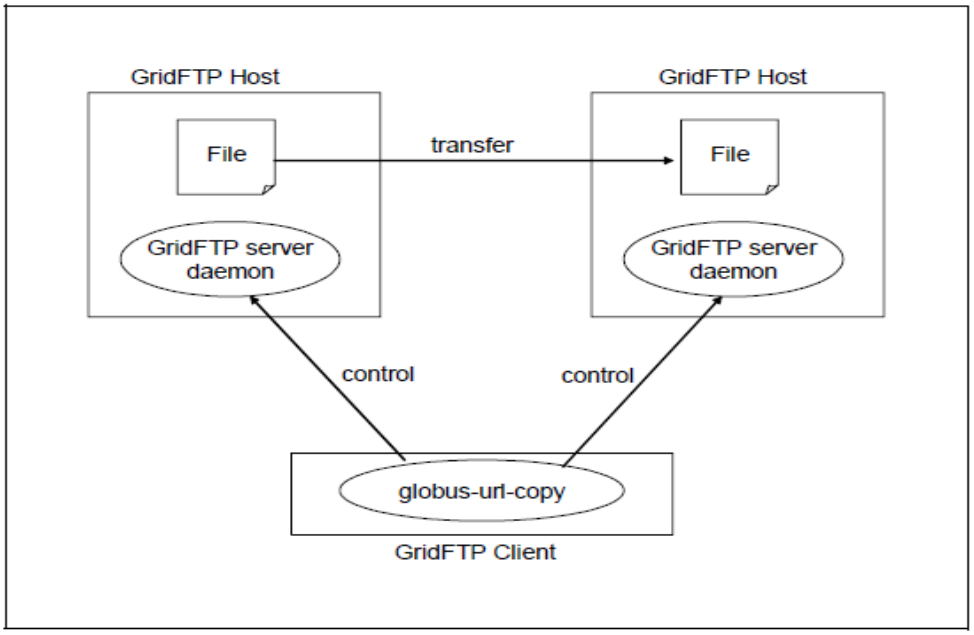


Figure 10-2 GridFTP third-party transfer

Reliable File Transfer (RFT)

- ✓ Reliable File Transfer provides a Web service interface for transfer and deletion of files. RFT receives requests via SOAP messages over HTTP and utilizes GridFTP. RFT also uses a database to store the list of file transfers and their states, and is capable of recovering a transfer request that was interrupted. Figure 10-3 shows how RFT and GridFTP work.

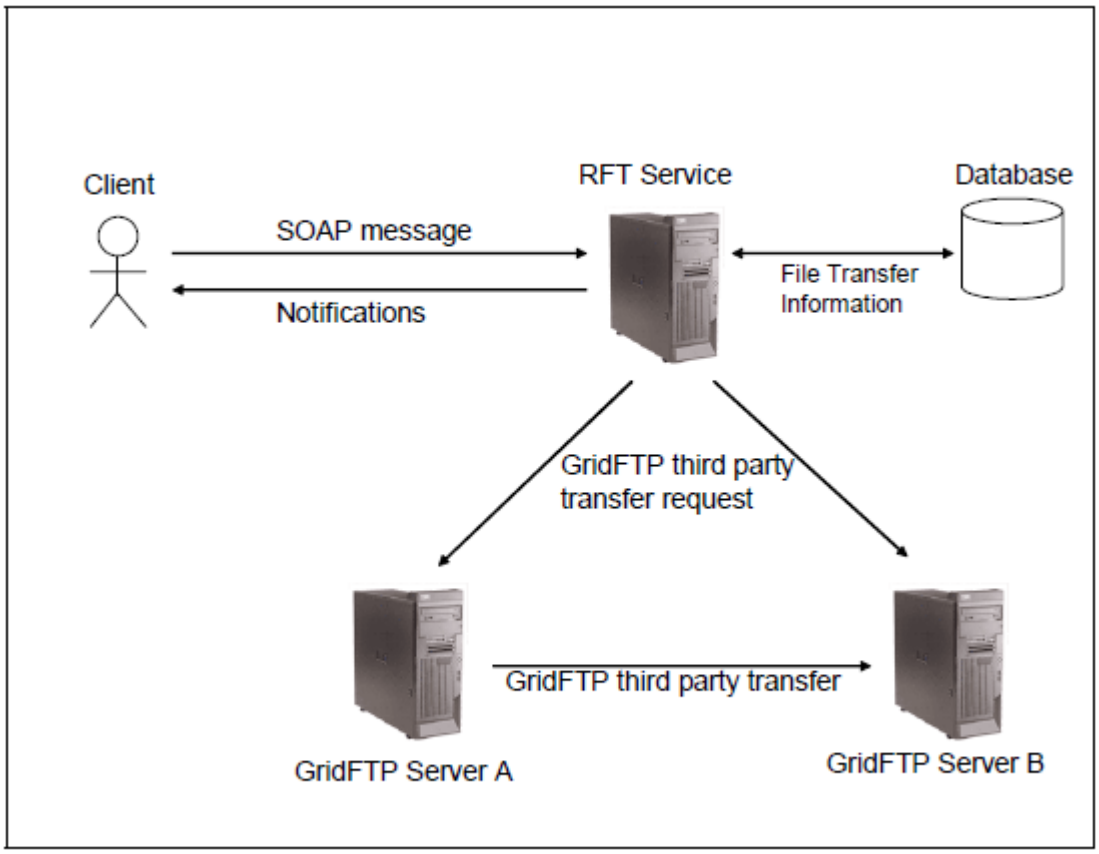


Figure 10-3 How RFT and GridFTP works

Replica Location Service (RLS)

- ✓ The Replica Location Service maintains and provides access to information about the physical locations of replicated data. This component can map multiple physical replicas to one single logical file, and enables data redundancy in a grid environment.

OGSA-DAI

- ✓ OGSA-DAI enables a general grid interface for accessing grid data sources such as relational database management systems and XML repositories, through query languages like SQL, XPat, and XQuery. Currently, OGSA-DAI is a technical preview component. That is, the implementation is functional, but not necessarily complete, and its implementation and interfaces may change in the future.

Data Replication Service (DRS)

- ✓ Data Replication Service provides a system for making replicas of files in the grid environment, and registering them to RLS. DRS uses RFT and GridFTP to transfer the files, and it uses RLS to locate and register the replicas. Currently, DRS is a technical preview component.

4.Monitoring and Discovery Services

- ✓ The Monitoring and Discovery Services (MDS) are mainly concerned with the collection, distribution, indexing, archival, and otherwise processing information about the state of various resources, services, and system configurations.
- ✓ The information collected is used to either discover new services or resources, or to enable monitoring of system status. The GT4 provides a WS-RF and WS-Notification compliant version of MDS, also known as MDS4.
- ✓ The resource properties provided by a WS-RF compliant resource can be registered with MDS4 services for information collection purposes. The GT4 WS-RF compliant services such as GRAM and RFT provide such properties.
- ✓ Upon GT4 container startup these services are registered with MDS4 services. MDS4 consists of two higher-level services, an Index service and a Trigger service, which are based on the Aggregator Framework that is briefly described next.
    - ✓ Index service
    - ✓ Trigger service
    - ✓ Aggregator Framework
    - ✓ WebMDS

Index service

- ✓ The Index service is the central component of the GT4 MDS implementation. Every instance of a GT4 container has a default indexing service (DefaultIndexService) exposed as a WSRF service. The Index service interacts with data sources via standard WS-RF resource property and subscription/notification interfaces (WS-ResourceProperties and WS-BaseNotification).
- ✓ A WSRF-based service can make information available as resource properties. An Index service can potentially collect information from many sources and publish it in only one place.
- ✓ Various WSRF registrations with the Index service are maintained as Service Group Entries by the Index service. The contents of the Index service can be queried via XPath queries. As noted earlier, each GT4 container has a default index service instance registered with it.
- ✓ Therefore, a grid computing site with multiple nodes can potentially have multiple instances of index services available for use. Often virtual organizations configure an instance of Index service to keep track of all relevant resources, containers, and services within their domain.

The following are some of the key features of an Index service:

- ➢ Index services can be configured in hierarchies, but there is no single global index that provides information about every resource on the Grid.

- The presence of a resource in an Index service makes no guarantee about the availability of the resource for users of that Index.
- Information published with MDS is recent but not the absolute latest.
- Each registration into an Index service has a lifetime and requires periodic renewal of registrations to indicate the continued existence of a resource or a service.

Trigger service

- ✓ The MDS Trigger service collects information and compares that data against a set of conditions defined in a configuration file. When a condition is met an action is executed. The condition is specified as an XPath expression; that, for example, may compare the value of a property to a threshold and send an alert e-mail to an administrator by executing a script. The name and location of the script can be configured with the MDS Trigger service.

Aggregator Framework

- ✓ The MDS-Index service and the MDS-Trigger service are specializations of a general Aggregator Framework. The Aggregator Framework is a software framework for building software services that collect and aggregate data. These services are also known as aggregator services.
- ✓ An aggregator service collects information from one of the three types of aggregator sources such as a query source that utilizes WS-ResourceProperty mechanisms to collect data, a subscription source that uses a WS-Notification subscription/notification mechanism to collect data, or an execution source that executes an administrator-provided application to collect information in XML format.

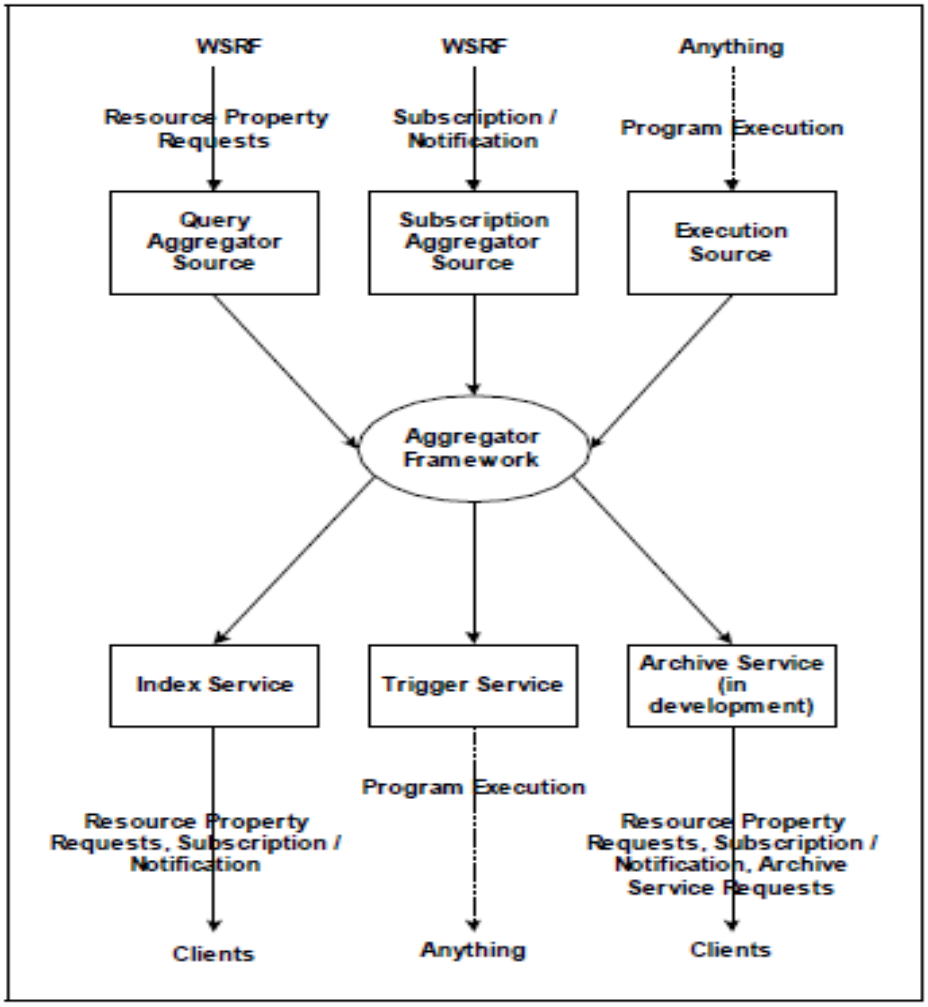


Figure 10-4 MDS4 Aggregator Framework

- ✓ An aggregator source retrieves information from an external component called an information provider. In the case of a query and subscription source, the information provider is a WSRF-compliant service. For an execution source, the information provider is an executable program that obtains data via some application-specific mechanism.

WebMDS

- ✓ WebMDS is a Web-based interface to WS-RF resource property information that can be used as a user-friendly front-end to the Index service. WebMDS uses standard resource property requests to query resource property data and transforms data for a user-friendly display. Web site administrators can customize their own WebMDS deployments by using HTML form options and creating their own XSLT transformations.

5.Execution management

- ✓ Globus Toolkit 4 provides various tools that enable execution management in a grid environment.
  - ➢ WS GRAM
  - ➢ Community Scheduler Framework 4 (CSF4)
  - ➢ Globus Teleoperations Control Protocol (GTCP)
  - ➢ Workspace Management Service (WMS)

WS GRAM

- ✓ WS GRAM is the Grid service that provides the remote execution and status management of jobs. When a job is submitted by a client, the request is sent to the remote host as a SOAP message, and handled by WS GRAM service located in the remote host.
- ✓ The WS GRAM service is capable of submitting those requests to local job schedulers such as Platform LSF or Altair PBS. The WS GRAM service returns status information of the job using WSNotification.
- ✓ The WS GRAM service can collaborate with the RFT service for staging files required by jobs. In order to enable staging with RFT, valid credentials should be delegated to the RFT service by the Delegation service. Figure 10-5 shows how job staging works.

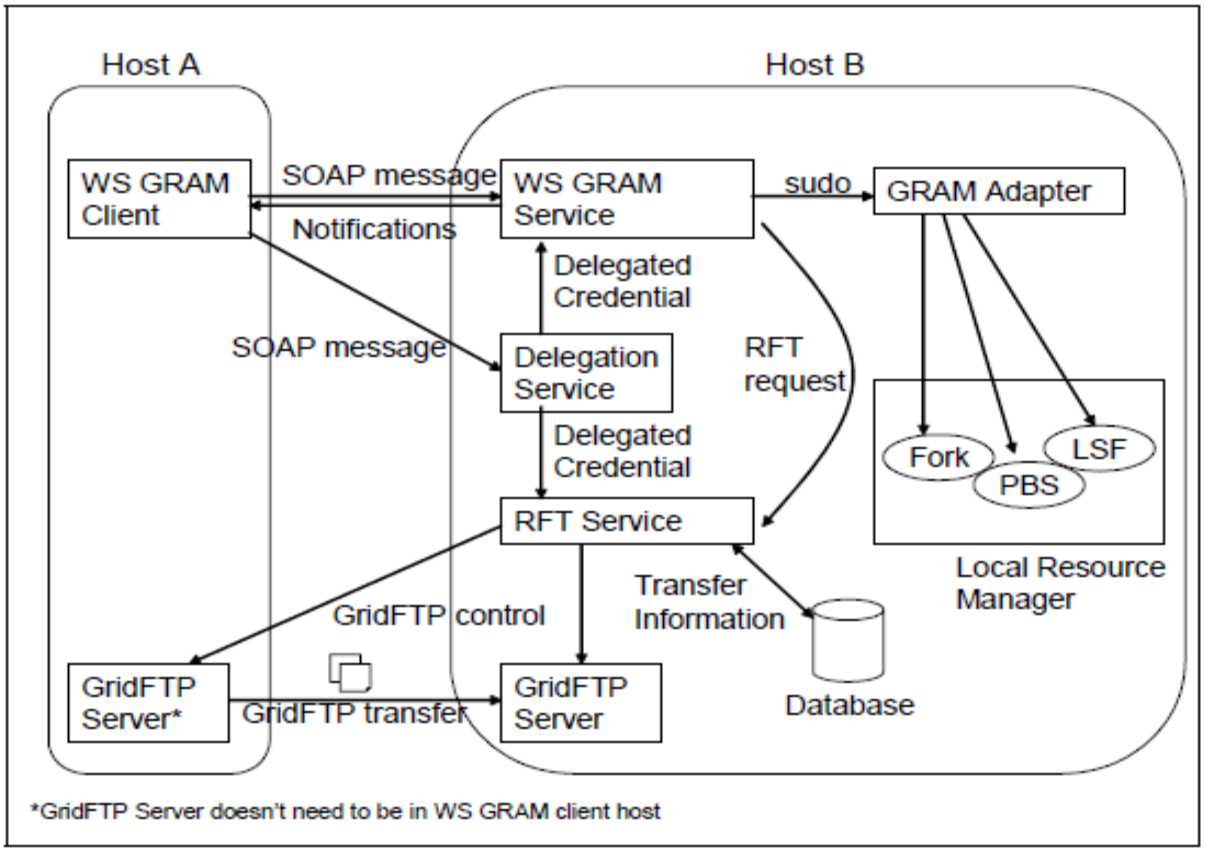


Figure 10-5 Execution of staging job

Community Scheduler Framework 4 (CSF4)

- ✓ The Community Scheduler Framework 4 (CSF4) provides an intelligent,policy-based meta-scheduling facility for building grids where there are multiple types of job schedulers involved.
- ✓ It enables a single interface for different resource managers, such as Platform LSF and Altair PBS. Currently, CSF4 is a technical preview component.

Globus Teleoperations Control Protocol (GTCP)

- ✓ Globus Teleoperations Control Protocol is the WSRF version of NEESgrid Teleoperations Control Protocol (NTCP). Currently, GTCP is a technical preview component.

Workspace Management Service (WMS)

- ✓ The Workspace Management Service enables a grid client to dynamically create,manage, and delete user accounts in a remote site. Currently, WMS is a technical preview component, and only supports management of UNIX accounts.

5.Explain in detail about the Hadoop Framework

- ✓ The Hadoop Framework This section contains a general overview of the components and their association in the Hadoop Framework without getting into the depths of what exactly they do. The Hadoop Framework consists of several modules which provide different parts of the necessary functionality to distribute tasks and data across a cluster.

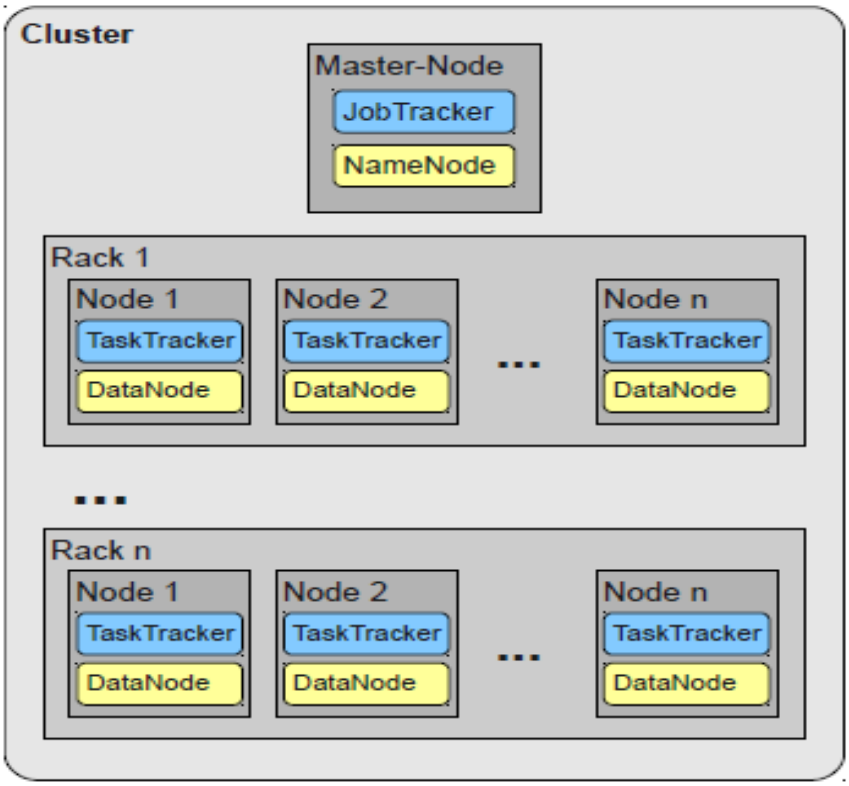


Fig. 1. Abstract view on a Cluster. A cluster consists of several nodes organized into racks, each node running a TaskTracker for the MapReduce tasks and a DataNode for the distributed storage system. One special node, the Master-Node runs the JobTracker and the NameNode which are organizing the distribution of tasks and data.

Hadoop Framework

- ✓ The Hadoop Framework is written in Java but can run MapReduce programs expressed in various languages, e.g. Ruby, Python and C++. This functionality is implemented using Unix standard streams, the input for the Map function is streamed to the program which implements it, no matter which language it is implemented in as long as it can read and write from standard input and to standard output.
- ✓ Another way to achieve this is with the \Hadoop Pipes", an interface to C++ which uses sockets for the communication with the Map and Reduce functions.
- ✓ The central parts of the framework are in the Common module which contains generally availabe interfaces and tools to support the usage of the other modules.
- ✓ These modules include MapReduce, a framework to distribute the processing of large data sets on compute clusters, and HDFS, the Hadoop Distributed Filesys- tem which enables large clusters to save and access data with high throughput.

Communication

- ✓ In every cluster running Hadoop there is one node addressed as the \Master-Node". This node constitutes a single point of failure which is why Hadoop is not a high-availability system. Communication from outside of the cluster is completely handled by the Master-Node, because it keeps all the necessary information about the cluster,usage of the nodes, disk-space and distribution of files.
- ✓ Every MapReduce task is sent to the MasterNode where the JobTracker, the component that manages the MapReduce jobs, and the NameNode, which is responsible for everything concerning the system, are running. The JobTracker communicates with Task-Trackers on the nodes which receive work units and send reports about their status back to the JobTracker.

Cluster

- ✓ The frameworks' con_guration should include information about the toplogy to allow the underlying filesytem the use of the location for its replication strategy. Also it allows the framework to place MapReduce tasks at least near the data they operate on if it happens to be impossible to place the task at the same machine because that way the \not-so-precious" in-rack bandwith is used.

6.Give The Parts of a Hadoop MapReduce Job

- ✓ The user configures and submits a MapReduce job (or just *job* for short) to the framework, which will decompose the job into a set of map tasks, shuffles, a sort, and a set of reduce tasks.
- ✓ The framework will then manage the distribution and execution of the tasks, collect the output, and report the status to the user.

The job consists of the parts shown in Figure 2-1 and listed in Table 2-1.

Table 2-1. *Parts of a MapReduce Job*

Part	Handled By
Configuration of the job	User
Input splitting and distribution	Hadoop framework
Start of the individual map tasks with their input split	Hadoop framework
Map function, called once for each input key/value pair	User
Shuffle, which partitions and sorts the per-map output	Hadoop framework
Sort, which merge sorts the shuffle output for each partition of all map outputs	Hadoop framework
Start of the individual reduce tasks, with their input partition	Hadoop framework
Reduce function, which is called once for each unique input key, with all of the input values that share that key	User
Collection of the output and storage in the configured job output directory, in *N* parts, where *N* is the number of reduce tasks	Hadoop framework

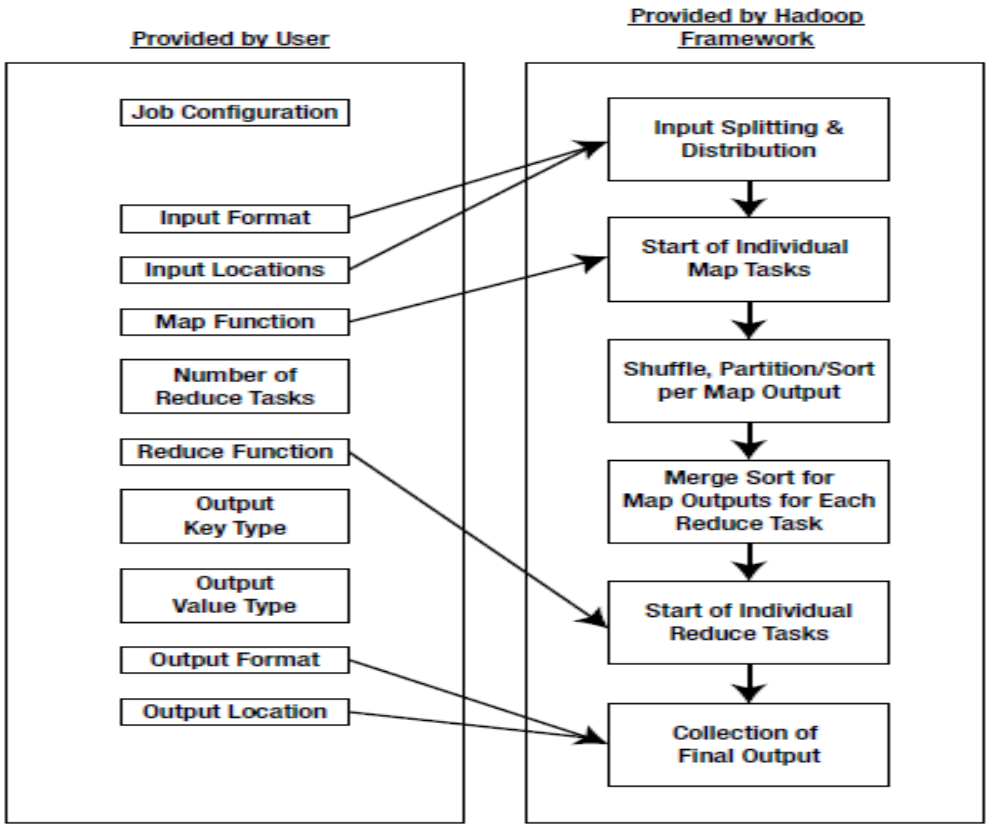


Figure 2-1. *Parts of a MapReduce job*

- ✓ The user is responsible for handling
  - The job setup
  - Specifying the input location(s)
  - Specifying the input and
  - Ensuring the input is in the expected format and location.
- ✓ The framework is responsible for distributing the job among the TaskTracker nodes of the cluster; running the map, shuffle, sort, and reduce phases; placing the output in the output directory; and informing the user of the job-completion status.
- ✓ All the examples in this chapter are based on the file MapReduceIntro.java, shown in Listing 2-1. The job created by the code in MapReduceIntro.java will read all of its textual input line by line, and sort the lines based on that portion of the line before the first tab character.

If there are no tab characters in the line, the sort will be based on the entire line. The MapReduceIntro.java file is structured to provide a simple example of configuring and running a MapReduce job.

```
package com.apress.hadoopbook.examples.ch2;

import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
```

```
import org.apache.hadoop.mapred.RunningJob;
import org.apache.hadoop.mapred.lib.IdentityMapper;
import org.apache.hadoop.mapred.lib.IdentityReducer;
import org.apache.log4j.Logger;

        /** A very simple MapReduce example that reads textual input where
        * each record is a single line, and sorts all of the input lines into
        * a single output file.
        *
        * The records are parsed into Key and Value using the first TAB
        * character as a separator. If there is no TAB character the entire
        * line is the Key. *
        *
        * @author Jason Venner
        *
        */
        public class MapReduceIntro {
        protected static Logger logger = Logger.getLogger(MapReduceIntro.class);
        /**
        * Configure and run the MapReduceIntro job.
        *
        * @param args
        * Not used.
        */
        public static void main(final String[] args) {
                try {
        /** Construct the job conf object that will be used to submit this job
        * to the Hadoop framework. ensure that the jar or directory that
        * contains MapReduceIntroConfig.class is made available to all of the
        * Tasktracker nodes that will run maps or reduces for this job.
        */
        final JobConf conf = new JobConf(MapReduceIntro.class);
        /**
        * Take care of some housekeeping to ensure that this simple example
        * job will run
        */
        MapReduceIntroConfig.
        exampleHouseKeeping(conf,
        MapReduceIntroConfig.getInputDirectory(),
        MapReduceIntroConfig.getOutputDirectory());
        /**
        * This section is the actual job configuration portion /**
        * Configure the inputDirectory and the type of input. In this case
        * we are stating that the input is text, and each record is a
        * single line, and the first TAB is the separator between the key
        * and the value of the record.
        */
conf.setInputFormat(KeyValueTextInputFormat.class);
FileInputFormat.setInputPaths(conf,
MapReduceIntroConfig.getInputDirectory());
```

```
        /** Inform the framework that the mapper class will be the
        * {@link IdentityMapper}. This class simply passes the
        * input Key Value pairs directly to its output, which in
        * our case will be the shuffle.
        */
        conf.setMapperClass(IdentityMapper.class);
        /** Configure the output of the job to go to the output
        * directory. Inform the framework that the Output Key
        * and Value classes will be {@link Text} and the output
        * file format will {@link TextOutputFormat}. The
        * TextOutput format class joins produces a record of
        * output for each Key,Value pair, with the following
        * format. Formatter.format( "%s\t%s%n", key.toString(),
        * value.toString() );.
        *
        * In addition indicate to the framework that there will be
        * 1 reduce. This results in all input keys being placed
        * into the same, single, partition, and the final output
        * being a single sorted file.
        */
FileOutputFormat.setOutputPath(conf,
MapReduceIntroConfig.getOutputDirectory());
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(Text.class);
conf.setNumReduceTasks(1);
        /** Inform the framework that the reducer class will be the {@link
        * IdentityReducer}. This class simply writes an output record key,
        * value record for each value in the key, valueset it receives as
        * input. The value ordering is arbitrary.
        */
conf.setReducerClass(IdentityReducer.class);
logger .info("Launching the job.");
        /** Send the job configuration to the framework and request that the
        * job be run.
        */
        final RunningJob job = JobClient.runJob(conf);
        logger.info("The job has completed.");
        if (!job.isSuccessful()) {
        logger.error("The job failed.");
        System.exit(1);
        }
        logger.info("The job completed successfully.");
        System.exit(0);
        } catch (final IOException e) {
        logger.error("The job has failed due to an IO Exception", e);
        e.printStackTrace();
        }
        }
        }
```

7. Explain Input Splitting concept in Hadoop Framework

- ✓ For the framework to be able to distribute pieces of the job to multiple machines, it needs to fragment the input into individual pieces, which can in turn be provided as input to the individual distributed tasks.
- ✓ Each fragment of input is called an *input split*. The default rules for how input splits are constructed from the actual input files are a combination of configuration parameters and the capabilities of the class that actually reads the input records.
- ✓ An input split will normally be a contiguous group of records from a single input file, and in this case, there will be at least *N* input splits, where *N* is the number of input files. If the number of requested map tasks is larger than this number, or the individual files are larger than the suggested fragment size, there may be multiple input splits constructed of each input file.
- ✓ The user has considerable control over the number of input splits. The number and size of the input splits strongly influence overall job performance.

8. Write short notes on A Simple Map Function: IdentityMapper

- ✓ The Hadoop framework provides a very simple map function, called IdentityMapper. It is used in jobs that only need to reduce the input, and not transform the raw input.

We are going to examine the code of the IdentityMapper class, shown in Listing 2-2, in this section.

Listing 2-2. *IdentityMapper.java*

```
/**
* Licensed to the Apache Software Foundation (ASF) under one
* or more contributor license agreements. See the NOTICE file
* distributed with this work for additional information
* regarding copyright ownership. The ASF licenses this file
* to you under the Apache License, Version 2.0 (the
* "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package org.apache.hadoop.mapred.lib;
import java.io.IOException;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.MapReduceBase;
/** Implements the identity function, mapping inputs directly to outputs. */
public class IdentityMapper<K, V>
extends MapReduceBase implements Mapper<K, V, K, V> {
```

```
        /** The identify function. Input key/value pair is written directly to
        * output.*/
    public void map(K key, V val,
    OutputCollector<K, V> output, Reporter reporter)
    throws IOException {
    output.collect(key, val);
    }
    }
```

- ✓ The magic piece of code is the line output.collect(key, val), which passes a key/value pair back to the framework for further processing. All map functions must implement the Mapper interface, which guarantees that the map function will always be called with a key.
- ✓ The key is an instance of a WritableComparable object, a value that is an instance of a Writable object, an output object, and a reporter. For now, just remember that the reporter is useful. The framework will make one call to your map function for each record in your input.
- ✓ There will be multiple instances of your map function running, potentially in multiple Java Virtual Machines (JVMs), and potentially on multiple machines.

COMMON MAPPERS

One common mapper drops the values and passes only the keys forward:

```
public void map(K key,
                V val,
                OutputCollector<K, V> output,
                Reporter reporter)
    throws IOException {

    output.collect(key, null); /** Note, no value, just a null */

  }
```

Another common mapper converts the key to lowercase:

```
/** put the keys in lower case. */
public void map(Text key,
                V val,
                OutputCollector<Text, V> output,
                Reporter reporter)
    throws IOException {

     Text lowerCaseKey = new Text( key.toString().toLowerCase());
     output.collect(lowerCaseKey, value);

  }
```

7. Write short notes on A Simple Reduce Function: IdentityReducer

- ✓ The Hadoop framework calls the reduce function one time for each unique key. The framework provides the key and the set of values that share that key.
- ✓ The framework-supplied class IdentityReducer is a simple example that produces one output record for every value. Listing 2-3 shows this class.

Listing 2-3. *IdentityReducer.java*

```
        /**
        * Licensed to the Apache Software Foundation (ASF) under one
```

```
* or more contributor license agreements. See the NOTICE file
* distributed with this work for additional information
* regarding copyright ownership. The ASF licenses this file
* to you under the Apache License, Version 2.0 (the
* "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package org.apache.hadoop.mapred.lib;
    import java.io.IOException;
    import java.util.Iterator;
    import org.apache.hadoop.mapred.Reducer;
    import org.apache.hadoop.mapred.OutputCollector;
    import org.apache.hadoop.mapred.Reporter;
    import org.apache.hadoop.mapred.MapReduceBase;
        /** Performs no reduction, writing all input values directly to the output. */
        public class IdentityReducer<K, V>
        extends MapReduceBase implements Reducer<K, V, K, V> {
        /** Writes all keys and values directly to output. */
    public void reduce(K key, Iterator<V> values,
    OutputCollector<K, V> output, Reporter reporter)
    throws IOException {
    while (values.hasNext()) {
    output.collect(key, values.next());
    }
    }
```

- ✓ If you require the output of your job to be sorted, the reducer function must pass the key objects to the output.collect() method unchanged. The reduce phase is, however, free to output any number of records, including zero records, with the same key and different values.
- ✓ This particular constraint is also why the map tasks may be multithreaded, while the reduce tasks are explicitly only single-threaded.

COMMON REDUCERS

A common reducer drops the values and passes only the keys forward:

```
public void map(K key,
                V val,
                OutputCollector<K, V> output,
                Reporter reporter)
    throws IOException {

    output.collect(key, null);

  }
```

Another common reducer provides count information for each key:

```
  protected Text count = new Text();
  /** Writes all keys and values directly to output. */
  public void reduce(K key, Iterator<V> values,
                     OutputCollector<K, V> output, Reporter reporter)
    throws IOException {
    int i = 0;
    while (values.hasNext()) {
        i++
    }
    count.set( "" + i );
    output.collect(key, count);
  }
```

9. Write short notes on Configuring a Job

- ✓ All Hadoop jobs have a driver program that configures the actual MapReduce job and submits it to the Hadoop framework. This configuration is handled through the JobConf object.
- ✓ The sample class MapReduceIntro provides a walk-through for using the JobConf object to configure and submit a job to the Hadoop framework for execution.

The code relies on a class called MapReduceIntroConfig, shown in Listing 2-4, which ensures that the input and output directories are set up and ready.

Listing 2-4. *MapReduceIntroConfig.java*

```
package com.apress.hadoopbook.examples.ch2;
        import java.io.IOException;
        import java.util.Formatter;
        import java.util.Random;
        import org.apache.hadoop.fs.FSDataOutputStream;
        import org.apache.hadoop.fs.FileStatus;
        import org.apache.hadoop.fs.FileSystem;
        import org.apache.hadoop.fs.Path;
        import org.apache.hadoop.mapred.JobConf;
        import org.apache.log4j.Logger;
        /** A simple class to handle the housekeeping for the MapReduceIntro
```

```
        * example job.
        *
        *
        * <p>
        * This job explicitly configures the job to run, locally and without a
        * distributed file system, as a stand alone application.
        * </p>
        * <p>
        * The input is read from the directory /tmp/MapReduceIntroInput and
        * the output is written to the directory
        * /tmp/MapReduceIntroOutput. If the directory
        * /tmp/MapReduceIntroInput is missing or empty, it is created and
        * some input data files generated. If the directory
        * /tmp/MapReduceIntroOutput is present, it is removed.
        * </p>
        *
        * @author Jason Venner
        */
public class MapReduceIntroConfig {
        /**
        * Log4j is the recommended way to provide textual information to the user
        * about the job.
        */
protected static Logger logger =
Logger.getLogger(MapReduceIntroConfig.class);
        /** Some simple defaults for the job input and job output. */
        /**
        * This is the directory that the framework will look for input files in.
        * The search is recursive if the entry is a directory.
        */
protected static Path inputDirectory =
new Path("file:///tmp/MapReduceIntroInput");
        /**
        * This is the directory that the job output will be written to. It must not
        * exist at Job Submission time.
        */
protected static Path outputDirectory =
new Path("file:///tmp/MapReduceIntroOutput");
        /**
        * Ensure that there is some input in the <code>inputDirectory</code>,
        * the <code>outputDirectory</code> does not exist and that this job will
        * be run as a local stand alone application.
        *
        * @param conf
        * The {@link JobConf} object that is required for doing file
        * system access.
        * @param inputDirectory
        * The directory the input will reside in.
        * @param outputDirectory
        * The directory that the output will reside in
```

```
        * @throws IOException
        */
protected static void exampleHouseKeeping(final JobConf conf,
final Path inputDirectory, final Path outputDirectory)
throws IOException {
        /**
        * Ensure that this job will be run stand alone rather than relying on
        * the services of an external JobTracker.
        */
conf.set("mapred.job.tracker", "local");
         that no global file system is required to run this job. */
conf.set("fs.default.name", "file:///");
        /**
        * Reduce the in ram sort space, so that the user does not need to
        * increase the jvm memory size. This sets the sort space to 1 Mbyte,
        * which is very small for a real job.
        */
conf.setInt("io.sort.mb", 1);
        /**
        * Generate some sample input if the <code>inputDirectory</code> is
        * empty or absent.
        */
generateSampleInputIf(conf, inputDirectory);
        /**
        * Remove the file system item at <code>outputDirectory</code> if it
        * exists.
        */
if (!removeIf(conf, outputDirectory)) {
logger.error("Unable to remove " + outputDirectory + "job aborted");
System.exit(1);
}
}
        /**
        * Generate <code>fileCount</code> files in the directory
        * <code>inputDirectory</code>, where the individual lines of the file
        * are a random integer TAB file name.
        *
        * The file names will be file-N where N is between 0 and
        * <code>fileCount</code> - 1. There will be between 1 and
        * <code>maxLines</code> + 1 lines in each file.
        *
        * @param fs
        * The file system that <code>inputDirectory</code> exists in.
        * @param inputDirectory
        * The directory to create the files in. This directory must
        * already exist.
        * @param fileCount
        * The number of files to create.
        * @param maxLines
        * The maximum number of lines to write to the file.
```

```
        * @throws IOException
        */
protected static void generateRandomFiles(final FileSystem fs,
final Path inputDirectory, final int fileCount, final int maxLines)
throws IOException {
final Random random = new Random();
logger .info( "Generating 3 input files of random data," +
"each record is a random number TAB the input file name");
for (int file = 0; file < fileCount; file++) {
final Path outputFile = new Path(inputDirectory, "file-" + file);
final String qualifiedOutputFile = outputFile.makeQualified(fs)
.toUri().toASCIIString();
FSDataOutputStream out = null;
try {
        /**
        * This is the standard way to create a file using the Hadoop
        * Framework. An error will be thrown if the file already
        * exists.
        */
out = fs.create(outputFile);
final Formatter fmt = new Formatter(out);
final int lineCount = (int) (Math.abs(random.nextFloat())
* maxLines + 1);
for (int line = 0; line < lineCount; line++) {
fmt.format("%d\t%s%n", Math.abs(random.nextInt()),
qualifiedOutputFile);
}
fmt.flush();
} finally {
        /**
        * It is very important to ensure that file descriptors are
        * closed. The distributed file system code can run out of file
        * descriptors and the errors generated in that case are
        * misleading.
        */
out.close();
}
}
}
        /**
        * This method will generate some sample input, if the
        * <code>inputDirectory</code> is missing or empty.
        *
        * This method also demonstrates some of the basic APIs for interacting
        * with file systems and files. Note: the code has no particular knowledge
        * of the type of file system.
        *
        * @param conf
        * The Job Configuration object, used for acquiring the
        * {@link FileSystem} objects.
```

```
        * @param inputDirectory
        * The directory to ensure has sample files.
        * @throws IOException
        */
protected static void generateSampleInputIf(final JobConf conf,
final Path inputDirectory) throws IOException {
boolean inputDirectoryExists;
final FileSystem fs = inputDirectory.getFileSystem(conf);
if ((inputDirectoryExists = fs.exists(inputDirectory))
&& !isEmptyDirectory(fs, inputDirectory)) {
if (logger.isDebugEnabled()) {
logger
.debug("The inputDirectory "
+ inputDirectory
+ " exists and is either a"
+ " file or a non empty directory");
}
return;
}
        /**
        * We should only get here if <code>inputDirectory</code> does not
        * exist, or is an empty directory.
        */
if (!inputDirectoryExists) {
if (!fs.mkdirs(inputDirectory)) {
logger.error("Unable to make the inputDirectory "
+ inputDirectory.makeQualified(fs) + " aborting job");
System.exit(1);
}
}
final int fileCount = 3;
final int maxLines = 100;
generateRandomFiles(fs, inputDirectory, fileCount, maxLines);
}
        /**
        * bean access getter to the {@link #inputDirectory} field.
        *
        * @return the value of inputDirectory.
        */
public static Path getInputDirectory() {
return inputDirectory;
}
        /**
        * bean access getter to the {@link outputDirectory} field.
        *
        * @return the value of outputDirectory.
        */
public static Path getOutputDirectory() {
return outputDirectory;
}
```

```
/**
        * Determine if a directory has any non zero files in it or its descendant
        * directories.
        *
        * @param fs
        * The {@link FileSystem} object to use for access.
        * @param inputDirectory
        * The root of the directory tree to search
        * @return true if the directory is missing or does not contain at least one
        * non empty file.
        * @throws IOException
        */
private static boolean isEmptyDirectory(final FileSystem fs,
final Path inputDirectory) throws IOException {
        /**
        * This is the standard way to read a directory's contents. This can be
        * quite expensive for a large directory.
        */
final FileStatus[] statai = fs.listStatus(inputDirectory);
        /**
        * This method returns null under some circumstances, in particular if
        * the directory does not exist.
        */
if ((statai == null) || (statai.length == 0)) {
if (logger.isDebugEnabled()) {
logger.debug(inputDirectory.makeQualified(fs).toUri()
+ " is empty or missing");
}
return true;
}
if (logger.isDebugEnabled()) {
logger.debug(inputDirectory.makeQualified(fs).toUri()
+ " is not empty");
}
        /** Try to find a file in the top level that is not empty. */
for (final FileStatus status : statai) {
if (!status.isDir() && (status.getLen() != 0)) {
if (logger.isDebugEnabled()) {
logger.debug("A non empty file "
+ status.getPath().makeQualified(fs).toUri()
+ " was found");
return false;
}
}
}
        /** Recurse if there are sub directories,
        * looking for a non empty file.
        */
for (final FileStatus status : statai) {
if (status.isDir() && isEmptyDirectory(fs, status.getPath())) {
```

```
continue;
}
        /**
        * If status is a directory it must not be empty or the previous
        * test block would have triggered.
        */
if (status.isDir()) {
return false;
}
}
        /**
        * Only get here if no non empty files were found in the entire subtree
        * of <code>inputPath</code>.
        */
return true;
}
        /**
        * Ensure that the <code>outputDirectory</code> does not exist.
        *
        * <p>
        * The framework requires that the output directory not be present at job
        * submission time.
        * </p>
        * <p>
        * This method also demonstrates how to remove a directory using the
        * {@link FileSystem} API.
        * </p>
        *
        * @param conf
        * The configuration object. This is needed to know what file
        * systems and file system plugins are being used.
        * @param outputDirectory
        * The directory that must be removed if present.
        * @return true if the the <code>outputPath</code> is now missing, or
        * false if the <code>outputPath</code> is present and was unable
        * to be removed.
        * @throws IOException
        * If there is an error loading or configuring the FileSystem
        * plugin, or other IO error when attempting to access or remove
        * the <code>outputDirectory</code>.
        */
protected static boolean removeIf(final JobConf conf,
final Path outputDirectory) throws IOException {
/** This is standard way to acquire a FileSystem object. */
final FileSystem fs = outputDirectory.getFileSystem(conf);
        /**
        * If the <code>outputDirectory</code> does not exist this method is
        * done.
        */
if (!fs.exists(outputDirectory)) {
```

```
if (logger.isDebugEnabled()) {
logger .debug("The output directory does not exist,"
+ " no removal needed.");
}
return true;
}
        /**
        * The getFileStatus command will throw an IOException if the path does
        * not exist.
        */

final FileStatus status = fs.getFileStatus(outputDirectory);
logger.info("The job output directory "
+ outputDirectory.makeQualified(fs) + " exists"
+ (status.isDir() ? " and is not a directory" : "")
+ " and will be removed");
        /**
        * Attempt to delete the file or directory. delete recursively just in
        * case <code>outputDirectory</code> is a directory with
        * sub-directories.
        */
if (!fs.delete(outputDirectory, true)) {
logger.error("Unable to delete the configured output directory "
+ outputDirectory);
return false;
}
        /** The outputDirectory did exist, but has now been removed. */
return true;
}
        /**
        * bean access setter to the {@link inputDirectory} field.
        *
        * @param inputDirectory
        * The value to set inputDirectory to.
        */
public static void setInputDirectory(final Path inputDirectory) {
MapReduceIntroConfig.inputDirectory = inputDirectory;
}
        /**
        * bean access setter for the {@link outpuDirectory field.
        *
        * @param outputDirectory
        * The value to set outputDirectory to.
        */
public static void setOutputDirectory(final Path outputDirectory) {
MapReduceIntroConfig.outputDirectory = outputDirectory;
}
}
```

- ✓ First, you must create a JobConf object. It is good practice to pass in a class that is contained in the JAR file that has your map and reduce functions. This ensures that the framework will make the JAR available to the map and reduce tasks run for your job.

JobConf conf = new JobConf(MapReduceIntro.class);

- ✓ Now that you have a JobConfig object, conf, you need to set the required parameters for the job. These include the input and output directory locations, the format of the input and output, and the mapper and reducer classes.
- ✓ All jobs will have a map phase, and the map phase is responsible for handling the job input. The configuration of the map phase requires you to specify the input locations and the class that will produce the key/value pairs from the input, the mapper class, and potentially, the suggested number of map tasks, map output types, and per-map task threading, as listed in Table 2-2.

Table 2-2. *Map Phase Configuration*

Element	Required?	Default
Input path(s)	Yes	
Class to read and convert the input path elements to key/value pairs	Yes	
Map output key class	No	Job output key class
Map output value class	No	Job output value class
Class supplying the map function	Yes	
Suggested minimum number of map tasks	No	Cluster default
Number of threads to run in each map task	No	1

10. Explain in detail about the Specifying Input Formats

- ✓ The Hadoop framework provides a large variety of input formats. The major distinctions are between textual input formats and binary input formats.
- ✓ The following are the available formats:
  - **KeyValueTextInputFormat**: Key/value pairs, one per line.
  - **TextInputFormant:** The key is the line number, and the value is the line.
  - **NLineInputFormat:** Similar to KeyValueTextInputFormat, but the splits are based on *N* lines of input rather than *Y* bytes of input.
  - **MultiFileInputFormat:** An abstract class that lets the user implement an input format that aggregates multiple files into one split.
  - **SequenceFIleInputFormat:** The input file is a Hadoop sequence file, containing serialized key/value pairs.
- ✓ KeyValueTextInputFormat and SequenceFileInputFormat are the most commonly used input formats. The examples in this chapter use KeyValueTextInputFormat, as the input files are human-readable.

The following block of code informs the framework of the type and location of the job input:

```
/**
* This section is the actual job configuration portion /**
* Configure the inputDirectory and the type of input. In this case
```

```
        * we are stating that the input is text, and each record is a
        * single line, and the first TAB is the separator between the key
        * and the value of the record.
        */
conf.setInputFormat(KeyValueTextInputFormat.class);
FileInputFormat.setInputPaths(conf,
MapReduceIntroConfig.getInputDirectory());
```

- ✓ The line conf.setInputFormat(KeyValueTextInputFormat.class) informs the framework that all of the files used for input will be textual key/value pairs, one per line.

- ✓ Now that the framework knows where to look for the input files and the class to use to generate key/value pairs from the input files, you need to inform the framework which map function to use.

```
        /** Inform the framework that the mapper class will be the {@link
        * IdentityMapper}. This class simply passes the input key-value
        * pairs directly to its output, which in our case will be the
        * shuffle.
        */
conf.setMapperClass(IdentityMapper.class);
```

11. Explain in detail about the Setting the Output Parameters

- ✓ The framework requires that the output parameters be configured, even if the job will not produce any output. The framework will collect the output from the specified tasks (either the output of the map tasks for a MapReduce job that did not include reduce tasks or the output of the job's reduce tasks) and place them into the configured output directory.
- ✓ To avoid issues with file name collisions when placing the task output into the output directory, the framework requires that the output directory not exist when you start the job.
- ✓ In our simple example, the MapReduceIntroConfig class handles ensuring that the output directory does not exist and provides the path to the output directory. The output parameters are actually a little more comprehensive than just the setting of the output path.
- ✓ The code will also set the output format and the output key and value classes.
- ✓ The Text class is the functional equivalent of a String. It implements the WritableComparable interface, which is necessary for keys, and the Writable interface (which is actually a subset of WritableComparable), which is necessary for values.
- ✓ Unlike String, Text is mutable, and the Text class has some explicit methods for UTF-8 byte handling. **The key feature of a Writable is that the framework knows how to** serialize and deserialize a Writable object. The WritableComparable adds the compareTo interface so the framework knows how to sort the WritableComparable objects.
- ✓ The interface references for Writable Comparable and Writable are shown in Listings 2-5 and 2-6.

The following code block provides an example of the minimum required configuration for the output of a MapReduce job:

```
        /** Configure the output of the job to go to the output directory.
        * Inform the framework that the Output Key and Value classes will be
        * {@link Text} and the output file format will {@link
        * TextOutputFormat}. The TextOutput format class produces a record of
        * output for each Key,Value pair, with the following format.
        * Formatter.format( "%s\t%s%n", key.toString(), value.toString() );.
        * In addition indicate to the framework that there will be
        * 1 reduce. This results in all input keys being placed
        * into the same, single, partition, and the final output
```

```
        * being a single sorted file.
        */
FileOutputFormat.setOutputPath(conf,
MapReduceIntroConfig.getOutputDirectory());
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(Text.class);
```

- ✓ The FileOutputFormat.setOutputPath(conf, MapReduceIntroConfig.getOutputDirectory()) setting is familiar from the input example discussed earlier in the chapter.
- ✓ The conf.setOutputKeyClass(Text.class) and conf.setOutputValueClass(Text.class) settings are new. These settings inform the framework of the types of the key/value pairs to expect for the reduce phase. By default, these classes will also be used to set the values the framework will expect from the map output. Unsurprisingly, the method to set the output key class for the map output is conf.setMapOutputKeyClass(Class<? extends WritableComparable>).
- ✓ To set the output value class, the method is conf.setMapOutputValueClass(Class<? extends Writable>).

Listing 2-5. *WritableComparable.java*

```
        /**
        * Licensed to the Apache Software Foundation (ASF) under one
        * or more contributor license agreements. See the NOTICE file
        * distributed with this work for additional information
        * regarding copyright ownership. The ASF licenses this file
        * to you under the Apache License, Version 2.0 (the
        * "License"); you may not use this file except in compliance
        * with the License. You may obtain a copy of the License at
        *
        * http://www.apache.org/licenses/LICENSE-2.0
        *
        * Unless required by applicable law or agreed to in writing, software
        * distributed under the License is distributed on an "AS IS" BASIS,
        * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
        * See the License for the specific language governing permissions and
        * limitations under the License.
        */

package org.apache.hadoop.io;
        /**
        * A {@link Writable} which is also {@link Comparable}.
        *
        * <p><code>WritableComparable</code>s can be compared to each other, typically
        * via <code>Comparator</code>s. Any type which is to be used as a
        * <code>key</code> in the Hadoop Map-Reduce framework should implement this
        * interface.</p>
        *
        * <p>Example:</p>
        * <p><blockquote><pre>
        * public class MyWritableComparable implements WritableComparable {
        * // Some data
        * private int counter;
        * private long timestamp;
```

```
*
* public void write(DataOutput out) throws IOException {
* out.writeInt(counter);
* out.writeLong(timestamp);
* }
*
* public void readFields(DataInput in) throws IOException {
* counter = in.readInt();
* timestamp = in.readLong();
* }
*
* public int compareTo(MyWritableComparable w) {
* int thisValue = this.value;
* int thatValue = ((IntWritable)o).value;
* return (thisValue &amp;lt; thatValue ? -1 : (thisValue==thatValue ? 0 : 1));
* }
* }
* </pre></blockquote></p>
*/
public interface WritableComparable<T> extends Writable, Comparable<T> {
}
```

Listing 2-6. *Writable.java*

```
/**
* Licensed to the Apache Software Foundation (ASF) under one
* or more contributor license agreements. See the NOTICE file
* distributed with this work for additional information
* regarding copyright ownership. The ASF licenses this file * to you under the Apache License,
Version 2.0 (the
* "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package org.apache.hadoop.io;
import java.io.DataOutput;
import java.io.DataInput;
import java.io.IOException;
/**
* A serializable object which implements a simple, efficient, serialization
* protocol, based on {@link DataInput} and {@link DataOutput}.
*
* <p>Any <code>key</code> or <code>value</code> type in the Hadoop Map-Reduce
* framework implements this interface.</p>
```

```
		*
		* <p>Implementations typically implement a static <code>read(DataInput)</code>
		* method which constructs a new instance, calls {@link #readFields(DataInput)}
		* and returns the instance.</p>
		*
		* <p>Example:</p>
		* <p><blockquote><pre>
		* public class MyWritable implements Writable {
		* // Some data
		* private int counter;
		* private long timestamp;
		*
		* public void write(DataOutput out) throws IOException {
		* out.writeInt(counter);
		* out.writeLong(timestamp);
		* }
		*
		* public void readFields(DataInput in) throws IOException {
		* counter = in.readInt();
		* timestamp = in.readLong();
		* }

		* public static MyWritable read(DataInput in) throws IOException {
		* MyWritable w = new MyWritable();
		* w.readFields(in);
		* return w;
		* }
		* }
		* </pre></blockquote></p>
		*/
public interface Writable {
		/**
		* Serialize the fields of this object to <code>out</code>.
		*
		* @param out <code>DataOuput</code> to serialize this object into.
		* @throws IOException
		*/
void write(DataOutput out) throws IOException;
		/**
		* Deserialize the fields of this object from <code>in</code>.
		*
		* <p>For efficiency, implementations should attempt to re-use storage in the
		* existing object where possible.</p>
		*
		* @param in <code>DataInput</code> to deseriablize this object from.
		* @throws IOException
		*/
void readFields(DataInput in) throws IOException;
}
```

12. Explain in detail about the Configuring the Reduce Phase

To configure the reduce phase, the user must supply the framework with **five pieces of information**:

vi. The number of reduce tasks; if zero, no reduce phase is run
vii. The class supplying the reduce method
viii. The input key and value types for the reduce task; by default, the same as the reduce output
ix. The output key and value types for the reduce task
x. The output file type for the reduce task output

- ✓ The configured number of reduce tasks determines the number of output files for a job that will run the reduce phase. Tuning this value will have a significant impact on the overall performance of your job.
- ✓ The time spent sorting the keys for each output file is a function of the number of keys. In addition, the number of reduce tasks determines the maximum number of reduce tasks that can be run in parallel.
- ✓ The framework generally has a default number of reduce tasks configured. This value is set by the mapred.reduce.tasks parameter, which defaults to 1.
- ✓ This will result in a single output file containing all of the output keys, in sorted order. There will be one reduce task, run on a single machine that processes every key.

The number of reduce tasks is commonly set in the configuration phase of a job.

```
conf.setNumReduceTasks(1);
```

In general, unless there is a significant need for a single output file, the number of reduce tasks is set to roughly the number of simultaneous execution slots in the cluster.

The reducer class needs to be set only if the number of reduce tasks is not zero. It is very common to not need a reducer, since frequently you do not require sorted output or value grouping by key.

The actual setting of the reducer class is straightforward:

```
/** Inform the framework that the reducer class will be the
* {@link IdentityReducer}. This class simply writes an output record
* key/value record for each value in the key/value set it receives as
* input. The value ordering is arbitrary.
*/
conf.setReducerClass(IdentityReducer.class);
```

A COMMON EXCEPTION

The framework relies on the output parameters being set correctly. One of the more common errors is to have each reduce task fail with an exception of the form:

```
java.io.IOException: Type mismatch in key from map: expected
org.apache.hadoop.io.LongWritable, recieved org.apache.hadoop.io.Text
```

This error indicates that output key class has been defaulted by the framework, or was set incorrectly during the job configuration.

To correct this, use the following:

```
conf.setOutputKeyClass( Text.class )
```

Or if your map output is not the same as your job output, use this form:

```
conf.setMapOutputKeyClass( Text.class )
```

This error may occur for the value class as well:

```
java.io.IOException: Type mismatch in value from map: expected
org.apache.hadoop.io.LongWritable, recieved org.apache.hadoop.io.Text
```

The corresponding `setOutputValueClass()` or `setMapOutputValue()` class methods are needed to correct this.

13. Explain in detail about the Running a Job

- ✓ The ultimate aim of all your MapReduce job configuration is to actually run that job. The MapReduceIntro.java example (Listing 2-1) demonstrates a common and simple way to run a job:

```
logger .info("Launching the job.");
    /** Send the job configuration to the framework
    * and request that the job be run.
    */
final RunningJob job = JobClient.runJob(conf);
logger.info("The job has completed.");
```

- ✓ The method runJob() submits the configuration information to the framework and waits for the framework to finish running the job. The response is provided in the job object. The RunningJob class provides a number of methods for examining the response. Perhaps the most useful is job.isSuccessful().

Run MapReduceIntro.java as follows (using the CH2.jar file provided with this book's downloadable code):

```
hadoop jar DOWNLOAD_PATH/ch2.jar ➥
com.apress.hadoopbook.examples.ch2.MapReduceIntro
```

The response should be as follows:

```
ch2.MapReduceIntroConfig: Generating 3 input files of random data, each record is a random number TAB the input file name
ch2.MapReduceInt o: Launching the job.
jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
    mapred.JobClient: Use GenericOptionsParser for parsing the arguments.
Applications should implement Tool for the same.
    mapred.FileInputFormat: Total input paths to process : 3
    mapred.FileInputFormat: Total input paths to process : 3
    mapred.FileInputFormat: Total input paths to process : 3
```

```
        mapred.FileInputFormat: Total input paths to process : 3
        mapred.JobClient: Running job: job_local_0001
        mapred.MapTask: numReduceTasks: 1
        mapred.MapTask: io.sort.mb = 1
        mapred.MapTask: data buffer = 796928/996160
        mapred.MapTask: record buffer = 2620/3276
        mapred.MapTask: Starting flush of map output
        mapred.MapTask: bufstart = 0; bufend = 664; bufvoid = 996160
        mapred.MapTask: kvstart = 0; kvend = 14; length = 3276
        mapred.MapTask: Index: (0, 694, 694)
        mapred.MapTask: Finished spill 0
        mapred.LocalJobRunner: file:/tmp/MapReduceIntroInput/file-2:0+664
        mapred.TaskRunner: Task 'attempt_local_0001_m_000000_0' done.
        mapred.TaskRunner: Saved output of task 'attempt_local_0001_m_000000_0' to
file:/tmp/MapReduceIntroOutput
        mapred.MapTask: numReduceTasks: 1
        mapred.MapTask: io.sort.mb = 1
        mapred.MapTask: data buffer = 796928/996160
        mapred.MapTask: record buffer = 2620/3276
        mapred.MapTask: Starting flush of map output
        mapred.MapTask: bufstart = 0; bufend = 3418; bufvoid = 996160
        mapred.MapTask: kvstart = 0; kvend = 72; length = 3276
        mapred.MapTask: Index: (0, 3564, 3564)
        mapred.MapTask: Finished spill 0
        mapred.LocalJobRunner: file:/tmp/MapReduceIntroInput/file-1:0+3418
        mapred.TaskRunner: Task 'attempt_local_0001_m_000001_0' done.
        mapred.TaskRunner: Saved output of task 'attempt_local_0001_m_000001_0' to
file:/tmp/MapReduceIntroOutput
mapred.MapTask: numReduceTasks: 1
        mapred.MapTask: io.sort.mb = 1
        mapred.MapTask: data buffer = 796928/996160
        mapred.MapTask: record buffer = 2620/3276
        mapred.MapTask: Starting flush of map output
        mapred.MapTask: bufstart = 0; bufend = 3986; bufvoid = 996160
        mapred.MapTask: kvstart = 0; kvend = 84; length = 3276
        mapred.MapTask: Index: (0, 4156, 4156)
        mapred.MapTask: Finished spill 0
        mapred.LocalJobRunner: file:/tmp/MapReduceIntroInput/file-0:0+3986
        mapred.TaskRunner: Task 'attempt_local_0001_m_000002_0' done.
        mapred.TaskRunner: Saved output of task 'attempt_local_0001_m_000002_0' to
file:/tmp/MapReduceIntroOutput
        mapred.ReduceTask: Initiating final on-disk merge with 3 files
        mapred.Merger: Merging 3 sorted segments
        mapred.Merger: Down to the last merge-pass, with 3 segments left of total size:8414 bytes
        mapred.LocalJobRunner: reduce > reduce
        mapred.TaskRunner: Task 'attempt_local_0001_r_000000_0' done.
        mapred.TaskRunner: Saved output of task 'attempt_local_0001_r_000000_0' to
file:/tmp/MapReduceIntroOutput
        mapred.JobClient: Job complete: job_local_0001
        mapred.JobClient: Counters: 11
```

```
        mapred.JobClient: File Systems
        mapred.JobClient: Local bytes read=230060
        mapred.JobClient: Local bytes written=319797
        mapred.JobClient: Map-Reduce Framework
        mapred.JobClient: Reduce input groups=170
        mapred.JobClient: Combine output records=0
        mapred.JobClient: Map input records=170
        mapred.JobClient: Reduce output records=170
        mapred.JobClient: Map output bytes=8068
        mapred.JobClient: Map input bytes=8068
        mapred.JobClient: Combine input records=0
        mapred.JobClient: Map output records=170
        mapred.JobClient: Reduce input records=170
ch2.MapReduceIntro: The job has completed.
ch2.MapReduceIntro: The job completed successfully.
```

Congratulations, you have run a MapReduce job.

- ✓ The single output file of the reduce task in the file /tmp/MapReduceIntroOutput/part-00000 will have a series of lines of the form *Number* TAB file:/tmp/MapReduceIntroInput/file-*N*. The first thing you will notice is that the numbers don't seem to be in order.
- ✓ The code that generates the input produces a random number for the key of each line, but the example tells the framework that the keys are Text. Therefore, the numbers have been sorted as text rather than as numbers.

14. Explain in detail about the Design of Hadoop file system or The Design of HDFS

- ✓ HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware. Let's examine this statement in more detail:

Very large files

- ✓ "Very large" in this context means files that are hundreds of megabytes, gigabytes, or terabytes in size. There are Hadoop clusters running today that store petabytes of data.

Streaming data access

- ✓ HDFS is built around the idea that the most efficient data processing pattern is a write-once, read-many-times pattern. A dataset is typically generated or copied from source, then various analyses are performed on that dataset over time.
- ✓ Each analysis will involve a large proportion, if not all, of the dataset, so the time to read the whole dataset is more important than the latency in reading the first record.

Commodity hardware

- ✓ Hadoop doesn't require expensive, highly reliable hardware to run on. It's designed to run on clusters of commodity hardware (commonly available hardware available from multiple vendors3) for which the chance of node failure across the cluster is high, at least for large clusters.
- ✓ HDFS is designed to carry on working without a noticeable interruption to the user in the face of such failure. It is also worth examining the applications for which using HDFS does not work so well. While this may change in the future, these are areas where HDFS is not a good fit today:

Low-latency data access

- ✓ Applications that require low-latency access to data, in the tens of milliseconds range, will not work well with HDFS. Remember, HDFS is optimized for delivering a high throughput of data, and this may be at the expense of latency.

Lots of small files

- ✓ Since the namenode holds filesystem metadata in memory, the limit to the number of files in a filesystem is governed by the amount of memory on the namenode. As a rule of thumb, each file, directory, and block takes about 150 bytes.
- ✓ So, for example, if you had one million files, each taking one block, you would need at least 300 MB of memory. While storing millions of files is feasible, billions is beyond the capability of current hardware.4

Multiple writers, arbitrary file modifications

- ✓ Files in HDFS may be written to by a single writer. Writes are always made at the end of the file. There is no support for multiple writers, or for modifications at arbitrary offsets in the file. (These might be supported in the future, but they are likely to be relatively inefficient.)

15. Explain in detail about the HDFS Concepts

HDFS Concepts -

v. Blocks
vi. Namenodes and Datanodes
vii. HDFS Federation
viii. HDFS High-Availability

i.Blocks

- ✓ A disk has a block size, which is the minimum amount of data that it can read or write. Filesystems for a single disk build on this by dealing with data in blocks, which are an integral multiple of the disk block size. Filesystem blocks are typically a few kilobytes in size, while disk blocks are normally 512 bytes. This is generally transparent to the filesystem user who is simply reading or writing a file—of whatever length.
- ✓ However, there are tools to perform filesystem maintenance, such as *df* and *fsck*, that operate on the filesystem block level. HDFS, too, has the concept of a block, but it is a much larger unit—64 MB by default.
- ✓ Like in a filesystem for a single disk, files in HDFS are broken into block-sized chunks,which are stored as independent units. Unlike a filesystem for a single disk, a file in HDFS that is smaller than a single block does not occupy a full block's worth of underlying storage. When unqualified, the term "block" in this book refers to a block in HDFS.

Why Is a Block in HDFS So Large?

HDFS blocks are large compared to disk blocks, and the reason is to minimize the cost of seeks. By making a block large enough, the time to transfer the data from the disk can be made to be significantly larger than the time to seek to the start of the block. Thus the time to transfer a large file made of multiple blocks operates at the disk transfer rate.

A quick calculation shows that if the seek time is around 10 ms, and the transfer rate is 100 MB/s, then to make the seek time 1% of the transfer time, we need to make the block size around 100 MB. The default is actually 64 MB, although many HDFS installations use 128 MB blocks. This figure will continue to be revised upward as transfer speeds grow with new generations of disk drives.

This argument shouldn't be taken too far, however. Map tasks in MapReduce normally operate on one block at a time, so if you have too few tasks (fewer than nodes in the cluster), your jobs will run slower than they could otherwise.

- ✓ Having a block abstraction for a distributed filesystem brings several benefits.
- ✓ **The first benefit** is the most obvious: a file can be larger than any single disk in the network. There's nothing that requires the blocks from a file to be stored on the same disk, so they can take advantage of any of the disks in the cluster. In fact, it would be possible, if unusual, to store a single file on an HDFS cluster whose blocks filled all the disks in the cluster.
- ✓ **Second,** making the unit of abstraction a block rather than a file simplifies the storage subsystem. Simplicity is something to strive for all in all systems, but is especially important for a distributed system in which the failure modes are so varied.
- ✓ The storage subsystem deals with blocks, simplifying storage management (since blocks are a fixed size, it is easy to calculate how many can be stored on a given disk) and eliminating metadata concerns (blocks are just a chunk of data to be stored—file metadata such as permissions information does not need to be stored with the blocks, so another system can handle metadata separately).
- ✓ Furthermore, blocks fit well with replication for providing fault tolerance and availability. To insure against corrupted blocks and disk and machine failure, each block is replicated to a small number of physically separate machines (typically three).
- ✓ If a block becomes unavailable, a copy can be read from another location in a way that is transparent to the client. A block that is no longer available due to corruption or machine failure can be replicated from its alternative locations to other live machines to bring the replication factor back to the normal level.
- ✓ Similarly, some applications may choose to set a high replication factor for the blocks in a popular file to spread the read load on the cluster.

Like its disk filesystem cousin, HDFS's fsck command understands blocks. For example,running:
% **hadoop fsck / -files –blocks**

ii.Namenodes and Datanodes

- ✓ An HDFS cluster has two types of node operating in a master-worker pattern:
  - ➢ a ***namenode*** (the master) and
  - ➢ a number of ***datanodes*** (workers).
- ✓ The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. This information is stored persistently on the local disk in the form of two files: the namespace image and the edit log.

- ✓ The namenode also knows the datanodes on which all the blocks for a given file are located, however, it does not store block locations persistently, since this information is reconstructed from datanodes when the system starts.
- ✓ A ***client*** **accesses** the filesystem on behalf of the user by communicating with the namenode and datanodes. The client presents a POSIX-like filesystem interface, so the user code does not need to know about the namenode and datanode to function.
- ✓ Datanodes are the workhorses of the filesystem. They store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing.
- ✓ Without the namenode, the filesystem cannot be used. In fact, if the machine running the namenode were obliterated, all the files on the filesystem would be lost since there would be no way of knowing how to reconstruct the files from the blocks on the datanodes.
- ✓ For this reason, it is important to make the namenode resilient to failure, and Hadoop provides two mechanisms for this.
- ✓ The first way is to back up the files that make up the persistent state of the filesystem metadata. Hadoop can be configured so that the namenode writes its persistent state to multiple filesystems. These writes are synchronous and atomic. The usual configuration choice is to write to local disk as well as a remote NFS mount.
- ✓ It is also possible to run a *secondary namenode*, which despite its name does not act as a namenode. Its main role is to periodically merge the namespace image with the edit log to prevent the edit log from becoming too large.
- ✓ The secondary namenode usually runs on a separate physical machine, since it requires plenty of CPU and as much memory as the namenode to perform the merge. It keeps a copy of the merged namespace image, which can be used in the event of the namenode failing.
- ✓ However, the state of the secondary namenode lags that of the primary, so in the event of total failure of the primary, data loss is almost certain. The usual course of action in this case is to copy the namenode's metadata files that are on NFS to the secondary and run it as the new primary.

iii.HDFS Federation

- ✓ The namenode keeps a reference to every file and block in the filesystem in memory, which means that on very large clusters with many files, memory becomes the limiting factor for scaling HDFS Federation, introduced in the 0.23 release series, allows a cluster to scale by adding namenodes, each of which manages a portion of the filesystem namespace.
- ✓ For example, **one namenode** might manage all the files rooted under */user*, say, and a **second namenode** might handle files under */share*. Under federation, each namenode manages a *namespace volume*, which is made up of the metadata for the namespace, and a *block pool* containing all the blocks for the files in the namespace.
- ✓ Namespace volumes are independent of each other, which means namenodes do not communicate with one another, and furthermore the failure of one namenode does not affect the availability of the namespaces managed by other namenodes.
- ✓ Block pool storage is *not* partitioned, however, so datanodes register with each namenode in the cluster and store blocks from multiple block pools.
- ✓ To access a federated HDFS cluster, clients use client-side mount tables to map file paths to namenodes. This is managed in configuration using the ViewFileSystem, and *viewfs://* URIs.

iv. HDFS High-Availability

- ✓ The combination of replicating namenode metadata on multiple filesystems, and using the secondary namenode to create checkpoints protects against data loss, but does not provide high-availability of the filesystem. The namenode is still a *single point of failure* (SPOF), since if it did fail, all clients—

including MapReduce jobs—would be unable to read, write, or list files, because the namenode is the sole repository of the metadata and the file-to-block mapping.

- ✓ In such an event the whole Hadoop system would effectively be out of service until a new namenode could be brought online. To recover from a failed namenode in this situation, an administrator starts a new primary namenode with one of the filesystem metadata replicas, and configures datanodes and clients to use this new namenode.

  The new namenode is not able to serve requests until it has
  - ➢ Loaded its namespace image into memory,
  - ➢ Replayed its edit log, and
  - ➢ Received enough block reports from the datanodes to leave safe mode.
- ✓ On large clusters with many files and blocks, the time it takes for a namenode to start from cold can be 30 minutes or more. The long recovery time is a problem for routine maintenance too. In fact, since unexpected failure of the namenode is so rare, the case for planned downtime is actually more important in practice.
- ✓ The 0.23 release series of Hadoop remedies this situation by adding support for HDFS high-availability (HA). In this implementation there is a pair of namenodes in an activestandby configuration. In the event of the failure of the active namenode, the standby takes over its duties to continue servicing client requests without a significant interruption.

A few architectural changes are needed to allow this to happen:

- ➢ The namenodes must use highly-available shared storage to share the edit log. (In the initial implementation of HA this will require an NFS filer, but in future releases more options will be provided, such as a BookKeeper-based system built on Zoo-Keeper.) When a standby namenode comes up it reads up to the end of the shared edit log to synchronize its state with the active namenode, and then continues to read new entries as they are written by the active namenode.
- ➢ Datanodes must send block reports to both namenodes since the block mappings are stored in a namenode's memory, and not on disk.
- ➢ Clients must be configured to handle namenode failover, which uses a mechanism that is transparent to users.
- ✓ If the active namenode fails, then the standby can take over very quickly (in a few tens of seconds) since it has the latest state available in memory: both the latest edit log entries, and an up-to-date block mapping.
- ✓ The actual observed failover time will be longer in practice (around a minute or so), since the system needs to be conservative in deciding that the active namenode has failed.
- ✓ In the unlikely event of the standby being down when the active fails, the administrator can still start the standby from cold. This is no worse than the non-HA case, and from an operational point of view it's an improvement, since the process is a standard operational procedure built into Hadoop.
- ✓ Failover and fencing The transition from the active namenode to the standby is managed by a new entity in the system called the *failover controller*. Failover controllers are pluggable, but the first implementation uses ZooKeeper to ensure that only one namenode is active. Each namenode runs a lightweight failover controller process whose job it is to monitor its namenode for failures (using a simple heartbeating mechanism) and trigger a failover should a namenode fail.
- ✓ Failover may also be initiated manually by an adminstrator, in the case of routine maintenance, for example. This is known as a *graceful failover*, since the failover controller arranges an orderly transition for both namenodes to switch roles.
- ✓ In the case of an ungraceful failover, however, it is impossible to be sure that the failed namenode has stopped running. For example, a slow network or a network partition can trigger a failover transition, even though the previously active namenode is still running, and thinks it is still the active namenode.
- ✓ The HA implementation goes to great lengths to ensure that the previously active namenode is prevented from doing any damage and causing corruption—a method known as *fencing*. The system employs a range of fencing mechanisms, including killing the namenode's process, revoking its access to the

shared storage directory (typically by using a vendor-specific NFS command),and disabling its network port via a remote management command.

- ✓ As a last resort, the previously active namenode can be fenced with a technique rather graphically known as *STONITH*, or "shoot the other node in the head", which uses a specialized power distribution unit to forcibly power down the host machine.
- ✓ Client failover is handled transparently by the client library. The simplest implementation uses client-side configuration to control failover. The HDFS URI uses a logical hostname which is mapped to a pair of namenode addresses (in the configuration file), and the client library tries each namenode address until the operation succeeds.

16. Explain in detail about The Command-Line Interface

- ✓ We're going to have a look at HDFS by interacting with it from the command line. There are many other interfaces to HDFS, but the command line is one of the simplest and, to many developers, the most familiar.
- ✓ There are **two properties** that we set in the pseudo-distributed configuration that deserve further explanation.
- ✓ **The first is fs.default.name, set to *hdfs://localhost/*,** which is used to set a default filesystem for Hadoop. Filesystems are specified by a URI, and here we have used an hdfs URI to configure Hadoop to use HDFS by default. The HDFS daemons will use this property to determine the host and port for the HDFS namenode. We'll be running it on localhost, on the default HDFS port, 8020. And HDFS clients will use this property to work out where the namenode is running so they can connect to it.
- ✓ **We set the second property, dfs.replication, to 1** so that HDFS doesn't replicate filesystem blocks by the default factor of three. When running with a single datanode, HDFS can't replicate blocks to three datanodes, so it would perpetually warn about blocks being under-replicated. This setting solves that problem.

Basic Filesystem Operations

- ✓ The filesystem is ready to be used, and we can do all of the usual filesystem operations such as reading files, creating directories, moving files, deleting data, and listing directories.
- ✓ You can type hadoop fs -help to get detailed help on every command. Start by copying a file from the local filesystem to HDFS:

% **hadoop fs -copyFromLocal input/docs/quangle.txt hdfs://localhost/user/tom/quangle.txt**

- ✓ This command invokes Hadoop's filesystem shell command fs, which supports a number of subcommands—in this case, we are running -copyFromLocal. The local file *quangle.txt* is copied to the file */user/tom/quangle.txt* on the HDFS instance running on localhost. In fact, we could have omitted the scheme and host of the URI and picked up the default, hdfs://localhost, as specified in *core-site.xml*:

% **hadoop fs -copyFromLocal input/docs/quangle.txt /user/tom/quangle.txt**

We could also have used a relative path and copied the file to our home directory in HDFS, which in this case is */user/tom*:

% **hadoop fs -copyFromLocal input/docs/quangle.txt quangle.txt**

Let's copy the file back to the local filesystem and check whether it's the same:

% **hadoop fs -copyToLocal quangle.txt quangle.copy.txt**
% **md5 input/docs/quangle.txt quangle.copy.txt**
MD5 (input/docs/quangle.txt) = a16f231da6b05e2ba7a339320e7dacd9
MD5 (quangle.copy.txt) = a16f231da6b05e2ba7a339320e7dacd9

The MD5 digests are the same, showing that the file survived its trip to HDFS and is back intact.
Finally, let's look at an HDFS file listing. We create a directory first just to see how it is displayed in the listing:

52 | Chapter 3: The Hadoop Distributed Filesystem
% **hadoop fs -mkdir books**
% **hadoop fs -ls .**
Found 2 items
drwxr-xr-x - tom supergroup 0 2009-04-02 22:41 /user/tom/books
-rw-r--r-- 1 tom supergroup 118 2009-04-02 22:29 /user/tom/quangle.txt

- ✓ The information returned is very similar to the Unix command ls -l, with a few minor differences.
- ✓ **The first column** shows the file mode.
- ✓ **The second column** is the replication factor of the file (something a traditional Unix filesystem does not have).
- ✓ Remember we set the default replication factor in the site-wide configuration to be 1, which is why we see the same value here. The entry in this column is empty for directories since the concept of replication does not apply to them—directories are treated as metadata and stored by the namenode, not the datanodes.
- ✓ **The third and fourth columns** show the file owner and group.
- ✓ **The fifth column** is the size of the file in bytes, or zero for directories.
- ✓ **The sixth and seventh columns** are the last modified date and time.
- ✓ Finally, **the eighth column** is the absolute name of the file or directory.

17. Explain in detail about the Hadoop Files ystems

- ✓ Hadoop has an abstract notion of filesystem, of which HDFS is just one implementation. The Java abstract class org.apache.hadoop.fs.FileSystem represents a filesystem in Hadoop, and there are several concrete implementations, which are described in Table 3-1.

Table 3-1. Hadoop filesystems

Filesystem	URI scheme	Java implementation (all under org.apache.hadoop)	Description
Local	*file*	`fs.LocalFileSystem`	A filesystem for a locally connected disk with client-side checksums. Use `RawLocalFileSystem` for a local filesystem with no checksums. See "LocalFileSystem" on page 84.
HDFS	*hdfs*	`hdfs.DistributedFileSystem`	Hadoop's distributed filesystem. HDFS is designed to work efficiently in conjunction with MapReduce.
HFTP	*hftp*	`hdfs.HftpFileSystem`	A filesystem providing read-only access to HDFS over HTTP. (Despite its name, HFTP has no connection with FTP.) Often used with *distcp* (see "Parallel Copying with distcp" on page 76) to copy data between HDFS clusters running different versions.
HSFTP	*hsftp*	`hdfs.HsftpFileSystem`	A filesystem providing read-only access to HDFS over HTTPS. (Again, this has no connection with FTP.)
WebHDFS	*webhdfs*	`hdfs.web.WebHdfsFileSystem`	A filesystem providing secure read-write access to HDFS over HTTP. WebHDFS is intended as a replacement for HFTP and HSFTP.
HAR	*har*	`fs.HarFileSystem`	A filesystem layered on another filesystem for archiving files. Hadoop Archives are typically used for archiving files in HDFS to reduce the namenode's memory usage. See "Hadoop Archives" on page 78.
KFS (CloudStore)	*kfs*	`fs.kfs.KosmosFileSystem`	CloudStore (formerly Kosmos filesystem) is a distributed filesystem like HDFS or Google's GFS, written in C++. Find more information about it at *http://kosmosfs.sourceforge.net/*.
FTP	*ftp*	`fs.ftp.FTPFileSystem`	A filesystem backed by an FTP server.
S3 (native)	*s3n*	`fs.s3native.NativeS3FileSystem`	A filesystem backed by Amazon S3. See *http://wiki.apache.org/hadoop/AmazonS3*.
S3 (block-based)	*s3*	`fs.s3.S3FileSystem`	A filesystem backed by Amazon S3, which stores files in blocks (much like HDFS) to overcome S3's 5 GB file size limit.

Filesystem	URI scheme	Java implementation (all under org.apache.hadoop)	Description
Distributed RAID	*hdfs*	`hdfs.DistributedRaidFileSystem`	A "RAID" version of HDFS designed for archival storage. For each file in HDFS, a (smaller) parity file is created, which allows the HDFS replication to be reduced from three to two, which reduces disk usage by 25% to 30%, while keeping the probability of data loss the same. Distributed RAID requires that you run a `RaidNode` daemon on the cluster.
View	*viewfs*	`viewfs.ViewFileSystem`	A client-side mount table for other Hadoop filesystems. Commonly used to create mount points for federated namenodes (see "HDFS Federation" on page 49).

- ✓ Hadoop provides many interfaces to its filesystems, and it generally uses the URI scheme to pick the correct filesystem instance to communicate with. For example, the filesystem shell that we met in the previous section operates with all Hadoop filesystems.
- ✓ To list the files in the root directory of the local filesystem, type:

  % **hadoop fs -ls file:///**

Although it is possible (and sometimes very convenient) to run MapReduce programs that access any of these filesystems, when you are processing large volumes of data, you should choose a distributed filesystem that has the data locality optimization, notably HDFS .

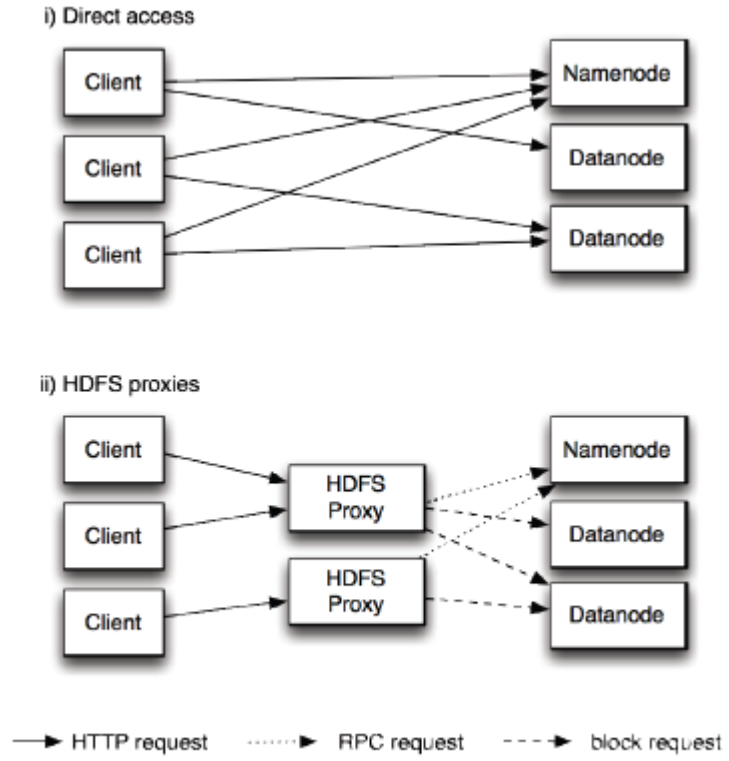


Figure 3-1. Accessing HDFS over HTTP directly, and via a bank of HDFS proxies

Interfaces

- ✓ Hadoop is written in Java, and all Hadoop filesystem interactions are mediated through the Java API. The filesystem shell, for example, is a Java application that uses the Java FileSystem class to provide filesystem operations.
- ✓ The other filesystem interfaces are discussed briefly in this section. These interfaces are most commonly used with HDFS, since the other filesystems in Hadoop typically have existing tools to access the underlying filesystem (FTP clients for FTP, S3 tools for S3, etc.), but many of them will work with any Hadoop filesystem.

HTTP

There are two ways of accessing HDFS over HTTP: directly, where the HDFS daemons serve HTTP requests to clients; and via a proxy (or proxies), which accesses HDFS on the client's behalf using the usual DistributedFileSystem API.

The two ways are illustrated in Figure 3-1.

In the first case, directory listings are served by the namenode's embedded web server (which runs on port 50070) formatted in XML or JSON, while file data is streamed from datanodes by their web servers (running on port 50075).

- ✓ The original direct HTTP interface (HFTP and HSFTP) was read-only, while the new WebHDFS implementation supports all filesystem operations, including Kerberos authentication.
- ✓ WebHDFS must be enabled by setting dfs.webhdfs.enabled to true, for you to be able to use *webhdfs* URIs.

The second way of accessing HDFS over HTTP relies on one or more standalone proxy servers. (The proxies are stateless so they can run behind a standard load balancer.)

- ✓ All traffic to the cluster passes through the proxy. This allows for stricter firewall and bandwidth limiting policies to be put in place. It's common to use a proxy for transfers between Hadoop clusters located in different data centers.

- ✓ The original HDFS proxy (in *src/contrib/hdfsproxy*) was read-only, and could be accessed by clients using the HSFTP FileSystem implementation (*hsftp* URIs). From release 0.23, there is a new proxy called HttpFS that has read and write capabilities, and which exposes the same HTTP interface as WebHDFS, so clients can access either using *webhdfs* URIs.
- ✓ The HTTP REST API that WebHDFS exposes is formally defined in a specification, so it is likely that over time clients in languages other than Java will be written that use it directly.

C

- ✓ Hadoop provides a C library called *libhdfs* that mirrors the Java FileSystem interface (it was written as a C library for accessing HDFS, but despite its name it can be used to access any Hadoop filesystem). It works using the *Java Native Interface* (JNI) to call a Java filesystem client.
- ✓ The C API is very similar to the Java one, but it typically lags the Java one, so newer features may not be supported. You can find the generated documentation for the C API in the *libhdfs/docs/api* directory of the Hadoop distribution. Hadoop comes with prebuilt *libhdfs* binaries for 32-bit Linux, but for other platforms, you will need to build them yourself using the instructions at *http://wiki.apache.org/hadoop/LibHDFS*.

FUSE

- ✓ *Filesystem in Userspace* (FUSE) allows filesystems that are implemented in user space to be integrated as a Unix filesystem. Hadoop's Fuse-DFS contrib module allows any Hadoop filesystem (but typically HDFS) to be mounted as a standard filesystem.
- ✓ You can then use Unix utilities (such as ls and cat) to interact with the filesystem, as well as POSIX libraries to access the filesystem from any programming language.
- ✓ Fuse-DFS is implemented in C using *libhdfs* as the interface to HDFS. Documentation for compiling and running Fuse-DFS is located in the *src/contrib/fuse-dfs* directory of the Hadoop distribution.

18. Explain in detail about the The Java Interface

Reading Data from a Hadoop URL One of the simplest ways to read a file from a Hadoop filesystem is by using a java.net.URL object to open a stream to read the data from.
The general idiom is:

```
InputStream in = null;
try {
in = new URL("hdfs://host/path").openStream();
// process in
} finally {
IOUtils.closeStream(in);
}
```

- ✓ There's a little bit more work required to make Java recognize Hadoop's hdfs URL scheme. This is achieved by calling the setURLStreamHandlerFactory method on URL with an instance of FsUrlStreamHandlerFactory.
- ✓ This method can only be called once per JVM, so it is typically executed in a static block. This limitation means that if some other part of your program—perhaps a third-party component outside your control—sets a URLStreamHandlerFactory, you won't be able to use this approach for reading data from Hadoop.

Example 3-1 shows a program for displaying files from Hadoop filesystems on standard output, like the Unix cat command.

Example 3-1. Displaying files from a Hadoop filesystem on standard output using a URLStreamHandler

```
public class URLCat {

  static {
    URL.setURLStreamHandlerFactory(new FsUrlStreamHandlerFactory());
  }

  public static void main(String[] args) throws Exception {
    InputStream in = null;
    try {
      in = new URL(args[0]).openStream();
      IOUtils.copyBytes(in, System.out, 4096, false);
    } finally {
      IOUtils.closeStream(in);
    }
  }
}
```

- ✓ We make use of the handy IOUtils class that comes with Hadoop for closing the stream in the finally clause, and also for copying bytes between the input stream and the output stream (System.out in this case).
- ✓ The last two arguments to the copyBytes method are the buffer size used for copying and whether to close the streams when the copy is complete. We close the input stream ourselves, and System.out doesn't need to be closed.

Here's a sample run:6

% **hadoop URLCat hdfs://localhost/user/tom/quangle.txt**
On the top of the Crumpetty Tree The Quangle Wangle sat,
But his face you could not see,
On account of his Beaver Hat.

16. Explain in detail about the Reading Data Using the FileSystem API

- ✓ As the previous section explained, sometimes it is impossible to set a URLStreamHand lerFactory for your application. In this case, you will need to use the FileSystem API to open an input stream for a file.
- ✓ A file in a Hadoop filesystem is represented by a Hadoop Path object (and not a java.io.File object, since its semantics are too closely tied to the local filesystem).
- ✓ You can think of a Path as a Hadoop filesystem URI, such as *hdfs://localhost/user/tom/quangle.txt*.
- ✓ FileSystem is a general filesystem API, so the first step is to retrieve an instance for the filesystem we want to use—HDFS in this case.

There are several static factory methods for getting a FileSystem instance:

public static FileSystem get(Configuration conf) throws IOException
public static FileSystem get(URI uri, Configuration conf) throws IOException
public static FileSystem get(URI uri, Configuration conf, String user) throws IOException

- ✓ A Configuration object encapsulates a client or server's configuration, which is set using configuration files read from the classpath, such as *conf/core-site.xml*.

The first method returns the default filesystem (as specified in the file *conf/core-site.xml*, or the default local filesystem if not specified there).

The second uses the given URI's scheme and authority to determine the filesystem to use, falling back to the default filesystem if no scheme is specified in the given URI.

The third retrieves the filesystem as the given user.In some cases, you may want to retrieve a local filesystem instance, in which case you can use the convenience method, getLocal():

public static LocalFileSystem getLocal(Configuration conf) throws IOException

With a FileSystem instance in hand, we invoke an open() method to get the input stream for a file:

public FSDataInputStream open(Path f) throws IOException
public abstract FSDataInputStream open(Path f, int bufferSize) throws IOException

The first method uses a default buffer size of 4 K.
Putting this together, we can rewrite Example 3-1 as shown in Example 3-2.

Example 3-2. Displaying files from a Hadoop filesystem on standard output by using the FileSystem directly

```
public class FileSystemCat {

  public static void main(String[] args) throws Exception {
    String uri = args[0];
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(URI.create(uri), conf);
    InputStream in = null;
    try {
      in = fs.open(new Path(uri));
      IOUtils.copyBytes(in, System.out, 4096, false);
    } finally {
      IOUtils.closeStream(in);
    }
  }
}
```

The program runs as follows:

```
% hadoop FileSystemCat hdfs://localhost/user/tom/quangle.txt
On the top of the Crumpetty Tree
The Quangle Wangle sat,
But his face you could not see,
On account of his Beaver Hat.
```

FSDataInputStream

- ✓ The open() method on FileSystem actually returns a FSDataInputStream rather than a standard java.io class. This class is a specialization of java.io.DataInputStream with support for random access, so you can read from any part of the stream:

package org.apache.hadoop.fs;

public class FSDataInputStream extends DataInputStream
implements Seekable, PositionedReadable {
// implementation elided
}
The Seekable interface permits seeking to a position in the file and a query method for the current offset from the start of the file (getPos()):
public interface Seekable {

```
void seek(long pos) throws IOException;
long getPos() throws IOException;
}
```

- ✓ Calling seek() with a position that is greater than the length of the file will result in an IOException. Unlike the skip() method of java.io.InputStream that positions the stream at a point later than the current position, seek() can move to an arbitrary, absolute position in the file.

Example 3-3 is a simple extension of Example 3-2 that writes a file to standard out twice: after writing it once, it seeks to the start of the file and streams through it once again.

Example 3-3. Displaying files from a Hadoop filesystem on standard output twice, by using seek

```
public class FileSystemDoubleCat {
public static void main(String[] args) throws Exception {
String uri = args[0];
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(URI.create(uri), conf);
FSDataInputStream in = null;
try {
in = fs.open(new Path(uri));
IOUtils.copyBytes(in, System.out, 4096, false);
in.seek(0); // go back to the start of the file
IOUtils.copyBytes(in, System.out, 4096, false);
} finally {
IOUtils.closeStream(in);
}
}
}
```

Here's the result of running it on a small file:

% **hadoop FileSystemDoubleCat hdfs://localhost/user/tom/quangle.txt**

```
On the top of the Crumpetty Tree
The Quangle Wangle sat,
But his face you could not see,
On account of his Beaver Hat.
On the top of the Crumpetty Tree
The Quangle Wangle sat,
But his face you could not see,
On account of his Beaver Hat.
```

FSDataInputStream also implements the PositionedReadable interface for reading parts of a file at a given offset:

```
public interface PositionedReadable {
public int read(long position, byte[] buffer, int offset, int length)
throws IOException;
public void readFully(long position, byte[] buffer, int offset, int length)
throws IOException;
public void readFully(long position, byte[] buffer) throws IOException;
}
```

- ✓ The read() method reads up to length bytes from the given position in the file into the buffer at the given offset in the buffer. The return value is the number of bytes actually read: callers should check this value as it may be less than length.

- ✓ The readFully() methods will read length bytes into the buffer (or buffer.length bytes for the version that just takes a byte array buffer), unless the end of the file is reached, in which case an EOFException is thrown.
- ✓ All of these methods preserve the current offset in the file and are thread-safe, so they provide a convenient way to access another part of the file—metadata perhaps—while reading the main body of the file.
- ✓ In fact, they are just implemented using the Seekable interface using the following pattern:

```
long oldPos = getPos();
try {
seek(position);
// read data
} finally {
seek(oldPos);
}
```

- ✓ Finally, bear in mind that calling seek() is a relatively expensive operation and should be used sparingly. You should structure your application access patterns to rely on streaming data, (by using MapReduce, for example) rather than performing a large number of seeks.

Writing Data

- ✓ The FileSystem class has a number of methods for creating a file. The simplest is the method that takes a Path object for the file to be created and returns an output stream to write to:

```
public FSDataOutputStream create(Path f) throws IOException
```

- ✓ There are overloaded versions of this method that allow you to specify whether to forcibly overwrite existing files, the replication factor of the file, the buffer size to use when writing the file, the block size for the file, and file permissions.

The create() methods create any parent directories of the file to be written that don't already exist. Though convenient, this behavior may be unexpected. If you want the write to fail if the parent directory doesn't exist, then you should check for the existence of the parent directory first by calling the exists() method.

There's also an overloaded method for passing a callback interface, Progressable, so your application can be notified of the progress of the data being written to the datanodes:

```
package org.apache.hadoop.util;
public interface Progressable {
public void progress();
}
```

- ✓ As an alternative to creating a new file, you can append to an existing file using the append() method (there are also some other overloaded versions):

```
public FSDataOutputStream append(Path f) throws IOException
```

- ✓ The append operation allows a single writer to modify an already written file by opening it and writing data from the final offset in the file. With this API, applications that produce unbounded files, such as logfiles, can write to an existing file after a restart, for example.
- ✓ The append operation is optional and not implemented by all Hadoop filesystems. For example, HDFS supports append, but S3 filesystems don't.

Example 3-4 shows how to copy a local file to a Hadoop filesystem. We illustrate progress by printing a period every time the progress() method is called by Hadoop, which is after each 64 K packet of data is written to the datanode pipeline. (Note that this particular behavior is not specified by the API, so it is subject to change in later versions of Hadoop. The API merely allows you to infer that "something is happening.")

Example 3-4. Copying a local file to a Hadoop filesystem

```
public class FileCopyWithProgress {
  public static void main(String[] args) throws Exception {
    String localSrc = args[0];
    String dst = args[1];

    InputStream in = new BufferedInputStream(new FileInputStream(localSrc));

    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(URI.create(dst), conf);
    OutputStream out = fs.create(new Path(dst), new Progressable() {
      public void progress() {
        System.out.print(".");
      }
    });

    IOUtils.copyBytes(in, out, 4096, true);
  }
}
```

Typical usage:

```
% hadoop FileCopyWithProgress input/docs/1400-8.txt hdfs://localhost/user/tom/
      1400-8.txt
---------------
```

- ✓ Currently, none of the other Hadoop filesystems call progress() during writes. Progress is important in MapReduce applications, as you will see in later chapters.

FSDataOutputStream

- ✓ The create() method on FileSystem returns an FSDataOutputStream, which, like FSDataInputStream, has a method for querying the current position in the file:
  package org.apache.hadoop.fs;
  public class FSDataOutputStream extends DataOutputStream implements Syncable {
  public long getPos() throws IOException {
  // implementation elided
  }
  // implementation elided
  }
- ✓ However, unlike FSDataInputStream, FSDataOutputStream does not permit seeking. This is because HDFS allows only sequential writes to an open file or appends to an already written file. In other words, there is no support for writing to anywhere other than the end of the file, so there is no value in being able to seek while writing.

Directories

FileSystem provides a method to create a directory:

public boolean mkdirs(Path f) throws IOException

- ✓ This method creates all of the necessary parent directories if they don't already exist, just like the java.io.File's mkdirs() method. It returns true if the directory (and all parent directories) was (were) successfully created.

Often, you don't need to explicitly create a directory, since writing a file, by calling create(), will automatically create any parent directories.

Querying the Filesystem

File metadata: FileStatus

An important feature of any filesystem is the ability to navigate its directory structure and retrieve information about the files and directories that it stores. The FileStatus class encapsulates filesystem metadata for files and directories, including file length, block size, replication, modification time, ownership, and permission information.

Example 3-5. Demonstrating file status information

```
public class ShowFileStatusTest {
private MiniDFSCluster cluster; // use an in-process HDFS cluster for testing
private FileSystem fs;
@Before
public void setUp() throws IOException {
Configuration conf = new Configuration();
if (System.getProperty("test.build.data") == null) {
    System.setProperty("test.build.data", "/tmp");
}
cluster = new MiniDFSCluster(conf, 1, true, null);
fs = cluster.getFileSystem();
OutputStream out = fs.create(new Path("/dir/file"));
out.write("content".getBytes("UTF-8"));
out.close();
}
@After
public void tearDown() throws IOException {
if (fs != null) { fs.close(); }
if (cluster != null) { cluster.shutdown(); }
}
@Test(expected = FileNotFoundException.class)
public void throwsFileNotFoundForNonExistentFile() throws IOException {
fs.getFileStatus(new Path("no-such-file"));
}
@Test
public void fileStatusForFile() throws IOException {
Path file = new Path("/dir/file");
FileStatus stat = fs.getFileStatus(file);
assertThat(stat.getPath().toUri().getPath(), is("/dir/file"));
assertThat(stat.isDir(), is(false));
assertThat(stat.getLen(), is(7L));
assertThat(stat.getModificationTime(),
is(lessThanOrEqualTo(System.currentTimeMillis())));
assertThat(stat.getReplication(), is((short) 1));
assertThat(stat.getBlockSize(), is(64 * 1024 * 1024L));
assertThat(stat.getOwner(), is("tom"));
```

```
assertThat(stat.getGroup(), is("supergroup"));
assertThat(stat.getPermission().toString(), is("rw-r--r--"));
}
@Test
public void fileStatusForDirectory() throws IOException {
Path dir = new Path("/dir");
FileStatus stat = fs.getFileStatus(dir);
assertThat(stat.getPath().toUri().getPath(), is("/dir"));
assertThat(stat.isDir(), is(true));
assertThat(stat.getLen(), is(0L));
assertThat(stat.getModificationTime(),
is(lessThanOrEqualTo(System.currentTimeMillis())));
assertThat(stat.getReplication(), is((short) 0));
assertThat(stat.getBlockSize(), is(0L));
assertThat(stat.getOwner(), is("tom"));
assertThat(stat.getGroup(), is("supergroup"));
assertThat(stat.getPermission().toString(), is("rwxr-xr-x"));
}
    }
```

- ✓ If no file or directory exists, a FileNotFoundException is thrown. However, if you are interested only in the existence of a file or directory, then the exists() method on FileSystem is more convenient:

  public boolean exists(Path f) throws IOException

Listing files

- ✓ Finding information on a single file or directory is useful, but you also often need to be able to list the contents of a directory. That's what FileSystem's listStatus() methods are for:

  public FileStatus[] listStatus(Path f) throws IOException
  public FileStatus[] listStatus(Path f, PathFilter filter) throws IOException
  public FileStatus[] listStatus(Path[] files) throws IOException
  public FileStatus[] listStatus(Path[] files, PathFilter filter) throws IOException
- ✓ When the argument is a file, the simplest variant returns an array of FileStatus objects of length . When the argument is a directory, it returns zero or more FileStatus objects representing the files and directories contained in the directory.
- ✓ Overloaded variants allow a PathFilter to be supplied to restrict the files and directories to matc.
- ✓ Finally, if you specify an array of paths, the result is a shortcut for calling the equivalent single-path listStatus method for each path in turn and accumulating the FileStatus object arrays in a single array. This can be useful for building up lists of input files to process from

distinct parts of the filesystem tree.

Example 3-6 is a simple demonstration of this idea. Note the use of stat2Paths() in FileUtil for turning an array of FileStatus objects to an array of Path objects.

Example 3-6. Showing the file statuses for a collection of paths in a Hadoop filesystem

```
public class ListStatus {

  public static void main(String[] args) throws Exception {
    String uri = args[0];
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(URI.create(uri), conf);

    Path[] paths = new Path[args.length];
    for (int i = 0; i < paths.length; i++) {
      paths[i] = new Path(args[i]);
    }

    FileStatus[] status = fs.listStatus(paths);
    Path[] listedPaths = FileUtil.stat2Paths(status);
    for (Path p : listedPaths) {
      System.out.println(p);
    }
  }
}
```

We can use this program to find the union of directory listings for a collection of paths:

```
% hadoop ListStatus hdfs://localhost/ hdfs://localhost/user/tom
hdfs://localhost/user
hdfs://localhost/user/tom/books
hdfs://localhost/user/tom/quangle.txt
```

File patterns

- ✓ It is a common requirement to process sets of files in a single operation. For example, a MapReduce job for log processing might analyze a month's worth of files contained in a number of directories. Rather than having to enumerate each file and directory to specify the input, it is convenient to use wildcard characters to match multiple files with a single expression, an operation that is known as *globbing*.
- ✓ Hadoop provides two FileSystem method for processing globs:
  public FileStatus[] globStatus(Path pathPattern) throws IOException
  public FileStatus[] globStatus(Path pathPattern, PathFilter filter) throws IOException
- ✓ The globStatus() method returns an array of FileStatus objects whose paths match the supplied pattern, sorted by path. An optional PathFilter can be specified to restrict the matches further.
- ✓ Hadoop supports the same set of glob characters as Unix *bash* (see Table 3-2).

Table 3-2. Glob characters and their meanings

Glob	Name	Matches
*	*asterisk*	Matches zero or more characters
?	*question mark*	Matches a single character
[ab]	*character class*	Matches a single character in the set {a, b}
[^ab]	*negated character class*	Matches a single character that is not in the set {a, b}
[a-b]	*character range*	Matches a single character in the (closed) range [a, b], where a is lexicographically less than or equal to b
[^a-b]	*negated character range*	Matches a single character that is not in the (closed) range [a, b], where a is lexicographically less than or equal to b
{a,b}	*alternation*	Matches either expression a or b
\c	*escaped character*	Matches character c when it is a metacharacter

PathFilter

- ✓ Glob patterns are not always powerful enough to describe a set of files you want to access. For example, it is not generally possible to exclude a particular file using a glob pattern. The listStatus() and globStatus() methods of FileSystem take an optional PathFilter, which allows programmatic control over matching:

```
package org.apache.hadoop.fs;
public interface PathFilter {
boolean accept(Path path);
}
```

PathFilter is the equivalent of java.io.FileFilter for Path objects rather than File objects.

Example 3-7 shows a PathFilter for excluding paths that match a regular expression.

Example 3-7. A PathFilter for excluding paths that match a regular expression

```
public class RegexExcludePathFilter implements PathFilter {

  private final String regex;

  public RegexExcludePathFilter(String regex) {
    this.regex = regex;
  }

  public boolean accept(Path path) {
    return !path.toString().matches(regex);
  }
}
```

- ✓ The filter passes only files that *don't* match the regular expression. We use the filter in conjunction with a glob that picks out an initial set of files to include: the filter is used to refine the results.
- ✓ For example: fs.globStatus(new Path("/2007/*/*"), new RegexExcludeFilter("^.*/2007/12/31$")) will expand to */2007/12/30*.

- ✓ Filters can only act on a file's name, as represented by a Path. They can't use a file's properties, such as creation time, as the basis of the filter. Nevertheless, they can perform matching that neither glob patterns nor regular expressions can achieve.
- ✓ For example, if you store files in a directory structure that is laid out by date (like in the previous section), then you can write a PathFilter to pick out files that fall in a given date range.

Deleting Data

- ✓ Use the delete() method on FileSystem to permanently remove files or directories:

  public boolean delete(Path f, boolean recursive) throws IOException

- ✓ If f is a file or an empty directory, then the value of recursive is ignored. A nonempty directory is only deleted, along with its contents, if recursive is true (otherwise an IOException is thrown).

19. Explain in detail about dataflow of File read & File write

Data Flow

Anatomy of a File Read

To get an idea of how data flows between the client interacting with HDFS, the namenode and the datanodes, consider Figure 3-2, which shows the main sequence of events when reading a file.

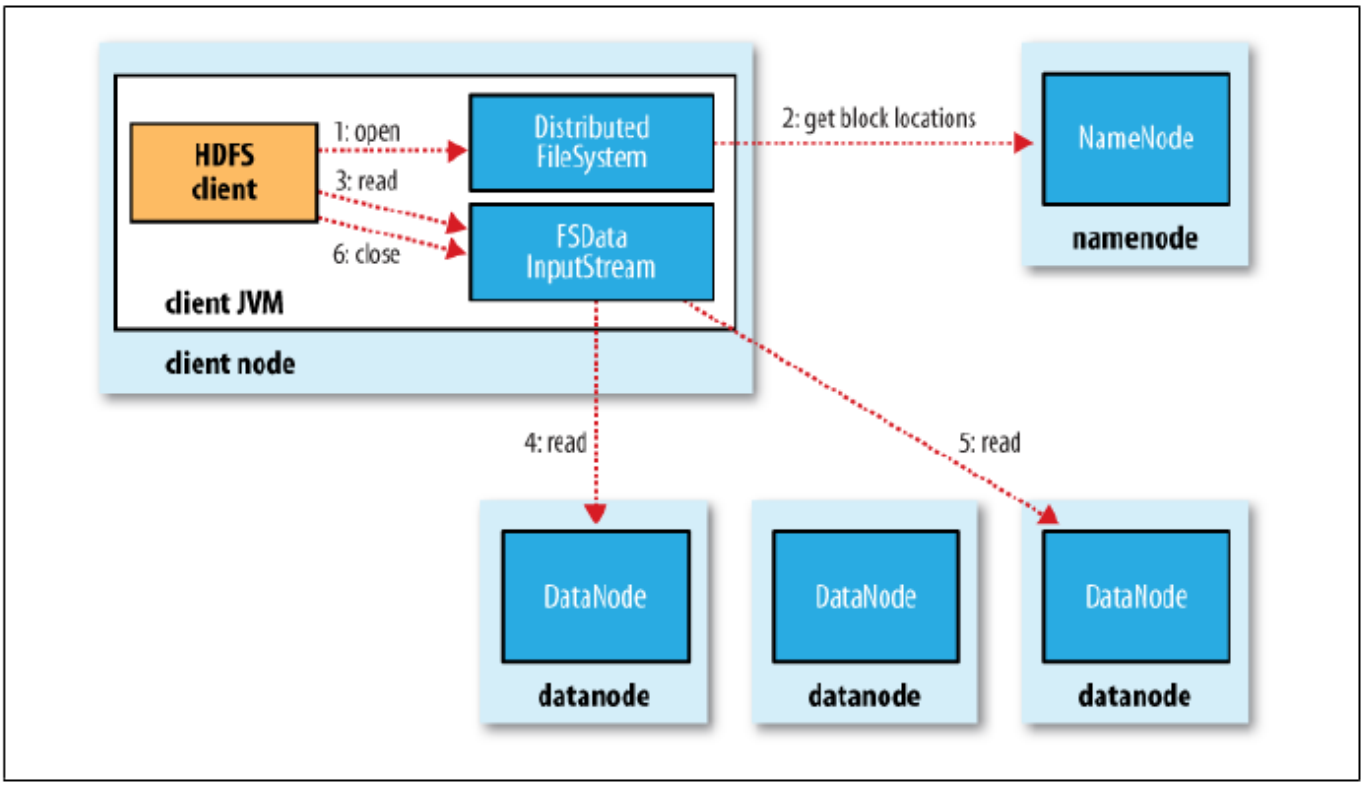


Figure 3-2. A client reading data from HDFS

- ✓ The client opens the file it wishes to read by calling open() on the FileSystem object, which for HDFS is an instance of DistributedFileSystem (step 1 in Figure 3-2).
- ✓ DistributedFileSystem calls the namenode, using RPC, to determine the locations of the blocks for the first few blocks in the file (step 2). For each block, the namenode returns the addresses of the datanodes that have a copy of that block.

- ✓ If the client is itself a datanode (in the case of a MapReduce task, for instance), then it will read from the local datanode, if it hosts a copy of the block (see also Figure 2-2).
- ✓ The DistributedFileSystem returns an FSDataInputStream (an input stream that supports file seeks) to the client for it to read data from. FSDataInputStream in turn wraps a DFSInputStream, which manages the datanode and namenode I/O.
- ✓ The client then calls read() on the stream (step 3). DFSInputStream, which has stored the datanode addresses for the first few blocks in the file, then connects to the first (closest) datanode for the first block in the file.
- ✓ Data is streamed from the datanode back to the client, which calls read() repeatedly on the stream (step 4). When the end of the block is reached, DFSInputStream will close the connection to the datanode, then find the best datanode for the next block (step 5).
- ✓ This happens transparently to the client, which from its point of view is just reading a continuous stream. Blocks are read in order with the DFSInputStream opening new connections to datanodes as the client reads through the stream.
- ✓ It will also call the namenode to retrieve the datanode locations for the next batch of blocks as needed. When the client has finished reading, it calls close() on the FSDataInputStream (step 6). During reading, if the DFSInputStream encounters an error while communicating with a datanode, then it will try the next closest one for that block. It will also remember datanodes that have failed so that it doesn't needlessly retry them for later blocks.
- ✓ The DFSInputStream also verifies checksums for the data transferred to it from the datanode. If a corrupted block is found, it is reported to the namenode before the DFSInput Stream attempts to read a replica of the block from another datanode.
- ✓ One important aspect of this design is that the client contacts datanodes directly to retrieve data and is guided by the namenode to the best datanode for each block. This design allows HDFS to scale to a large number of concurrent clients, since the data traffic is spread across all the datanodes in the cluster. The namenode meanwhile merely has to service block location requests (which it stores in memory, making them very efficient) and does not, for example, serve data, which would quickly become a bottleneck as the number of clients grew.

Anatomy of a File Write

- ✓ Next we'll look at how files are written to HDFS. Although quite detailed, it is instructive to understand the data flow since it clarifies HDFS's coherency model. The case we're going to consider is the case of creating a new file, writing data to it, then closing the file. See Figure 3-4.
- ✓ The client creates the file by calling create() on DistributedFileSystem (step 1 in Figure 3-4). DistributedFileSystem makes an RPC call to the namenode to create a new file in the filesystem's namespace, with no blocks associated with it (step 2).
- ✓ The namenode performs various checks to make sure the file doesn't already exist, and that the client has the right permissions to create the file. If these checks pass, the namenode makes a record of the new file; otherwise, file creation fails and the client is thrown an IOException. The DistributedFileSystem returns an FSDataOutputStream for the client to start writing data to. Just as in the read case, FSDataOutputStream wraps a DFSOutput Stream, which handles communication with the datanodes and namenode.
- ✓ As the client writes data (step 3), DFSOutputStream splits it into packets, which it writes to an internal queue, called the *data queue*. The data queue is consumed by the Data Streamer, whose responsibility it is to ask the namenode to allocate new blocks by picking a list of suitable datanodes to store the replicas.
- ✓ The list of datanodes forms a pipeline—

we'll assume the replication level is three, so **there are three nodes in the pipeline**.

The DataStreamer streams the packets to the first datanode in the pipeline,which stores the packet and forwards it to the second datanode in the pipeline. Similarly,the second datanode stores the packet and forwards it to the third (and last)datanode in the pipeline (step 4).*Figure 3-*

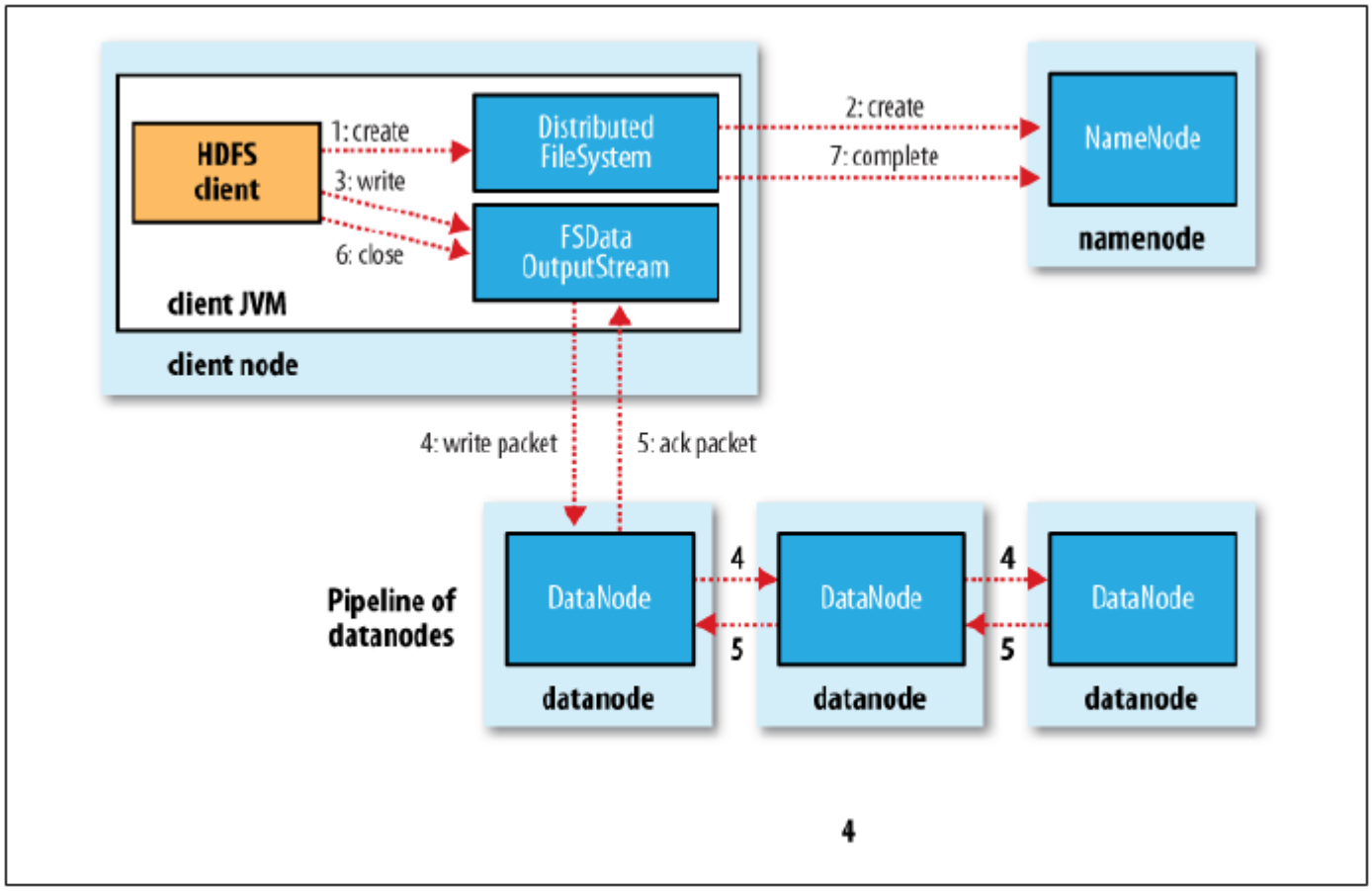


Figure 3-4. A client writing data to HDFS

- DFSOutputStream also maintains an internal queue of packets that are waiting to be acknowledged by datanodes, called the *ack queue*. A packet is removed from the ack queue only when it has been acknowledged by all the datanodes in the pipeline (step 5).
- If a datanode fails while data is being written to it, then the following actions are taken, which are transparent to the client writing the data.
- First the pipeline is closed, and any packets in the ack queue are added to the front of the data queue so that datanodes that are downstream from the failed node will not miss any packets. The current block on the good datanodes is given a new identity, which is communicated to the namenode,
- so that the partial block on the failed datanode will be deleted if the failed datanode recovers later on. The failed datanode is removed from the pipeline and the remainder of the block's data is written to the two good datanodes in the pipeline. The namenode notices that the block is under-replicated, and it arranges for a further replica to be created on another node. Subsequent blocks are then treated as normal.
- It's possible, but unlikely, that multiple datanodes fail while a block is being written. As long as dfs.replication.min replicas (default one) are written, the write will succeed, and the block will be asynchronously replicated across the cluster until its target replication factor is reached (dfs.replication, which defaults to three).
- When the client has finished writing data, it calls close() on the stream (step 6). This action flushes all the remaining packets to the datanode pipeline and waits for acknowledgments before contacting the namenode to signal that the file is complete (step 7). The namenode already knows which blocks the file is made up of (via Data Streamer asking for block allocations), so it only has to wait for blocks to be minimally replicated before returning successfully.

IMPORTANT QUESTIONS

PART-A

1. Define Globus Toolkit: Grid Computing Middleware
2. What Is the Globus Toolkit?
3. List the resources of The Grid Resource Information Service (GRIS)
4. What is a Job Manager ?
5. What is Globus Resource Allocation Manager (GRAM)
6. Define Namenodes and Datanodes
7. Define HDFS Federation
8. Define Directories in file system.
9. List out the GT Components are grouped into five main functional areas:
10. Define Security components
11. Define Delegation service
12. Define Reliable File Transfer (RFT)
13. What is Execution management
14. Define Open source grid middleware
15. Define Hadoop Framework
16. What is Mapreduce
17. Give the Design of Hadoop file system
18. Give the dataflow of File read & File write.

PART-B

1. Explain in detail about the Open source grid middleware packages.
2. Explain in detail about the Globus Toolkit (GT4) Architecture.
3. Give the Configuration of Globus Toolkit (GT4) in detail.
4. Explain Usageof Globus in detail.
5. Explain Main components and Programming model in Globus Toolkit (GT4) in detail
6. Explain in detail about the Hadoop Framework with neat sketch.
7. Give the Mapreduce, Input splitting and explain in detail
8. Briefly explain about the Design of Hadoop file system.
9. Give the Overview of HDFS concepts.
10. Explain in detail about the dataflow of File read & File write.

UNIT V
SECURITY

Trust models for Grid security environment - Authentication and Authorization methods Grid security infrastructure - Cloud Infrastructure security: network, host and application level - aspects of data security, provider data and its security, Identity and access management architecture, IAM practices in the cloud, SaaS, PaaS, IaaS availability in the cloud, Key privacy issues in the cloud.

PART A

1. Trust worthiness is usually calculated in Reputation-Based Trust Model?

In a reputation-based model, jobs are sent to a resource site only when the site is trustworthy to meet users' demands. The site trustworthiness is usually calculated from the following information: the defense capability, direct reputation, and recommendation trust.

The defense capability refers to the site's ability to protect itself from danger. It is assessed according to such factors as intrusion detection, firewall, response capabilities and anti-virus capacity

2. What is the Authorization for Access Control?

The authorization is a process to exercise access control of shared resources. Decisions can be made either at the access point of service or at a centralized place. Typically, the resource is a host that provides processors and storage for services deployed on it.

Based on a set predefined policies or rules, the resource may enforce access for local services. The central authority is a special entity which is capable of issuing and revoking polices of access rights granted to remote accesses.

3. How can we classify the authority?

The authority can be classified into three categories: attribute authorities, policy authorities, and identity authorities. Attribute authorities issue attribute assertions; policy authorities issue authorization policies; identity authorities issue certificates. The authorization server makes the final authorization decision.

4. List the Three Authorization Models?

The **subject-push model** is shown at the top diagram. The user conducts handshake with the authority first and then with the resource site in a sequence.

The **resource-pulling model** puts the resource in the middle. The user checks the resource first. Then the resource contacts its authority to verify the request, and the authority authorizes at step finally the resource accepts or rejects the request from the subject at step

The **authorization agent model** puts the authority in the middle. The subject check with the authority at step 1 and the authority makes decisions on the access of the requested resources. The authorization process is complete at steps 3 and 4 in the reverse direction

5. List the four distinct functions?

GSI may be thought of as being composed of ***four distinct functions***:

- Message protection,

- Authentication
- Delegation
- Authorization

6. What is Transport-Level Security?

Transport-level security entails SOAP messages conveyed over a network connection protected by TLS.

TLS provides for both integrity protection and privacy (via encryption). Transport-level security is normally used in conjunction with X.509 credentials for authentication, but can also be used without such credentials to provide message protection without authentication, often referred to as ―anonymous transport-level security.‖

In this mode of operation, authentication may be done by username and password in a SOAP message

7. What are the message-level securities for message protection for SOAP messages provided by GSI?

GSI also provides message-level security for message protection for SOAP messages by implementing the ***WS-Security standard*** and the ***WS-Secure Conversation specification***. The WS-Security standard from OASIS defines a framework for applying security to individual SOAP messages;WS-Secure Conversation is a proposed standard from IBM and Microsoft that allows for an initial exchange of messages to establish a security context which can then be used to protect subsequent messages in a manner that requires less computational overhead

8. What are the four primary pieces of information the certificate includes in GSI authentication?

As a central concept in GSI authentication, a certificate includes four primary pieces of information

(1) A subject name, which identifies the person or object that the certificate represents

(2) The public key belonging to the subject

(3) The identity of a CA that has signed the certificate to certify that the public key and the identity both belong to the subject

(4) The digital signature of the named CA. X.509 provides each entity with a unique identifier and a method to assert that identifier to another party through the use of an asymmetric key pair bound to the identifier by the certificate

9. What are the four significant risk factors Network Level infrastructure security? There are *four significant risk factors* they are

1) Ensuring the confidentiality and integrity
2) Ensuring proper access control
3) Ensuring the availability of the Internet-facing resources
4) Replacing the established model of network zones and tiers with domains

10. What is prefix hijacking?

Prefix hijacking involves announcing an autonomous system address space that belongs to someone else without her permission. Such announcements often occur because of a

configuration mistake, but that misconfiguration may still affect the availability of your cloud-based resources

11. What does Denial-of-Service Attack (DoS) mean?

A denial-of-service (DoS) is any type of attack where the attackers (hackers) attempt to prevent legitimate users from accessing the service. In a DoS attack, the attacker usually sends excessive messages asking the network or server to authenticate requests that have invalid return addresses. The network or server will not be able to find the return address of the attacker when sending the authentication approval, causing the server to wait before closing the connection. When the server closes the connection, the attacker sends more authentication messages with invalid return addresses. Hence, the process of authentication and server wait will begin again, keeping the network or server busy

12. What are the basic types of DoS attack ?

A DoS attack can be done in a several ways. The basic types of DoS attack include:

1. Flooding the network to prevent legitimate network traffic
2. Disrupting the connections between two machines, thus preventing access to a service
3. Preventing a particular individual from accessing a service.
4. Disrupting a service to a specific system or individual
5. Disrupting the state of information, such resetting of TCP sessions

13. What are the problems caused by the DOS attacks?

DoS attacks can cause the following problems:

1. Ineffective services
2. Inaccessible services
3. Interruption of network traffic
4. Connection interference

14. What is Distributed Denial-of –service (DDoS)?

A distributed denial-of-service (DDoS) attack is one in which a multitude of compromised systems attack a single target, thereby causing denial of service for users of the targeted system. The flood of incoming messages to the target system essentially forces it to shut down, thereby denying service to the system to legitimate users.

15. Lists security controls at the network level?

Threat outlook	**Low (with the exception of DoS attacks)**
Preventive controls	Network access control supplied by provider (e.g., firewall), encryption of data in transit (e.g., SSL, IPSec)
Detective controls	Provider-managed aggregation of security event logs (security incident and event management, or SIEM), network-based intrusion detection system/intrusion prevention system (IDS/IPS)

16. What Is Privacy?

- ✓ The concept of privacy varies widely among (and sometimes within) countries, cultures, and jurisdictions. It is shaped by public expectations and legal interpretations; as such, a concise definition is elusive if not impossible.
- ✓ Privacy rights or obligations are related to the collection, use, disclosure, storage, and destruction of personal data (or personally identifiable information—PII).
- ✓ Privacy is about the accountability of organizations to data subjects, as well as the transparency to an organization's practice around personal information.

17. What are the Enterprise IAM requirement?

Enterprise IAM requirements include:

- Provisioning of cloud service accounts to users, including administrators.
- Provisioning of cloud services for service-to-service integration
- SSO support for users based on federation standards (e.g., SAML support).
- Support for internal- and regulatory-policy compliance requirements, including segregation of duties using RBAC, rules, or claims-based authentication methodology.

18. List the user management function in the cloud?

User management functions in the cloud can be categorized as follows:

- Cloud identity administration
- Federation or SSO
- Authorization management
- Compliance management

19. List the new host security threats in public IaaS?

Some of the new host security threats in the public IaaS include:

- Stealing keys used to access and manage hosts (e.g., SSH private keys)
- Attacking unpatched, vulnerable services listening on standard ports (e.g., FTP, NetBIOS, SSH)
- Hijacking accounts that are not properly secured (i.e., weak or no passwords for standard accounts)
- Attacking systems that are not properly secured by host firewalls
- Deploying Trojans embedded in the software component in the VM or within the VM image (the OS) itself

20. What is Clearing?

Clearing is the process of eradicating the data on media before reusing the media in an environment that provides an acceptable level of protection for the data that was on the media before clearing. All internal memory, buffer, or other reusable memory shall be cleared to effectively deny access to previously stored information.

21.* What is mean by Sanitization?***

Sanitization is the process of removing the data from media before reusing the media in an environment that does not provide an acceptable level of protection for the data that was on the media before sanitizing. IS resources shall be sanitized before they are released from classified information controls or released for use at a lower classification level.

22. What Are the Key Privacy Concerns in the Cloud?and give the concept in detail

Privacy advocates have raised many concerns about cloud computing. These concerns typically mix security and privacy. Here are some additional considerations to be aware of:

- ✓ ***Access***
- ✓ ***Compliance***
- ✓ ***Storage***
- ✓ ***Retention***
- ✓ ***Destruction***
- ✓ ***Audit and monitoring***
- ✓ ***Privacy breaches***

23.What Is the Data Life Cycle?

✓ Personal information should be managed as part of the data used by the organization. It should be managed from the time the information is conceived through to its final disposition.

✓ Protection of personal information should consider the impact of the cloud on each of the following phases as detailed in Figure 7-1.

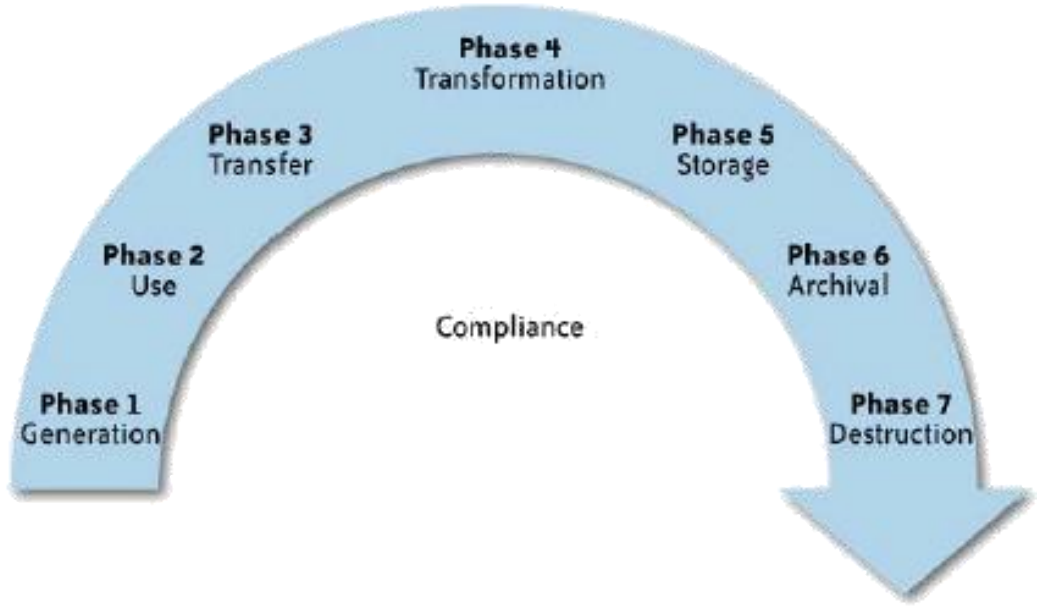


FIGURE 7-1. KPMG data life cycle

24.Define Cloud Identity Administration

Cloud identity administrative functions should focus on life cycle management of user identities in the cloud—provisioning, deprovisioning, identity federation, SSO, password or credentials management, profile management, and administrative management.Organizations that are not capable of supporting federation should explore cloud-based identity management services.

25.List the four key components of the IAM automation process:

• User Management, New Users
• User Management, User Modifications
• Authentication Management
• Authorization Management

26.Give The IAM processes to support the business can be broadly categorized as follows:

User management

Activities for the effective governance and management of identity life cycles

Authentication management

Activities for the effective governance and management of the process for determining that an entity is who or what it claims to be

Authorization management

Activities for the effective governance and management of the process for determining entitlement rights that decide what resources an entity is permitted to access in accordance with the organization's policies

Access management

Enforcement of policies for access control in response to a request from an entity (user, services) wanting to access an IT resource within the organization

Data management and provisioning

Propagation of identity and data for authorization to IT resources via automated or manual processes

Monitoring and auditing

Monitoring, auditing, and reporting compliance by users regarding access to resources within the organization based on the defined policies

27.Define Clearing

Clearing is the process of eradicating the data on media before reusing the media in an environment that provides an acceptable level of protection for the data that was on the media before clearing. All internal memory, buffer, or other reusable memory shall be cleared to effectively deny access to previously stored information.

28.Define Sanitization

Sanitization is the process of removing the data from media before reusing the media in an environment that does not provide an acceptable level of protection for the data that was on the media before sanitizing. IS resources shall be sanitized before they are released from classified information controls or released for use at a lower classification level.

29. How do you prove data provenance in a cloud computing scenario when you are using shared resources?

Those resources are not under your physical or even logical control, and you probably have no ability to track the systems used or their state at the times you used them—even if you know some identifying information about the systems (e.g., their IP addresses) and the ―general‖ location (e.g., a country, and not even a specific data center).

A final aspect of data security is *data remanence*. ―Data remanence is the residual representation of data that has been in some way nominally erased or removed. This residue may

be due to data being left intact by a nominal delete operation, or through physical properties of the storage medium.

Data remanence may make inadvertent disclosure of sensitive information possible, should the storage media be released into an uncontrolled environment (e.g., thrown in the trash, or given to a third party).‖

The risk posed by data remanence in cloud services is that an organization's data can be inadvertently exposed to an unauthorized party—regardless of which cloud service you are using (SaaS, PaaS, or IaaS). When using SaaS or PaaS, the risk is almost certainly unintentional or inadvertent exposure.

30.GSI may be thought of as being composed of *four distinct functions* what are they

- Message protection,
- Authentication
- Delegation
- Authorization

PART B

1. Explain in detail the Trust Models for Grid Security Enforcement?

Many potential security issues may occur in a grid environment if qualified security mechanisms are not in place. These issues include network sniffers, out-of-control access, faulty operation, malicious operation, integration of local security mechanisms, delegation, dynamic resources and services, attack provenance, and so on.

Computational grids are motivated by the desire to share processing resources among many organizations to solve large-scale problems. Indeed, grid sites may exhibit unacceptable security conditions and system vulnerabilities

On the one hand, a user job demands the resource site to provide security assurance by issuing a security demand (SD). On the other hand, the site needs to reveal its trustworthiness, called its trust index (TI). These two parameters must satisfy a security-assurance condition: TI ≥ SD during the job mapping process. When determining its security demand, users usually care about some typical attributes. These attributes and their values are dynamically changing and depend heavily on the trust model, security policy, accumulated reputation, self-defense capability, attack history, and site vulnerability

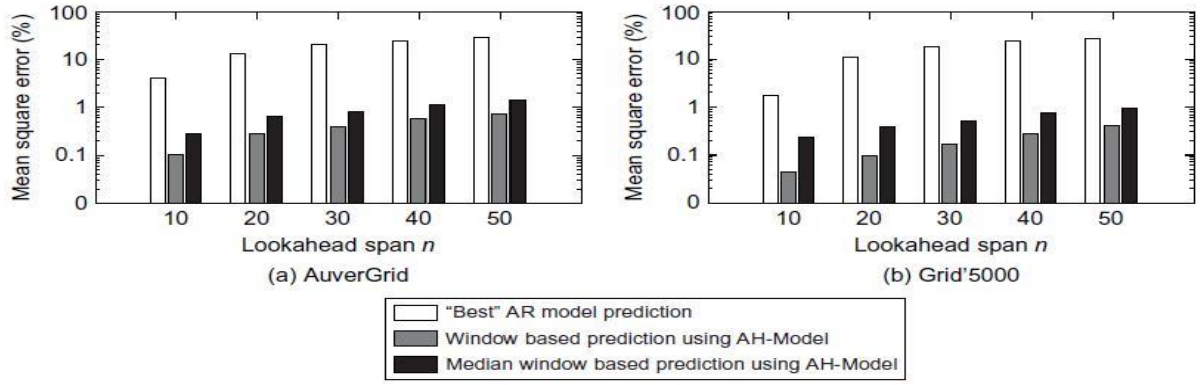


FIGURE 7.28 Mean square errors of two implementations of AH-Model, compared with the best MSE by using the AR model.

Three challenges are outlined below to establish the trust among grid sites.

First challenge is integration with existing systems and technologies. The resources sites in a grid are usually heterogeneous and autonomous. It is unrealistic to expect that a single type of security can be compatible with and adopted by every hosting environment.

At the same time, existing security infrastructure on the sites cannot be replaced overnight. Thus, to be successful, grid security architecture needs to step up to the challenge of integrating with existing security architecture and models across platforms and hosting environments.

Second challenge is interoperability with different ―hosting environments.‖ Services are often invoked across multiple domains, and need to be able to interact with one another.

The interoperation is demanded at the protocol, policy, and identity levels. For all these levels, interoperation must be protected securely.

Third challenge is to construct trust relationships among interacting hosting environments. Grid service requests can be handled by combining resources on multiple security domains. Trust relationships are required by these domains during the end-to-end traversals. A service needs to be open to friendly and interested entities so that they can submit requests and access securely.

Resource sharing among entities is one of the major goals of grid computing. A trust relationship must be established before the entities in the grid interoperate with one another. The entities have to choose other entities that can meet the requirements of trust to coordinate with.

The entities that submit requests should believe the resource providers will try to process their requests and return the results with a specified QoS.

To create the ***proper trust relationship between grid entities, two kinds of trust models are often used*. One** is the PKI-based model, which mainly exploits the PKI to authenticate and authorize entities; we will discuss this in the next section. The **second** is the reputation-based model. The grid aims to construct a large-scale network computing system by integrating distributed, heterogeneous, and autonomous resources.

2. Write short notes on Generalized Trust Model?

Figure 7.29 shows a general trust model. At the bottom, we identify three major factors which influence the trustworthiness of a resource site. An inference module is required to aggregate these factors. Followings are some existing inference or aggregation methods. An intra-site fuzzy inference procedure is called to assess defense capability and direct reputation. Defense capability is decided by the firewall, intrusion detection system (IDS), intrusion response capability, and anti-virus capacity of the individual resource site.

Direct reputation is decided based on the job success rate, site utilization, job turnaround time, and job slowdown ratio measured. Recommended trust is also known as secondary trust and is obtained indirectly over the grid network.

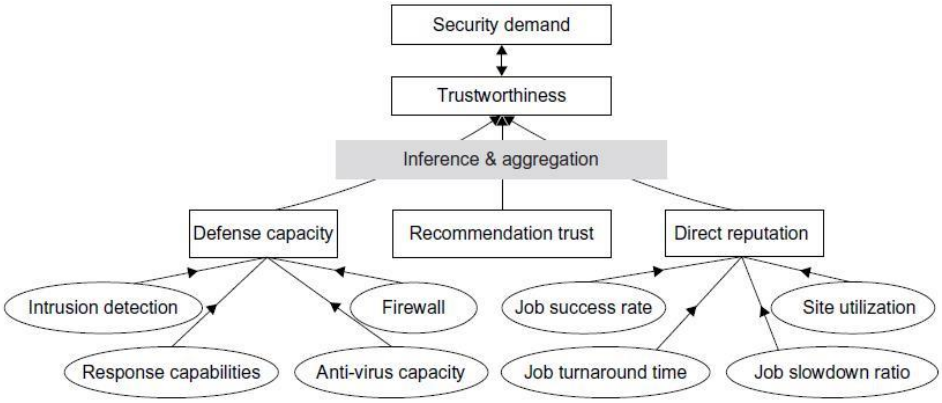


FIGURE 7.29 A general trust model for grid computing.

3. Write short notes on Reputation-Based Trust Model ?

In a reputation-based model, jobs are sent to a resource site only when the site is trustworthy to meet users' demands. The site trustworthiness is usually calculated from the following information: the defense capability, direct reputation, and recommendation trust.

The defense capability refers to the site's ability to protect itself from danger. It is assessed according to such factors as intrusion detection, firewall, response capabilities, anti-virus capacity, and

Direct reputation is based on experiences of prior jobs previously submitted to the site. The reputation is measured by many factors such as prior job execution success rate, cumulative site utilization, job turnaround time, job slowdown ratio, and so on.

A positive experience associated with a site will improve its reputation. On the contrary, a negative experience with a site will decrease its reputation.

4. Write short notes on a Fuzzy-Trust Model ?

The trust index (TI) of a resource site is aggregated through the fuzzy-logic inference process over all related parameters. Specifically, one can use a two-level fuzzy logic to estimate the aggregation of numerous trust parameters and security attributes into scalar quantities that are easy to use in the job scheduling and resource mapping process.

The TI is normalized as a single real number with 0 representing the condition with the highest risk at a site and 1 representing the condition which is totally risk-free or fully trusted. The fuzzy inference is accomplished through four steps:

- ✓ Fuzzification ,
- ✓ Inference ,
- ✓ Aggregation , and
- ✓ Defuzzification

The second salient feature of the trust model is that if a site's trust index cannot match the job security demand (i.e., SD > TI), the trust model could deduce detailed security features to guide the site security upgrade as a result of tuning the fuzzy system.

5. Explain in detail the Authentication and Authorization Methods?

The major authentication methods in the grid include passwords, PKI, and Kerberos. The password is the simplest method to identify users, but the most vulnerable one to use. The PKI is the most popular method supported by GSI.

To implement PKI, we use a trusted third party, called the certificate authority (CA). Each user applies a unique pair of public and private keys. The public keys are issued by the CA by issuing a certificate, after recognizing a legitimate user.

The private key is exclusive for each user to use, and is unknown to any other users. A digital certificate in IEEE X.509 format consists of the user name, user public key, CA name, and a secrete signature of the user.

The following example illustrates the use of a PKI service in a grid environment.

Example 7.12 Trust Delegation Using the Proxy Credential in GSI

The PKI is not strong enough for user authentication in a grid. Figure 7.30 shows a scenario where a sequence of trust delegation is necessary. Bob and Charlie both trust Alice, but Charlie does not trust Bob. Now, Alice submits a task Z to Bob.

The task Z demands many resources for Bob to use, independently. Bob forwards a subtask Y of Z to Charlie. Because Charlie does not trust Bob and is not sure whether Y is really originally requested by Alice, the subtask Y from Bob is rejected for resources by Charlie.

For Charlie to accept the subtask Y, Bob needs to show Charlie some proof of entrust from Alice.A proxy credential is the solution proposed by GSI. A proxy credential is a temporary certificate generated by a user.

Two benefits are seen by using proxy credentials.

First, the proxy credential is used by its holder to act on behalf of the original user or the delegating party. A user can temporarily delegate his right to a proxy.

Second, single sign-on can be achieved with a sequence of credentials passed along the trust chain. The delegating party (Alice) need not verify the remote intermediate parties in a trust chain.

The only difference between the proxy credential and a digital certificate is that the proxy credential is not signed by a CA. We need to known the relationship among the certificates of the CA and Alice, and proxy credential of Alice. The CA certificate is signed

- First with its own private key.
- Second, the certificate Alice holds is signed with the private key of the CA.
- Finally, the proxy credential sent to her proxy (Bob) is signed with her private key.

.

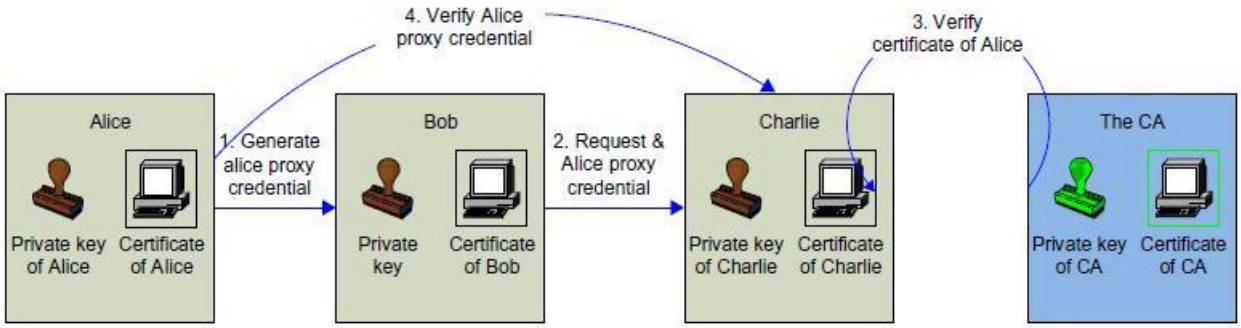


FIGURE 7.30 Interactions among multiple parties in a sequence of trust delegation operations using the PKI services in a GT4-enabled grid environment

The procedure delegates the rights of Alice to Bob by using the proxy credential

First, the generation of the proxy credential is similar to the procedure of generating a user certificate in the traditional PKI.
Second, when Bob acts on behalf of Alice, he sends the request together with Alice's proxy credential and the Alice certificate to Charlie.
Third, after obtaining the proxy credential, Charlie finds out that the proxy credential is signed by Alice. So he tries to verify the identity of Alice and finds Alice trustable.
Finally, Charlie accepts Bob's requests on behalf of Alice. This is called a trust delegation chain.

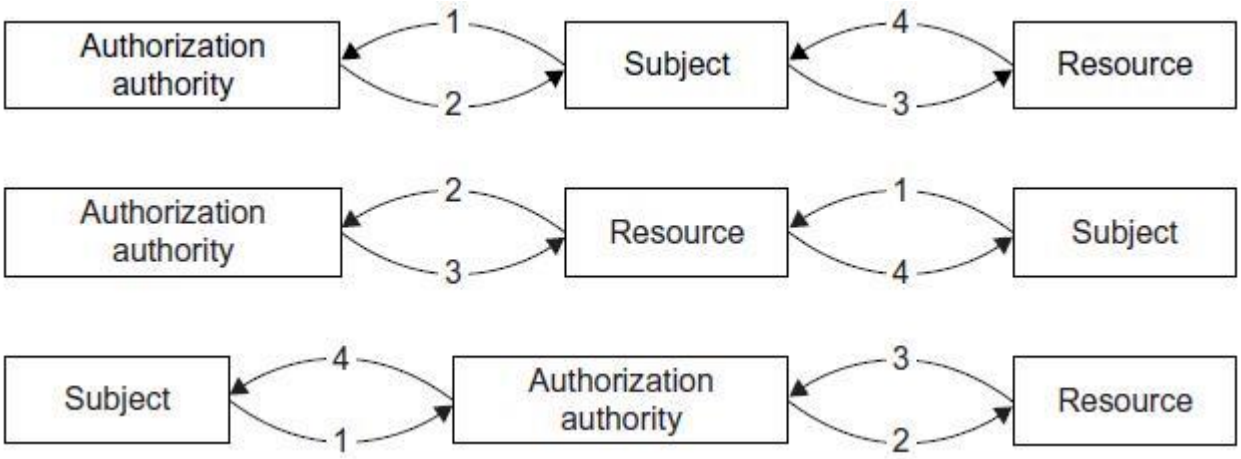


FIGURE 7.31 Three authorization models: the subject-push model, resource-pulling model, and the authorization agent model.

1. **Authorization for Access Control**
   - ✓ The authorization is a process to exercise access control of shared resources. Decisions can be made either at the access point of service or at a centralized place. Typically, the resource is a host that provides processors and storage for services deployed on it. Based on a set predefined policies or rules, the resource may enforce access for local services.
   - ✓ The central authority is a special entity which is capable of issuing and revoking polices of access rights granted to remote accesses. The authority can be classified into three categories: attribute authorities, policy authorities, and identity authorities.
   - ✓ Attribute authorities issue attribute assertions; policy authorities issue authorization policies; identity authorities issue certificates. The authorization server makes the final authorization decision.
2. **Three Authorization Models**
   - ✓ Figure 7.31 shows three authorization models The subject is the user and the resource refers to the machine side.

The **subject-push model** is shown at the top diagram. The user conducts handshake with the authority first and then with the resource site in a sequence.
The **resource-pulling model** puts the resource in the middle. The user checks the resource first. Then the resource contacts its authority to verify the request, and the authority authorizes at step finally the resource accepts or rejects the request from the subject at step
The **authorization agent model** puts the authority in the middle. The subject check with the authority at step 1 and the authority makes decisions on the access of the requested resources. The authorization process is complete at steps 3 and 4 in the reverse direction.

Grid Security Infrastructure (GSI) :

✓ Although the grid is increasingly deployed as a common approach to constructing dynamic, inter domain, distributed computing and data collaborations, ―lack of security/trust between different services‖ is still an important challenge of the grid.

The grid requires a security infrastructure with the following properties:

- Easy to use;
- Conforms with the VO's security needs while working well with site policies of each resource provider site
- Provides appropriate authentication and encryption of all interactions.

GSI is an important step toward satisfying these requirements. As a well-known security solution in the grid environment, GSI is a portion of the Globus Toolkit and provides fundamental security services needed to support grids, including supporting for message protection, authentication and delegation, and authorization.

GSI enables secure authentication and communication over an open network, and permits mutual authentication across and among distributed sites with single sign-on capability. No centrally managed security system is required, and the grid maintains the integrity of its members' local policies.

GSI supports both message-level security, which supports the WS-Security standard and the WS-Secure Conversation specification to provide message protection for SOAP messages, and transport-level security, which means authentication via TLS with support for X.509 proxy certificates.

6. Explain the GSI Functional Layers with neat diagram?

GT4 provides distinct WS and pre-WS authentication and authorization capabilities. Both build on the same base, namely the X.509 standard and entity certificates and proxy certificates, which are used to identify persistent entities such as users and servers and to support the temporary delegation of privileges to other entities, respectively.

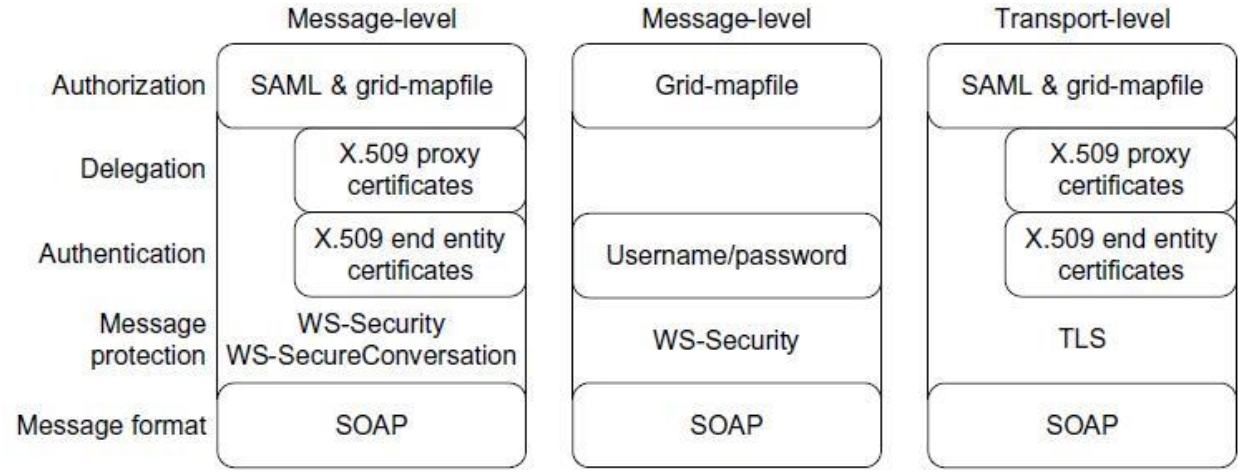


FIGURE 7.32 GSI functional layers at the message and transport levels.

As shown in Figure 7.32, GSI may be thought of as being composed of ***four distinct functions***:

- Message protection,
- Authentication
- Delegation
- Authorization

TLS (transport-level security) or WS-Security and WS-Secure Conversation (message-level) are used as message protection mechanisms in combination with SOAP. X.509 End Entity Certificates or Username and Password are used as authentication credentials. X.509 Proxy Certificates and WS-Trust are used for delegation. An Authorization Framework allows for a variety of authorization schemes, including a ―grid-mapfile‖ ACL, an ACL defined by a service, a custom authorization handler, and access to an authorization service via the SAML protocol.

Associated security tools provide for the storage of X.509 credentials (MyProxy and Delegation services), the mapping between GSI and other authentication mechanisms (e.g., KX509 and PKINIT for Kerberos, MyProxy for one-time passwords), and maintenance of information used for authorization (VOMS, GUMS, PERMIS).

The web services portions of GT4 use SOAP as their message protocol for communication. Message protection can be provided either by transport-level security, which transports SOAP messages over TLS, or by message-level security, which is signing and/or encrypting portions of the SOAP message using the WS-Security standard. Here we describe these two methods.

1. Transport-Level Security

Transport-level security entails SOAP messages conveyed over a network connection protected by TLS.

TLS provides for both integrity protection and privacy (via encryption). Transport-level security is normally used in conjunction with X.509 credentials for authentication, but can also be used without such credentials to provide message protection without authentication, often referred to as ―anonymous transport-level security.‖ In this mode of operation, authentication may be done by username and password in a SOAP message.

2. Message-Level Security

GSI also provides message-level security for message protection for SOAP messages by implementing

The ***WS-Security standard and the WS-Secure Conversation*** specification.

The ***WS-Security standard*** from OASIS defines a framework for applying security to individual SOAP messages;

WS-Secure Conversation is a proposed standard from IBM and Microsoft that allows for an initial exchange of messages to establish a security context which can then be used to protect subsequent messages in a manner that requires less computational overhead (i.e., it allows the trade-off of initial overhead for setting up the session for lower overhead for messages).

GSI uses these mechanisms to provide security on a per-message basis, that is, to an individual message without any preexisting context between the sender and receiver (outside of sharing some set of trust roots).

GSI, as described further in the subsequent section on authentication, allows for both X.509 public key credentials and the combination of username and password for authentication; however, differences still exist.

With username/password, only the WS-Security standard can be used to allow for authentication; that is, a receiver can verify the identity of the communication initiator.

GSI allows three additional protection mechanisms. The first is integrity protection, by which a receiver can verify that messages were not altered in transit from the sender. The second is encryption, by which messages can be protected to provide confidentiality.

The third is replay prevention, by which a receiver can verify that it has not received the same message previously. These protections are provided between WS-Security and WS-Secure Conversation. The former applies the keys associated with the sender and receiver's X.509 credentials. The X.509 credentials are used to establish a session key that is used to provide the message protection.

3. **Authentication and Delegation**

GSI has traditionally supported authentication and delegation through the use of X.509 certificates and public keys. As a new feature in GT4, GSI also supports authentication through plain usernames and passwords as a deployment option.. GSI uses X.509 certificates to identify persistent users and services.

As a central concept in GSI authentication, a certificate includes four primary pieces of information: a subject name, which identifies the person or object that the certificate represents; the public key belonging to the subject; the identity of a CA that has signed the certificate to certifythat the public key and the identity both belong to the subject; and the digital signature of the named CA. X.509 provides each entity with a unique identifier (i.e., a distinguished name) and a method to assert that identifier to another party through the use of an asymmetric key pair bound to the identifier by the certificate.

The X.509 certificate used by GSI are conformant to the relevant standards and conventions. Grid deployments around the world have established their own CAs based on third-party software to issue the X.509 certificate for use with GSI and the Globus Toolkit.

GSI also supports delegation and single sign-on through the use of standard X.509 proxy certificates.Proxy certificates allow bearers of X.509 to delegate their privileges temporarily to another entity. For the purposes of authentication and authorization, GSI treats certificates and proxy certificates equivalently. Authentication with X.509 credentials can be accomplished either via TLS, in the case of transport-level security, or via signature as specified by WS-Security, in the case of message-level security.

Example 7.13 Mutual Authentication between Two Parties Mutual authentication is a process by which two parties with certificates signed by the CA prove to each other that they are who they say they are based on the certificate and the trust of the CAs that signed each other's certificates.

GSI uses the Secure Sockets Layer (SSL) for its mutual authentication protocol. To mutually authenticate, the first person (Alice) establishes a connection to the second person (Bob) to start the authentication process.

Alice gives Bob her certificate. The certificate tells Bob who Alice is claiming to be (the identity), what Alice's public key is, and what CA is being used to certify the certificate. Bob will first make sure the certificate is valid by checking the CA's digital signature to make sure the CA actually signed the certificate and the certificate hasn't been tampered with.

Once Bob has checked out Alice's certificate, Bob must make sure Alice really is the person identified in the certificate. Bob generates a random message and sends it to Alice, asking Alice to encrypt it.

Alice encrypts the message using her private key, and sends it back to Bob. Bob decrypts the message using Alice's public key.

If this results in the original random message, Bob knows Alice is who she says she is. Now that Bob trusts Alice's identity, the same operation must happen in reverse.

Bob sends Alice his certificate, and Alice validates the certificate and sends a challenge message to be encrypted. Bob encrypts the message and sends it back to Alice, and Alice decrypts it and compares it with the original. If it matches, Alice knows Bob is who he says he is

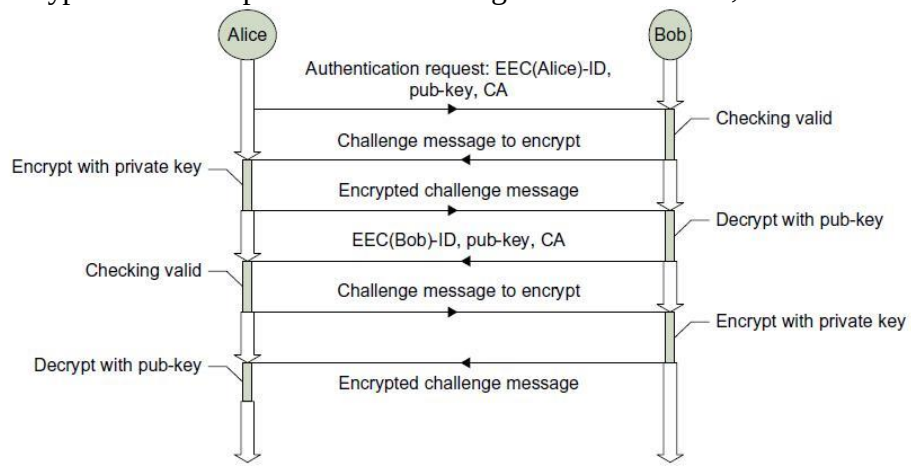


FIGURE 7.33 multiple handshaking in a mutual authentication scheme

4. Trust Delegation

To reduce or even avoid the number of times the user must enter his passphrase when several grids are used or have agents (local or remote) requesting services on behalf of a user, GSI provides a delegation capability and a delegation service that provides an interface to allow clients to delegate (and renew) X.509 proxy certificates to a service.

The interface to this service is based on the WS-Trust specification.

A proxy consists of a new certificate and a private key. The key pair that is used for the proxy, that is, the public key embedded in the certificate and the private key, may either be regenerated for each proxy or be obtained by other means.

The new certificate contains the owner's identity, modified slightly to indicate that it is a proxy. The new certificate is signed by the owner, rather than a CA (see Figure 7.34).

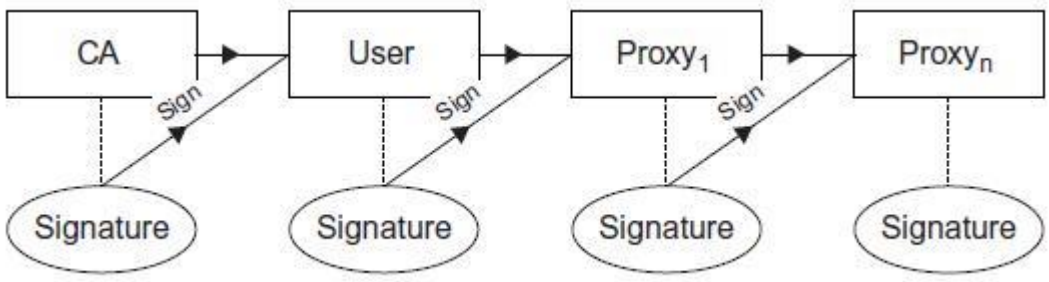


FIGURE 7.34 A sequence of trust delegations in which new certificates are signed by the owners rather by the CA.

The certificate also includes a time notation after which the proxy should no longer be accepted by others. Proxies have limited lifetimes. Because the proxy isn't valid for very long, it

doesn't have to stay quite as secure as the owner's private key, and thus it is possible to store the proxy's private key in a local storage system without being encrypted, as long as the permissions on the file prevent anyone else from looking at them easily.

Once a proxy is created and stored, the user can use the proxy certificate and private key for mutual authentication without entering a password. When proxies are used, the mutual authentication process differs slightly. The remote party receives not only the proxy's certificate (signed by the owner), but also the owner's certificate. During mutual authentication, the owner's public key (obtained from her certificate) is used to validate the signature on the proxy certificate.

The CA's public key is then used to validate the signature on the owner's certificate. This establishes a chain of trust from the CA to the last proxy through the successive owners of resources. The GSI uses WS-Security with textual usernames and passwords.

This mechanism supports more rudimentary web service applications. When using usernames and passwords as opposed to X.509 credentials, the GSI provides authentication, but no advanced security features such as delegation, confidentiality, integrity, and replay prevention. However, one can use usernames and passwords with anonymous transport-level security such as unauthenticated TLS to ensure privacy

Infrastructure Security IaaS application providers treat the applications within the customer virtual instance as a black box and therefore are completely indifferent to the operations and management of a applications of the customer . The entire pack (customer application and run time application) is run on the customers' server on provider infrastructure and is managed by customers themselves. it is important to note that the customer must take full responsibility for securing their cloud deployed applications

- ✓ Cloud deployed applications must be designed for the internet threat model.
- ✓ They must be designed with standard security countermeasures to guard against the common web vulnerabilities.
- ✓ Customers are responsible for keeping their applications up to date - and must therefore ensure they have a patch strategy to ensure their applications are screened from malware and hackers scanning for vulnerabilities to gain unauthorized access to their data within the cloud.
- ✓ Customers should not be tempted to use custom implementations of Authentication, Authorization and Accounting as these can become weak if not properly implemented.
- ✓ The foundational infrastructure for a cloud must be inherently secure whether it is a private or public cloud or whether the service is SAAS, PAAS or IAAS.

Inherent component-level security:

The cloud needs to be architected to be secure, built with inherently secure components, deployed and provisioned securely with strong interfaces to other components and supported securely, with vulnerability-assessment and change-management processes that produce management information and service-level assurances that build trust.

Stronger interface security:

The points in the system where interaction takes place (user-to-network, server-to application) require stronger security policies and controls that ensure consistency and accountability.

Resource lifecycle management:

The economics of cloud computing are based on multi-tenancy and the sharing of resources. As the needs of the customers and requirements will change, a service provider must provision and

decommission correspondingly those resources - bandwidth, servers, storage and security. This lifecycle process must be managed in order to build trust.

7. Explain in detail Infrastructure Security and its types?

The Network Level infrastructure security:

When looking at the network level of infrastructure security, it is important to distinguish between public clouds and private clouds, With private clouds, there are no new attacks, vulnerabilities, or changes in risk *specific to this topology* that information security personnel need to consider.

Although your organization's IT architecture may change with the implementation of a private cloud, your current network topology will probably not change significantly. If you have a private extranet in place (e.g., for premium customers or strategic partners), for practical purposes you probably have the network topology for a private cloud in place already.

The security considerations you have today apply to a private cloud infrastructure, too. And the security tools you have in place (or should have in place) are also necessary for a private cloud and operate in the same way. Figure 3-1 shows the topological similarities between a secure extranet and a private cloud.

However, if you choose to use public cloud services, changing security requirements will require changes to your network topology. You must address how your existing network topology interacts with your cloud provider's network topology.

There are ***four significant risk factors*** in this use case:

1) **Ensuring the confidentiality and integrity** of your organization's data-in-transit to and from your public cloud provider
2) **Ensuring proper access control** (authentication, authorization, and auditing) to whatever resources you are using at your public cloud provider
3) **Ensuring the availability of the Internet-facing resources** in a public cloud that are being used by your organization, or have been assigned to your organization by your public cloud providers
4) **Replacing the established model of network zones and tiers with domains**

1. Ensuring Data Confidentiality and Integrity

- ✓ Some resources and data previously confined to a private network are now exposed to the Internet, and to a shared public network belonging to a third-party cloud provider. In a blog post, the author detailed a flaw in the digital signature algorithm used when ―... making Query (aka REST) requests to Amazon
- ✓ SimpleDB, to Amazon Elastic Compute Cloud (EC2), or to Amazon Simple Queue Service (SQS) over HTTP.‖ Although use of HTTPS (instead of HTTP) would have mitigated the integrity risk, users not using HTTPS (but using HTTP) did face an increased risk that their data could have been altered in transit without their knowledge

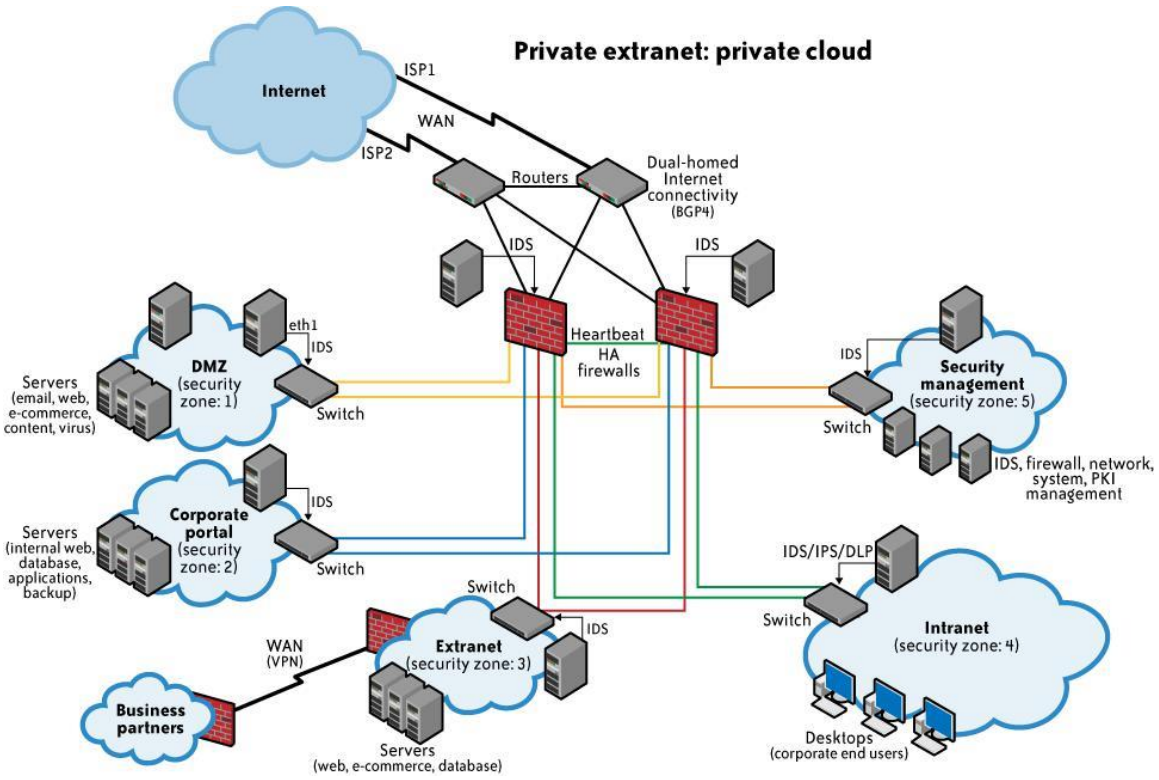


FIGURE 3-1. Generic network topology for private cloud computing

2. Ensuring Proper Access Control

Since some subset of these resources (or maybe even all of them) is now exposed to the Internet, an organization using a public cloud faces a significant increase in risk to its data. The ability to audit the operations of your cloud provider's network (let alone to conduct any real time monitoring, such as on your own network), even after the fact, is probably non-existent.

You will have decreased access to relevant network-level logs and data, and a limited ability to thoroughly conduct investigations and gather forensic data.

An ***example*** of the problems associated with this second risk factor is the issue of reused (reassigned) IP addresses. Generally speaking, cloud providers do not sufficiently ―age‖ IP addresses when they are no longer needed for one customer. Addresses are usually reassigned and reused by other customers as they become available.

From a cloud provider's perspective this makes sense. IP addresses are a finite quantity and a billable asset. However, from a customer's security perspective, the persistence of IP addresses that are no longer in use can present a problem.

A customer can't assume that network access to its resources is terminated upon release of its IP address. There is necessarily a lag time between the change of an IP address in DNS and the clearing of that address in DNS caches. There is a similar lag time between when physical (i.e., MAC) addresses are changed in ARP tables and when old ARP addresses are cleared from cache; an old address persists in ARP caches until they are cleared.

This means that even though addresses might have been changed, the (now) old addresses are still available in cache, and therefore they still allow users to reach these supposedly non-existent resources.

According to Simson Garfinkel:

A separate ongoing problem with the load balancers causes them to terminate any TCP/IP connection that contains more than 231 bytes. This means that objects larger than 2GB must be stored to S3 in several individual transactions, with each of those transactions referring to different byte ranges of the same object.

However, the issue of ―non-aged‖ IP addresses and unauthorized network access to resources does not apply only to routable IP addresses (i.e., resources intended to be reachable directly from the internet). The issue also applies to cloud providers' internal networks for customer use and the assignment of non-routable IP addresses.

Although your resources may not be directly reachable from the Internet, for management purposes your resources must be accessible within the cloud provider's network via private addressing. (Every public/Internetfacing resource also has a private address.)

Other customers of your cloud provider may not be well intentioned and might be able to reach your resources internally via the cloud provider's networks. As reported in *The Washington Post*, AWS has had problems with abuses of its resources affecting the public and other customers.

Some products emerging onto the market* will help alleviate the problem of IP address reuse, but unless cloud providers offer these products as managed services, customers are paying for yet another third-party product to solve a problem that their cloud provider's practices created for them.

3. Ensuring the Availability of Internet-Facing Resources

Reliance on network security has increased because an increased amount of data or an increased number of organizational personnel now depend on externally hosted devices to ensure the availability of cloud-provided resources. Consequently, the three risk factors enumerated in the preceding section must be acceptable to your organization.

Prefix hijacking involves announcing an autonomous system address space that belongs to someone else without her permission. Such announcements often occur because of a configuration mistake, but that misconfiguration may still affect the availability of your cloud-based resources.

According to a study presented to the North American Network Operators Group (NANOG) in February 2006, several hundred such misconfigurations occur per month. Probably the best known example of such a misconfiguration mistake occurred in February 2008 when Pakistan Telecom made an error by announcing a dummy route for YouTube to its own telecommunications partner, PCCW,based in Hong Kong.

The intent was to block YouTube within Pakistan because of some supposedly blasphemous videos hosted on the site.

The result was that YouTube was *globally* unavailable for two hours.In addition to misconfigurations, there are deliberate attacks as well. Although prefix hijacking due to deliberate attacks is far less common than misconfigurations, it still occurs and can block access to data.

According to the same study presented to NANOG, attacks occur fewer than 100 times per month. Although prefix hijackings are not new, that attack figure will certainly rise, and probably significantly, along with a rise in cloud computing.

As the use of cloud computing increases, the availability of cloud-based resources increases in value to customers. That increased value to customers translates to an increased risk of malicious activity to threaten that availability.

DNS attacks are another example of problems associated with this third risk factor. In fact, there are several forms of DNS attacks to worry about with regard to cloud computing. Although DNS attacks are not new and are not directly related to the use of cloud computing, the issue with DNS and cloud computing is an increase in an organization's risk at the network level because of increased external DNS querying along with some increased number of organizational personnel being more dependent on network security to ensure the availability of cloud-provided resources being used.

Other DNS problems impact cloud computing as well. Not only are there vulnerabilities in the DNS protocol and in implementations of DNS, but also there are fairly widespread DNS cache poisoning attacks whereby a DNS server is tricked into accepting incorrect information.

Although many people thought DNS cache poisoning attacks had been quashed several years ago, that is not true, and these attacks are still very much a problem—especially in the context of cloud computing. Variants of this basic cache poisoning attack include redirecting the target domain's name server (NS), redirecting the NS record to another target domain, and responding before the real NS (called *DNS forgery*).

A final example of problems associated with this third risk factor is ***denial of service (DoS)*** and ***distributed denial of service (DDoS)*** attacks. Again, although DoS/DDoS attacks are not new and are not directly related to the use of cloud computing, the issue with these attacks and cloud computing is an increase in an organization's risk at the network level because of some increased use of resources external to your organization's network.

For example, there continue to be rumors of continued DDoS attacks on AWS, making the services unavailablefor hours at a time to AWS users.§ (Amazon has not acknowledged that service interruptions are in fact due to DDoS attacks.)

However, when using IaaS, the risk of a DDoS attack is not only external (i.e., Internet-facing). There is also the risk of an internal DDoS attack through the portion of the IaaS provider's network used by customers (separate from the IaaS provider's corporate network).

That internal (non-routable) network is a shared resource, used by customers for access to their non-public instances (e.g., Amazon Machine Images or AMIs) as well as by the provider for management of its network and resources (such as physical servers).

If I were a rogue customer, there would be nothing to prevent me from using my customer access to this internal network to find and attack other customers, or the IaaS provider's infrastructure—and the provider would probably not have any detective controls in place to even notify it of such an attack.

The only preventive controls other customers would have would be how hardened their instances (e.g., AMIs) are, and whether they are taking advantage of a provider's capabilities to firewall off groups of instances (e.g., AWS).

8. Replacing the Established Model of Network Zones and Tiers with Domains?

The established isolation model of network zones and tiers no longer exists in the public IaaS and PaaS clouds. For years, network security has relied on zones, such as intranet versus extranet and development versus production, to segregate network traffic for improved security.

This model was based on exclusion—only individuals and systems in specific roles have access to specific zones. Similarly, systems within a specific tier often have only specific access within or across a specific tier. For example, systems within a presentation tier are not allowed to

communicate directly with systems in the database tier, but can communicate only with an authorized system within the application zone.

SaaS clouds built on public IaaS or PaaS clouds have similar characteristics. However, a public SaaS built on a private IaaS (e.g., Salesforce.com) may follow the traditional isolation model, but that topology information is not typically shared with customers.

The traditional model of network zones and tiers has been replaced in public cloud computing with ―security groups,‖ ―security domains,‖ or ―virtual data centers‖ that have logical separation between tiers but are less precise and afford less protection than the formerly established model.

For example, the security groups feature in AWS allows your virtual machines (VMs) to access each other using a virtual firewall that has the ability to filter traffic based on IP address (a specific address or a subnet), packet types (TCP, UDP, or ICMP), and ports (or a range of ports).

Domain names are used in various networking contexts and application-specific naming and addressing purposes, based on DNS. For example, Google's App Engine provides a logical grouping of applications based on domain names such as *mytestapp.test.mydomain.com* and *myprodapp.prod. mydomain.com*.

In the established model of network zones and tiers, not only were development systems logically separated from production systems at the network level, but these two groups of systems were also physically separated at the host level. With cloud computing, however, this separation no longer exists.

The cloud computing model of separation by domains provides logical separation for addressing purposes only. There is no longer any ―required‖ physical separation, as a test domain and a production domain may very well be on the same physical server.

Furthermore, the former logical network separation no longer exists; logical separation now is at the host level with both domains running on the same physical server and being separated only logically by VM monitors (hypervisors).

9. Explain in detail Network-Level Mitigation?

Given the factors, what can you do to mitigate these increased risk factors?

- ✓ First, note that network-level risks exist regardless of what aspects of ―cloud computing‖ services are being used (e.g., software-as-a-service, platform-as-a-service, or infrastructure-as-a-service). The primary determination of risk level is therefore not which *aaS is being used, but rather whether your organization intends to use or is using a public, private, or hybrid cloud.
- ✓ some IaaS clouds offer virtual network zoning, they may not match an internal private cloud environment that performs stateful inspection and other network security measures.

If your organization is large enough to afford the resources of a private cloud, your risks will decrease assuming you have a true private cloud that is internal to your network. In some cases, a private cloud located at a cloud provider's facility can help meet your security requirements but will depend on the provider capabilities and maturity.

You can reduce your confidentiality risks by using encryption; specifically by using validated implementations of cryptography for data-in-transit. Secure digital signatures make it much more difficult, if not impossible, for someone to tamper with your data, and this ensures data integrity.

Availability problems at the network level are far more difficult to mitigate with cloud computing—unless your organization is using a private cloud that is internal to your network topology. Even if your private cloud is a private (i.e., non-shared) external network at a cloud provider's facility, you will face increased risk at the network level.

A public cloud faces even greater risk. But let's keep some perspective here—greater than what? Even large enterprises with significant resources face considerable challenges at the network level of infrastructure security. Are the risks associated with cloud computing actually higher than the risks enterprises are facing today? Consider existing private and public extranets, and take into account partner connections when making such a comparison.

For large enterprises without significant resources, or for small to medium-size businesses (SMBs), is the risk of using public clouds (assuming that such enterprises lack the resources necessary for private
clouds) really higher than the risks inherent in their current infrastructures? In many cases, the answer is probably no—there is *not* a higher level of risk.

Table 3-1 lists security controls at the network level.

Threat outlook	**Low (with the exception of DoS attacks)**
Preventive controls	Network access control supplied by provider (e.g., firewall), encryption of data in transit (e.g., SSL, IPSec)
Detective controls	Provider-managed aggregation of security event logs (security incident and event management, or SIEM), network-based intrusion detection system/intrusion prevention system (IDS/IPS)

TABLE 3-1. Security controls at the network level

10.Explain in detail Host Level Infrastructure Security?

When reviewing host security and assessing risks, you should consider the context of cloud services delivery models (SaaS, PaaS, and IaaS) and deployment models (public, private, and hybrid).

Although there are no known new threats to hosts that are specific to cloud computing, some virtualization security threats—such as VM escape, system configuration drift, and insider threats by way of weak access control to the hypervisor—carry into the public cloud computing environment.

The dynamic nature (elasticity) of cloud computing can bring new operational challenges from a security management perspective. The operational model motivates rapid provisioning and fleeting instances of VMs. Managing vulnerabilities and patches is therefore much harder than just running a scan, as the rate of change is much higher than in a traditional data center.

SaaS and PaaS Host Security

In general, CSPs do not publicly share information related to their host platforms, host operating systems, and the processes that are in place to secure the hosts, since hackers can exploit that information when they are trying to intrude into the cloud service.

Hence, in the context of SaaS (e.g., Salesforce.com, Workday.com) or PaaS (e.g., Google App Engine, Salesforce.com's Force.com) cloud services, host security is opaque to customers and the responsibility of securing the hosts is relegated to the CSP.

To get assurance from the CSP on the security hygiene of its hosts, you should ask the vendor to share information under a nondisclosure agreement (NDA) or simply demand that the CSP share the information via a controls assessment framework such as SysTrust or ISO 27002.

From a controls assurance perspective, the CSP has to ensure that appropriate preventive and detective controls are in place and will have to ensure the same via a third-party assessment or ISO 27002 type assessment framework.

Since virtualization is a key enabling technology that improves host hardware utilization, among other benefits, it is common for CSPs to employ virtualization platforms, including Xen and VMware hypervisors, in their host computing platform architecture. You should understand how the provider is using virtualization technology and the provider's process for securing the virtualization layer.

Both the PaaS and SaaS platforms abstract and hide the host operating system from end users with a host abstraction layer. One key difference between PaaS and SaaS is the accessibility of the abstraction layer that hides the operating system services the applications consume.

In the case of SaaS, the abstraction layer is not visible to users and is available only to the developers and the CSP's operations staff, where PaaS users are given indirect access to the host abstraction layer in the form of a PaaS application programming interface (API) that in turn interacts with the host abstraction layer.

In short, if you are a SaaS or a PaaS customer, you are relying on the CSP to provide a secure host platform on which the SaaS or PaaS application is developed and deployed by the CSP and you, respectively.

IaaS Host Security

Unlike PaaS and SaaS, IaaS customers are primarily responsible for securing the hosts provisioned in the cloud. Given that almost all IaaS services available today employ virtualization at the host layer, host security in IaaS should be categorized as follows:

Virtualization software security

The software layer that sits on top of bare metal and provides customers the ability to create and destroy virtual instances. Virtualization at the host level can be accomplished using any of the virtualization models, including OS-level virtualization (Solaris containers, BSD jails, Linux-VServer), paravirtualization (a combination of the hardware version and versions of Xen and VMware), or hardware-based virtualization (Xen, VMware, Microsoft Hyper-V).

It is important to secure this layer of software that sits between the hardware and the virtual servers. In a public IaaS service, customers do not have access to this software layer; it is managed by the CSP only.

Customer guest OS or virtual server security

The virtual instance of an operating system that is provisioned on top of the virtualization layer and is visible to customers from the Internet; e.g., various flavors of Linux, Microsoft, and Solaris. Customers have full access to virtual servers.

Virtualization Software Security

Since the CSP manages the virtualization software that sits on top of the hardware, customers will have neither visibility nor access to this software. Hardware or OS virtualization enables the sharing of hardware resources across multiple guest VMs without interfering with

each other so that you can safely run several operating systems and applications at the same time on a single computer.

For the purpose of simplicity, we made an assumption that IaaS services are using ―bare metal hypervisor‖ technologies (also known as type 1 hypervisors), such as VMware ESX, Xen, Oracle VM, and Microsoft's Hyper-V.

These hypervisors support a variety of guest OSs, including Microsoft Windows, various Linux ―flavors,‖ and Sun's OpenSolaris.

Given that hypervisor virtualization is the essential ingredient that guarantees compartmentalization and isolation of customer VMs from each other in a multitenant environment, it is very important to protect the hypervisors from unauthorized users.

A new arms race between hacker and defender (CSP) in the realm of virtualization security is already underway. Since virtualization is very critical to the IaaS cloud architecture, any attack that could compromise the integrity of the compartments will be catastrophic to the entire customer base on that cloud.

A recent incident at a tiny UK-based company called Vaserv.com exemplifies the threat to hypervisor security. By exploiting a zero-day vulnerability in HyperVM, a virtualization application made by a company called Lxlabs, hackers destroyed 100,000 websites hosted by Vaserv.com.

The zero-day vulnerability gave the attackers the ability to execute sensitive Unix commands on the system, including rm -rf, which forces a recursive delete of all files. Evidently, just days before the intrusion, an anonymous user posted on a hacker website called milw0rm a long list of yet-unpatched vulnerabilities in Kloxo, a hosting control panel that integrates into HyperVM.

The situation was worse for approximately 50% of Vaserv's customers who signed up for unmanaged service, which doesn't include data backup. It remains unclear whether those website owners will ever be able to retrieve their lost data.

CSPs should institute the necessary security controls, including restricting physical and logical access to hypervisor and other forms of employed virtualization layers. IaaS customers should understand the technology and security process controls instituted by the CSP to protect the hypervisor.

This will help you to understand the compliance and gaps with reference to your host security standard, policies, and regulatory compliances. However, in general, CSPs lack transparency in this area and you may have no option but to take a leap of faith and trust CSPs to provide an ―isolated and secured virtualized guest OS.‖

Threats to the hypervisor

The integrity and availability of the hypervisor are of utmost importance and are key to guaranteeing the integrity and availability of a public cloud built on a virtualized environment.

A vulnerable hypervisor could expose all user domains to malicious insiders.
Furthermore, hypervisors are potentially susceptible to subversion attacks.

To illustrate the vulnerability of the virtualization layer, some members of the security research community demonstrated a ―Blue Pill‖ attack on a hypervisor. During Black Hat 2008 and Black Hat DC 2009‖ Joanna Rutkowska, Alexander Tereshkin, and Rafal Wojtczuk from Invisible Things Lab demonstrated a number of ways to compromise Xen's virtualization.#

Although Rutkowska and her team have identified problems with Xen implementations, generally they seem quite positive about the Xen approach. But their demonstration does illustrate the complexity of securing virtualized systems and the need for new approaches to protect hypervisors from such attacks.

Since virtualization layers within public clouds for the most part are proprietary and closed source (although some may employ a derivative of open source virtualization software such as Xen), the source code of software used by CSPs is not available for scrutiny by the security research community.

Virtual Server Security

Customers of IaaS have full access to the virtualized guest VMs that are hosted and isolated from each other by hypervisor technology. Hence customers are responsible for securing and ongoing security management of the guest VM.

A public IaaS, such as Amazon's Elastic Compute Cloud (EC2), offers a web services API to perform management functions such as provisioning, decommissioning, and replication of virtual servers on the IaaS platform. These system management functions, when orchestrated appropriately, can provide elasticity for resources to grow or shrink in line with workload demand.

The dynamic life cycle of virtual servers can result in complexity if the process to manage the virtual servers is not automated with proper procedures. From an attack surface perspective, the virtual server (Windows, Solaris, or Linux) may be accessible to anyone on the Internet, so sufficient network access mitigation steps should be taken to restrict access to virtual instances.

Typically, the CSP blocks all port access to virtual servers and recommends that customers use port 22 (Secure Shell or SSH) to administer virtual server instances. The cloud management API adds another layer of attack surface and must be included in the scope of securing virtual servers in the public cloud. Some of the new host security threats in the public IaaS include:

• Stealing keys used to access and manage hosts (e.g., SSH private keys)

• Attacking unpatched, vulnerable services listening on standard ports (e.g., FTP, NetBIOS, SSH)

• Hijacking accounts that are not properly secured (i.e., weak or no passwords for standard accounts)

• Attacking systems that are not properly secured by host firewalls

• Deploying Trojans embedded in the software component in the VM or within the VM image (the OS) itself

Securing virtual servers

The simplicity of self-provisioning new virtual servers on an IaaS platform creates a risk that insecure virtual servers will be created. Secure-by-default configuration needs to be ensured by following or exceeding available industry baselines. Securing the virtual server in the cloud requires strong operational security procedures coupled with automation of procedures. Here are some recommendations:

• **Use a secure-by-default configuration**. Harden your image and use a standard hardened image for instantiating VMs (the guest OS) in a public cloud. A best practice for cloud based applications is to build custom VM images that have only the capabilities and services necessary

to support the application stack. Limiting the capabilities of the underlying application stack not only limits the host's overall attack surface, but also greatly reduces the number of patches needed to keep that application stack secure.

• **Track the inventory of VM images and OS versions that are prepared for cloud hosting**. The IaaS provider provides some of these VM images. When a virtual image from the IaaS provider is used it should undergo the same level of security verification and hardening for hosts within the enterprise. The best alternative is to provide your own image that conforms to the same security standards as internal trusted hosts.

• **Protect the integrity of the hardened image from unauthorized access**.

• **Safeguard the private keys required to access hosts in the public cloud.**

In general, isolate the decryption keys from the cloud where the data is hosted—unless they are necessary for decryption, and then only for the duration of an actual decryption activity. If your application requires a key to encrypt and decrypt for continuous data processing, it may not be possible to protect the key since it will be collocated with the application.

• Include no authentication credentials in your virtualized images except for a key to decrypt the file system key.

• Do not allow password-based authentication for shell access.

• Require passwords for sudo* or role-based access (e.g., Solaris, SELinux).

• Run a host firewall and open only the minimum ports necessary to support the services on an instance.

• Run only the required services and turn off the unused services

• Install a host-based IDS such as OSSEC or Samhain .

• Enable system auditing and event logging, and log the security events to a dedicated log server. Isolate the log server with higher security protection, including accessing controls.

• If you suspect a compromise, shut down the instance, snapshot your block volumes, and back up the root file system. You can perform forensics on an uncompromised system later.

• Institute a process for patching the images in the cloud—both offline and instantiated images.

• Periodically review logs for suspicious activities.

Table 3-2 lists security controls at the host level.

Threat outlook	**High**
Preventive controls	Host firewall, access control, patching, hardening of system, strong authentication
Detective controls	Security event logs, host-based IDS/IPS

TABLE 3-2. Security controls at the host level

11.Explain in detail Application Level Infrastructure Security?

Infrastructure Security: The Application Level

Application or software security should be a critical element of your security program. Most enterprises with information security programs have yet to institute an application security program to address this realm.

Designing and implementing applications targeted for deployment on a cloud platform will require that existing application security programs reevaluate current practices and standards. The application security spectrum ranges from standalone single-user applications to sophisticated multiuser e-commerce applications used by millions of users.

Web applications such as content management systems (CMSs), wikis,portals, bulletin boards, and discussion forums are used by small and large organizations. A large number of organizations also develop and maintain custom-built web applications for their businesses using various web frameworks (PHP,† .NET,‡ J2EE,§ Ruby on Rails, Python, etc.).

According to SANS, until 2007 few criminals attacked vulnerable websites because other attack vectors were more likely to lead to an advantage in unauthorized economic or information access. Increasingly, however, advances in cross-site scripting (XSS) and other attacks have demonstrated that criminals looking for financial gain can exploit vulnerabilities resulting from web programming errors as new ways to penetrate important organizations.

In this section, we will limit our discussion to web application security: web applications in the cloud accessed by users with standard Internet browsers, such as Firefox, Internet Explorer, or Safari, from any computer connected to the Internet.

Since the browser has emerged as the end user client for accessing in-cloud applications, it is important for application security programs to include browser security into the scope of application security.

Together they determine the strength of end-to-end cloud security that helps protect the confidentiality, integrity, and availability of the information processed by cloud services.

12. Brief explanation about Application-Level Security Threats in cloud infra satructure

According to SANS,web application vulnerabilities in open source as well as custom-built applications accounted for almost half the total number of vulnerabilities discovered between November 2006 and October 2007.# The existing threats exploit well-known application vulnerabilities including cross-site scripting (XSS), SQL injection, malicious file execution, and other vulnerabilities resulting from programming errors and design flaws.

Armed with knowledge and tools, hackers are constantly scanning web applications (accessible from the Internet) for application vulnerabilities. They are then exploiting the vulnerabilities they discover for various illegal activities including financial fraud, intellectual property theft, converting trusted
websites into malicious servers serving client-side exploits, and phishing scams.

All web frameworks and all types of web applications are at risk of web application security defects, ranging from insufficient validation to application logic errors. It has been a common practice to use a combination of perimeter security controls and network- and host-based access controls to protect web applications deployed in a tightly controlled environment, including corporate intranets and private clouds, from external hackers.

Web applications built and deployed in a public cloud platform will be subjected to a high threat level, attacked, and potentially exploited by hackers to support fraudulent and illegal activities. In that threat model, web applications deployed in a public cloud (the SPI model) must be designed for an Internet threat model, and security must be embedded into the Software Development Life Cycle (SDLC); see Figure 3-2.

FIGURE 3-2. The SDLC

DoS and EDoS

Additionally, you should be cognizant of application-level DoS and DDoS attacks that can potentially disrupt cloud services for an extended time. These attacks typically originate from compromised computer systems attached to the Internet (routinely, hackers hijack and control computers infected by way of viruses/worms/malware and, in some cases, powerful unprotected servers).

Application-level DoS attacks could manifest themselves as high-volume web page reloads, XML* web services requests (over HTTP or HTTPS), or protocol-specific requests supported by a cloud service. Since these malicious requests blend with the legitimate traffic, it is extremely difficult to selectively filter the malicious traffic without impacting the service as a whole. For example, a DDoS attack on Twitter on August 6, 2009, brought the service down for several hours (see Figure 3-3).

FIGURE 3-3. DDoS attack on Twitter

Apart from disrupting cloud services, resulting in poor user experience and service-level impacts, DoS attacks can quickly drain your company's cloud services budget. DoS attacks on pay-as-you-go cloud applications will result in a dramatic increase in your cloud utility bill: you'll see increased use of network bandwidth, CPU, and storage consumption.

This type of attack is also being characterized as ***economic denial of sustainability (EDoS).***

The low barriers for small and medium-size enterprises to adopt cloud computing for legitimate use are also leveling the field for hackers. Using hijacked or exploited cloud accounts, hackers will be able to link together computing resources to achieve massive amounts of computing without any of the capital infrastructure costs. In the not-so-distant future, you might witness DoS attacks launched from IaaS or PaaS clouds against other cloud services (such hostile and offensive cloud models are being characterized as *dark clouds*).

End User Security

You, as a customer of a cloud service, are responsible for end user security tasks—security procedures to protect your Internet-connected PC—and for practicing ―safe surfing.‖ Protection measures include use of security software, such as anti-malware, antivirus, personal firewalls, security patches, and IPS-type software on your Internet-connected computer. The new mantra of ―the browser is your perating system‖ appropriately conveys the message that browsers have become the ubiquitous ―operating systems‖ for consuming cloud services.

All Internet browsers routinely suffer from software vulnerabilities that make them vulnerable to end user security attacks. Hence, our recommendation is that cloud customers take appropriate steps to protect browsers from attacks. To achieve end-to-end security in a cloud, it is essential for customers to maintain good browser hygiene.

The means keeping the browser (e.g.. Internet Explorer, Firefox, Safari) patched and updated to mitigate threats related to browser vulnerabilities. Currently, although browser security add-ons are not commercially available, users are encouraged to frequently check their browser vendor's website for security updates, use the auto-update feature, and install patches on a timely basis to maintain end user security

13. Brief explanation about Aspects of Data Security in cloud infra structure

With regard to data-in-transit, the primary risk is in not using a vetted encryption algorithm.Although this is obvious to information security professionals, it is not common for others to understand this requirement when using a public cloud, regardless of whether it is IaaS, PaaS, or SaaS.

It is also important to ensure that a protocol provides confidentiality as well as integrity (e.g., FTP over SSL [FTPS], Hypertext Transfer Protocol Secure [HTTPS], and Secure Copy Program [SCP])—particularly if the protocol is used for transferring data across the Internet. Merely encrypting data and using a non-secured protocol (e.g., ―vanilla‖ or ―straight‖ FTP or HTTP) can provide confidentiality, but does not ensure the integrity of the data (e.g., with the use of symmetric streaming ciphers).

Although using encryption to protect data-at-rest might seem obvious, the reality is not that simple. If you are using an IaaS cloud service (public or private) for simple storage (e.g., Amazon's Simple Storage Service or S3), encrypting data-at-rest is possible—and is strongly suggested.

However, encrypting data-at-rest that a PaaS or SaaS cloud-based application is using (e.g., Google Apps, Salesforce.com) as a compensating control is not always feasible.

Data-at-rest used by a cloud-based application is generally not encrypted, because encryption would prevent indexing or searching of that data. Generally speaking, with data-at-rest, the economics of cloud computing are such that PaaSbased applications and SaaS use a multitenancy architecture.

In other words, data, when processed by a cloud-based application or stored for use by a cloud-based application, is commingled with other users' data (i.e., it is typically stored in a massive data store, such as Google's BigTable).

Although applications are often designed with features such as data tagging to prevent unauthorized access to commingled data, unauthorized access is still possible through some exploit of an

application vulnerability (e.g., Google's unauthorized data sharing between users of Documents and Spreadsheets in March 2009). Although some cloud providers have their applications reviewed by third parties or verified with third-party application security tools, data is not on a platform dedicated solely to one organization.

Although an organization's data-in-transit might be encrypted during transfer to and from a cloud provider, and its data-at-rest might be encrypted if using simple storage (i.e., if it is not associated with a specification application), an organization's data is definitely not encrypted if it is processed in the cloud (public or private). For any application to process data, that data *must be* unencrypted. Until June 2009, there was no known method for fully processing encrypted data.

Therefore, unless the data is in the cloud for only simple storage, the data will be unencrypted during at least part of its life cycle in the cloud—processing at a minimum. In June 2009, IBM announced that one of its researchers, working with a graduate student from Stanford University, had developed a fully homomorphic encryption scheme which allows data to be processed *without being decrypted*.

This is a huge advance in cryptography, and it will have a significant positive impact on cloud computing as soon as it moves into at Stanford University, but IBM's announcement bettered even that promising work. Although the homomorphic scheme has broken the theoretical barrier to fully homomorphic encryption, it required immense computational effort. According to Ronald Rivest (MIT professor and coinventor of the famous RSA encryption scheme), the steps to make it practical won't be far behind.

Other cryptographic research efforts are underway to limit the amount of data that would need to be decrypted for processing in the cloud, such as predicate encryption. Whether the data an organization has put into the cloud is encrypted or not, it is useful and might be required (for audit or compliance purposes) to know exactly where and when the data was specifically located within the cloud. For **example**, the data might have been transferred to a cloud provider, such as Amazon Web Services (AWS), on date *x*1 at time *y*1 and stored in a bucket on Amazon's S3 in *example1.s3.amazonaws.com*, then processed on date*x*2 at time *y*2 on an instance being used by an organization on Amazon's Elastic Compute Cloud (EC2) in *ec2-67-202-51-223.compute-1.amazonaws.com*, then restored in another bucket,

example2.*s3.amazonaws.com*, before being brought back into the organization for storage in an internal data warehouse belonging to the marketing operations group on date *x*3 at time *y*3. Following the path of data (mapping application data flows or data path visualization) is known as *data lineage*, and it is important for an auditor's assurance (internal, external, and regulatory).

However, providing data lineage to auditors or management is time-consuming, even when
the environment is completely under an organization's control. Trying to provide accurate reporting on data lineage for a public cloud service is really not possible. In the preceding example, on what physical system is that bucket on *example1.s3.amazonaws.com*, and specifically where is (or was) that system located? What was the state of that physical system then, and how would a customer or auditor verify that information?

Even if data lineage can be established in a public cloud, for some customers there is an even
more challenging requirement and problem: proving data provenance—not just proving the integrity of the data, but the more specific provenance of the data. There is an important difference between the two terms. *Integrity of data* refers to data that has not been changed in an unauthorized manner or by an unauthorized person. *Provenance* means not only that the data has integrity, but also that it is computationally accurate; that is, the data was accurately calculated.

For **example**, consider the following financial equation: SUM((((2*3)*4)/6)−2) = $2.00 With that equation, the expected answer is $2.00. If the answer were different, there would be an integrity problem. Of course, the assumption is that the $2.00 is in U.S. dollars,

but the assumption could be incorrect if a different dollar is used with the following associated assumptions:

• The equation is specific to the Australian, Bahamian, Barbadian, Belize, Bermudian, Brunei, Canadian, Cayman Islands, Cook Islands, East Caribbean, Fijian, Guyanese, Hong Kong, Jamaican, Kiribati, Liberian, Namibian, New Zealand, Samoan, Singapore, Solomon Islands, Surinamese, New Taiwan, Trinidad and Tobago, Tuvaluan, or Zimbabwean dollar.

• The dollar is meant to be converted from another country's dollars into U.S. dollars.

• The correct exchange rate is used and the conversion is calculated correctly and can be proven.

In this ***example***, if the equation satisfies those assumptions, the equation has integrity but not provenance.

There are many real-world examples in which data integrity is insufficient and data provenance is also required. Financial and scientific calculations are two obvious examples.

How do you prove data provenance in a cloud computing scenario when you are using shared resources? Those resources are not under your physical or even logical control, and you probably have no ability to track the systems used or their state at the times you used them—even if you know some identifying information about the systems (e.g., their IP addresses) and the ―general‖ location (e.g., a country, and not even a specific data center).

A final aspect of data security is *data remanence*. ―Data remanence is the residual representation of data that has been in some way nominally erased or removed. This residue may be due to data being left intact by a nominal delete operation, or through physical properties of the storage medium.

Data remanence may make inadvertent disclosure of sensitive information possible, should the storage media be released into an uncontrolled environment (e.g., thrown in the trash, or given to a third party).‖

The risk posed by data remanence in cloud services is that an organization's data can be inadvertently exposed to an unauthorized party—regardless of which cloud service you are using (SaaS, PaaS, or IaaS). When using SaaS or PaaS, the risk is almost certainly unintentional or inadvertent exposure.

However, that is not reassuring after an unauthorized disclosure, and potential customers should question what third-party tools or reviews are used to help validate the security of the provider's applications or platform.

In spite of the increased importance of data security, the attention that cloud service providers (CSPs) pay to data remanence is strikingly low. Many do not even mention data remanence in their services. And if the subject of data security is broached, many CSPs rather glibly refer to compliance with U.S. Department of Defense (DoD) 5220.22-M (the National Industrial Security Program Operating Manual).

We say ―glibly‖ because it appears that providers (and other information technology vendors) have not actually read this manual. DoD 5220.22-M states the two approved methods of data (destruction) security, but does not provide any specific requirements for how these two methods are to be achieved, nor does it provide any standards for how these methods are to be accomplished.

"8-301. Clearing and Sanitization"

Instructions on clearing, sanitization, and release of information systems (IS) media shall be issued by the accrediting Cognizant Security Agency (CSA).

"a. Clearing"

Clearing is the process of eradicating the data on media before reusing the media in an environment that provides an acceptable level of protection for the data that was on the media before clearing. All internal memory, buffer, or other reusable memory shall be cleared to effectively deny access to previously stored information.

"b. Sanitization"

Sanitization is the process of removing the data from media before reusing the media in an environment that does not provide an acceptable level of protection for the data that was on the media before sanitizing. IS resources shall be sanitized before they are released from classified information controls or released for use at a lower classification level.

Data Security Mitigation

If prospective customers of cloud computing services expect that data security will serve as compensating controls for possibly weakened infrastructure security, since part of a customer's infrastructure security moves beyond its control and a provider's infrastructure security may (for many enterprises) or may not (for small to medium-size businesses, or SMBs) be less robust than expectations, you will be disappointed.

Although data-in-transit can and should be encrypted, any use of that data in the cloud, beyond simple storage, requires that it be decrypted. Therefore, it is almost certain that in the cloud, data will be unencrypted.

And if you are using a PaaS-based application or SaaS, customer-unencrypted data will also almost certainly be hosted in a multitenancy environment (in public clouds).

Add to that exposure even many providers' failure to adequately address such a basic security concern as data remanence, and the risks of data security for customers are significantly increased.

So, what should you do to mitigate these risks to data security? The only viable option for mitigation is to ensure that any sensitive or regulated data is not placed into a public cloud (or that you encrypt data placed into the cloud for simple storage only).

Given the economic considerations of cloud computing today, as well as the present limits of cryptography, CSPs are not offering robust enough controls around data security.

It may be that those economics change and that providers offer their current services, as well as a ―regulatory cloud environment‖ (i.e., an environment where customers are willing to pay more for enhanced security controls to properly handle sensitive and regulated data).

Currently, the only viable option for mitigation is to ensure that any sensitive or regulated data is not put into a public cloud.

Provider Data and Its Security

In addition to the security of your own customer data, customers should also be concerned about what data the provider collects and how the CSP protects that data. Specifically with regard to your customer data, what metadata does the provider have about your data, how is it secured, and what access do you, the customer, have to that metadata?

As your volume of data with a particular provider increases, so does the value of that metadata. Additionally, your provider collects and must protect a huge amount of security-related data.

For example, at the network level, your provider should be collecting, monitoring, and protecting firewall, intrusion prevention system (IPS), security incident and event management (SIEM), and router flow data.

At the host level your provider should be collecting system logfiles, and at the application level SaaS providers should be collecting application log data, including authentication and authorization information.

What data your CSP collects and how it monitors and protects that data is important to the provider for its own audit purposes (e.g., SAS 70, as discussed in Chapter 8). Additionally, this information is important to both providers and customers in case it is needed for incident response and any digital forensics required for incident analysis.

Storage

For data stored in the cloud (i.e., storage-as-a-service), we are referring to IaaS and not data associated with an application running in the cloud on PaaS or SaaS. The same three information security concerns are associated with this data stored in the cloud (e.g., Amazon's S3) as with data stored elsewhere: confidentiality, integrity, and availability.

Confidentiality

When it comes to the confidentiality of data stored in a public cloud, you have two potential concerns. First, what access control exists to protect the data? Access control consists of both authentication and authorization.

For large organizations, this coarse authorization presents significant security concerns unto itself. Often, the only authorization levels cloud vendors provide are administrator authorization (i.e., the owner of the account itself) and user authorization (i.e., all other authorized users)—with no levels in between (e.g., business unit administrators, who are authorized to approve access for their own business unit personnel).

Again, these access control issues are not unique to CSPs, and we discuss them in much greater detail in the following chapter.What is definitely relevant to this section, however, is the second potential concern: how is the data that is stored in the cloud actually protected? For all practical purposes, protection of data stored in the cloud involves the use of encryption.

So, is a customer's data actually encrypted when it is stored in the cloud? And if so, with what encryption algorithm, and with what key strength? It depends, and specifically, it depends on which CSP you are using.

For example, EMC's MozyEnterprise does encrypt a customer's data. However, AWS S3 does *not* encrypt a customer's data. Customers are able to encrypt their own data themselves prior to uploading, but S3 does not provide encryption.

If a CSP does encrypt a customer's data, the next consideration concerns what encryption algorithm it uses. Not all encryption algorithms are created equal. Cryptographically, many algorithms provide insufficient security.

Only algorithms that have been publicly vetted by a formal standards body (e.g., NIST) or at least informally by the cryptographic community should be used. Any algorithm that is proprietary should absolutely be avoided. Note that we are talking about symmetric encryption algorithms here.

Symmetric encryption (see Figure 4-1) involves the use of a single secret key for both the encryption and decryption of data. Only symmetric encryption has the speed and computational efficiency to handle encryption of large volumes of data. It would be highly unusual to use an asymmetric algorithm for this encryption use case. (See Figure 4-2.)

Although the example in Figure 4-1 is related to email, the same concept (i.e., a single shared, secret key) is used in data storage encryption.

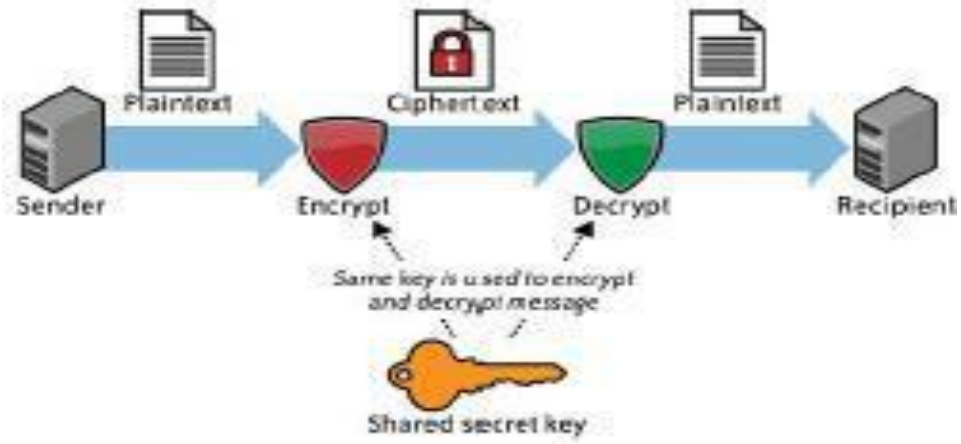


FIGURE 4-1. Symmetric encryption

Although the example in Figure 4-2 is related to email, the same concept (i.e., a public key and a private key) is *not* used in data storage encryption.

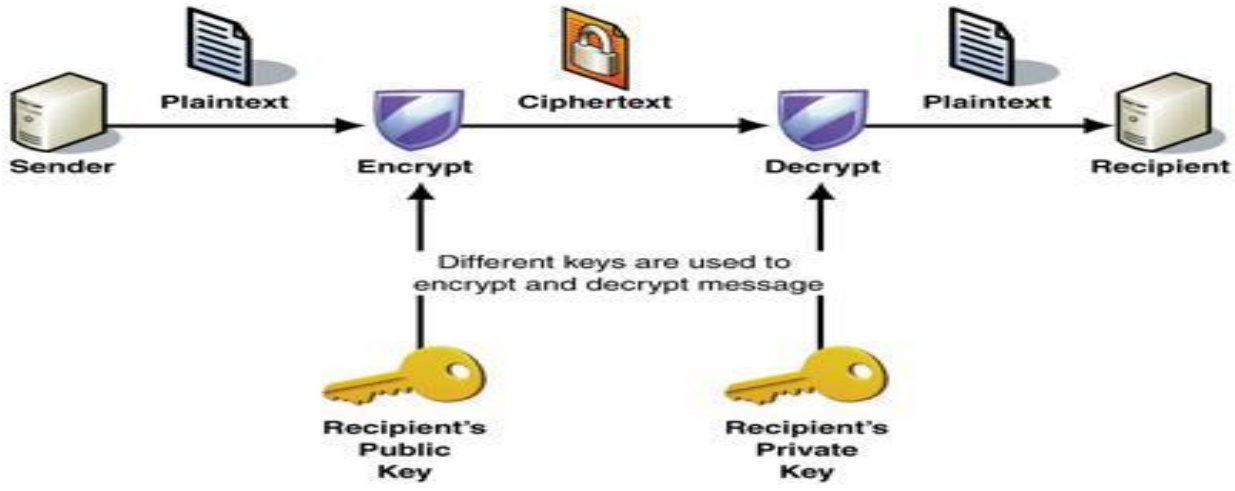


FIGURE 4-2. Asymmetric encryption

The next consideration for you is what key length is used. With symmetric encryption, the longer the key length (i.e., the greater number of bits in the key), the stronger the encryption. Although long key lengths provide more protection, they are also more computationally intensive, and may strain the capabilities of computer processors.

What can be said is that key lengths should be a minimum of 112 bits for Triple DES (Data Encryption Standard) and 128-bits for AES (Advanced Encryption Standard)—both NIST-approved algorithms.

Another confidentiality consideration for encryption is key management. How are the encryption keys that are used going to be managed—and by whom?

Are you going to manage your own keys? Hopefully, the answer is yes, and hopefully you have the expertise to manage your own keys. It is not recommended that you entrust a cloud provider to manage your keys—at least not the same provider that is handling your data. This means additional resources and capabilities are necessary.

That being said, proper key management is a complex and difficult task. At a minimum, a customer should consult all three parts of NIST's 800-57,―Recommendation for Key Management‖:

- ―Part 1: General‖
- ―Part 2: Best Practices for Key Management Organization‖
- ―Part 3: Application-Specific Key Management Guidance (Draft)‖

Because key management is complex and difficult for a single customer, it is even more complex and difficult for CSPs to try to properly manage multiple customers' keys. For that reason, several CSPs do not do a good job of managing customers' keys.

For example, it is common for a provider to encrypt all of a customer's data with a single key. Even worse, we are aware of one cloud storage provider that uses a single encryption key for all of its customers! The Organization for the Advancement of Structured Information Standards (OASIS) Key Management Interoperability Protocol (KMIP) is trying to address such issues.

Integrity

In addition to the confidentiality of your data, you also need to worry about the integrity of your data. Confidentiality does not imply integrity; data can be encrypted for confidentiality purposes, and yet you might not have a way to verify the integrity of that data.

Encryption alone is sufficient for confidentiality, but integrity also requires the use of message authentication codes (MACs). The simplest way to use MACs on encrypted data is to use a block symmetric algorithm (as opposed to a streaming symmetric algorithm) in cipher block chaining (CBC) mode, and to include a one-way hash function.

This is not for the cryptographically uninitiated—and it is one reason why effective key management is difficult. At the very least, cloud customers should be asking providers about these matters.

Not only is this important for the integrity of a customer's data, but it will also serve to provide insight on how sophisticated a provider's security program is—or is not. Remember, however, that not all providers encrypt customer data, especially for PaaS and SaaS services.

Another aspect of data integrity is important, especially with bulk storage using IaaS. Once a customer has several gigabytes (or more) of its data up in the cloud for storage, how does the customer check on the integrity of the data stored there?

There are IaaS transfer costs associated with moving data into and back down from the cloud,* as well as network utilization (bandwidth) considerations for the customer's own network. What a customer really wants to do is to validate the integrity of its data while that data remains in the cloud—without having to download and reupload that data.

This task is even more difficult because it must be done in the cloud without explicit knowledge of the whole data set. Customers generally do not know on which physical machines their data is stored, or where those systems are located. Additionally, that data set is probably dynamic and changing frequently.

Those frequent changes obviate the effectiveness of traditional integrity insurance techniques. What is needed instead is a proof of retrievability—that is, a mathematical way to verify the integrity of the data as it is dynamically stored in the cloud.†

Availability

Assuming that a customer's data has maintained its confidentiality and integrity, you must also be concerned about the availability of your data. There are currently three major threats in this regard—none of which are new to computing, but all of which take on increased importance in cloud computing because of increased risk.

The second threat to availability is the CSP's own availability. No CSPs offer the sought-after ―five 9s‖ (i.e., 99.999%) of uptime. A customer would be lucky to get ―three 9s‖ of uptime. As Table 4-1 shows, there is a considerable difference between five 9s and three 9s.

TABLE 4-1. Percentage of uptime

	Total downtime (HH:MM:SS)		
Availability	Per day	Per month	Per year
99.999%	00:00:00.4	00:00:26	00:05:15
99.99%	00:00:08	00:04:22	00:52:35
99.9%	00:01:26	00:43:49	08:45:56
99%	00:14:23	07:18:17	87:39:29

* For example, as of April 2009, AWS S3 charges $0.100 per gigabyte for all data transferred in, and $0.170 per gigabyte (for the first 10 TB) per month for all data transferred out.

A number of high-profile cloud provider outages have occurred. For example, Amazon's S3 suffered a 2.5-hour outage in February 2008 and an eight-hour outage in July 2008. AWS is one of the more mature cloud providers, so imagine the difficulties that other, smaller or less mature cloud providers are having.

These Amazon outages were all the more apparent because of the relatively large number of customers that the S3 service supports—and whom are highly (if not totally) reliant on S3's availability for their own operations.

In addition to service outages, in some cases data stored in the cloud has actually been lost. For example, in March 2009, ―cloud-based storage service provider Carbonite Inc. filed a lawsuit charging that faulty equipment from two hardware providers caused backup failures that resulted in the company losing data for 7,500 customers two years ago.‖

A larger question for cloud customers to consider is whether cloud storage providers will even be in business in the future. In February 2009, cloud provider Coghead suddenly shut down,giving

its customers fewer than 90 days (nine weeks) to get their data off its servers—or lose it altogether.

Finally, prospective cloud storage customers must be certain to ascertain just what services their provider is actually offering. Cloud storage does not mean the stored data is actually backed up. Some cloud storage providers do back up customer data, in addition to providing storage. However, many cloud storage providers do not back up customer data, or do so only as an additional service for an additional cost.

For example, ―data stored in Amazon S3, Amazon SimpleDB, or Amazon Elastic Block Store is redundantly stored in multiple physical locations as a normal part of those services and at no additional charge.‖ However, ―data that is maintained within running instances on Amazon EC2, or within Amazon S3 and Amazon SimpleDB, is all customer data and therefore AWS does not perform backups.‖ For availability, this is a seemingly simple yet critical question that customers should be asking of cloud storage providers.

All three of these considerations (confidentiality, integrity, and availability) should be encapsulated in a CSP's service-level agreement (SLA) to its customers. However, at this time, CSP SLAs are extremely weak—in fact, for all practical purposes, they are essentially worthless. Even where a CSP appears to have at least a partially sufficient SLA, how that SLA actually gets measured is problematic. For all of these reasons, data security considerations and how data is actually stored in the cloud should merit considerable attention by customers.

14.Explain in detail about Identity and Access Management

Why IAM?

Traditionally, organizations invest in IAM practices to improve operational efficiency and to comply with regulatory, privacy, and data protection requirements:

Improve operational efficiency

Properly architected IAM technology and processes can improve efficiency by automating user on-boarding and other repetitive tasks (e.g., self-service for users requesting password resets that otherwise will require the intervention of system administrators using a help desk ticketing system).

Regulatory compliance management

To protect systems, applications, and information from internal and external threats (e.g., disgruntled employees deleting sensitive files) and to comply with various regulatory, privacy, and data protection requirements (e.g., HIPAA, SOX), organizations implement an ―IT general and application-level controls‖ framework derived from industry standard frameworks such as ISO 27002 and Information Technology Infrastructure Library (ITIL).

IAM processes and practices can help organizations meet objectives in the area of access control and operational security (e.g., enforcement of compliance requirements such as ―segregation of duties‖ and assignment of limited privileges for staff members to perform their duties).

Auditors routinely map internal controls to IT controls as they support management of regulatory compliance processes including Payment Card Industry (PCI) Data Security Standards (DSSs) and the Sarbanes-Oxley Act of 2003 (SOX).

In addition to improving operational efficiencies and effective compliance management, IAM can enable new IT delivery and deployment models (i.e., cloud services). For example, federated identity, a key IAM component, enables the linking and portability of identity information across trust boundaries.

As such, it enables enterprises and cloud service providers to bridge security domains through web single sign-on and federated user provisioning.

Some of the cloud use cases that require IAM support from the CSP include:

• Employees and on-site contractors of an organization accessing a SaaS service using identity federation (e.g., sales and support staff members accessing Salesforce.com with corporate identities and credentials)
• IT administrators accessing the CSP management console to provision resources and access for users using a corporate identity (e.g., IT administrators of Newco.com provisioning virtual machines or VMs in Amazon's EC2 service, configured with identities, entitlements, and credentials for operating the VMs [i.e., start, stop, suspend, and delete VMs])
• Developers creating accounts for partner users in a PaaS platform (e.g., developers from Newco.com provisioning accounts in Force.com for Partnerco.com employees contracted to perform business process tasks for Newco.com)
• End users accessing storage service in the cloud (e.g., Amazon S3) and sharing files and objects with users, within and outside a domain using access policy management features
• An application residing in a cloud service provider (e.g., Amazon EC2) accessing storage from another cloud service (e.g., Mosso)

Since IAM features such as SSO allow applications to externalize authentication features, businesses can rapidly adopt *aaS services (an example is Salesforce.com) by reducing the time required to integrate with service providers.

IAM capabilities can also help a business outsource a process or service to partners with a reduced impact to the business's privacy and security; for example, employees of an order fulfillment partner of a merchant can use their federated product fulfillment process.

In short, extending your IAM strategy, practice, and architecture allows your organization to extend your user access management practices and processes to the cloud. Hence, organizations with established IAM practices can rapidly adopt cloud services while maintaining the efficiency and efficacy of their security controls.

IAM Challenges

One critical challenge of IAM concerns managing access for diverse user populations (employees, contractors, partners, etc.) accessing internal and externally hosted services. IT is constantly challenged to rapidly provision appropriate access to the users whose roles and responsibilities often change for business reasons. Another issue is the turnover of users within the organization.

Turnover varies by industry and function—seasonal staffing fluctuations in finance departments, for example—and can also arise from changes in the business, such as mergers and acquisitions, new product and service releases, business process outsourcing, and changing responsibilities. As a result, sustaining IAM processes can turn into a persistent challenge.

Access policies for information are seldom centrally and consistently applied. Organizations
can contain disparate directories, creating complex webs of user identities, access rights, and procedures. This has led to inefficiencies in user and access management processes while exposing these organizations to significant security, regulatory compliance, and reputation risks.

To address these challenges and risks, many companies have sought technology solutions to enable centralized and automated user access management. Many of these initiatives are entered into with high expectations, which is not surprising given that the problem is often large and complex.

Most often those initiatives to improve IAM can span several years and incur considerable cost. Hence, organizations should approach their IAM strategy and architecture with both business and IT drivers that address the core inefficiency issues while preserving the control's efficacy (related to access control). Only then will the organizations have a higher likelihood of success and return on investment.

IAM Definitions

The basic concepts and definitions of IAM functions for any service:

Authentication

Authentication is the process of verifying the identity of a user or system (e.g., Lightweight Directory Access Protocol [LDAP] verifying the credentials presented by the user, where the identifier is the corporate user ID that is unique and assigned to an employee or contractor). Authentication usually connotes a more robust form of identification.

In some use cases, such as service-to-service interaction, authentication involves verifying the network service requesting access to information served by another service (e.g., a travel web service that is connecting to a credit card gateway to verify the credit card on behalf of the user).

Authorization

Authorization is the process of determining the privileges the user or system is entitled to once the identity is established. In the context of digital services, authorization usually follows the authentication step and is used to determine whether the user or service has the necessary privileges to perform certain operations—in other words, authorization is the process of enforcing policies.

Auditing

In the context of IAM, auditing entails the process of review and examination of authentication, authorization records, and activities to determine the adequacy of IAM system controls, to verify compliance with established security policies and procedures (e.g., separation of duties), to detect reaches in security services (e.g., privilege escalation), and to recommend any changes that are indicated for countermeasures.

IAM Architecture and Practice

IAM is not a monolithic solution that can be easily deployed to gain capabilities immediately. It is as much an aspect of architecture (see Figure 5-1) as it is a collection of technology components, processes, and standard practices. Standard enterprise IAM architecture encompasses several layers of technology, services, and processes.

At the core of the deployment architecture is a directory service (such as LDAP or Active Directory) that acts as a repository for the identity, credential, and user attributes of the organization's user pool.

The directory interacts with IAM technology components such as authentication, user management, provisioning, and federation services that support the standard IAM practice and processes within the organization.

It is not uncommon for organizations to use several directories that were deployed for environment-specific reasons (e.g., Windows systems using Active Directory, Unix systems

using LDAP) or that were integrated into the environment by way of business mergers and acquisitions.

The IAM processes to support the business can be broadly categorized as follows:

User management

Activities for the effective governance and management of identity life cycles

Authentication management

Activities for the effective governance and management of the process for determining that an entity is who or what it claims to be

Authorization management

Activities for the effective governance and management of the process for determining entitlement rights that decide what resources an entity is permitted to access in accordance with the organization's policies

Access management

Enforcement of policies for access control in response to a request from an entity (user, services) wanting to access an IT resource within the organization

Data management and provisioning

Propagation of identity and data for authorization to IT resources via automated or manual processes

Monitoring and auditing

Monitoring, auditing, and reporting compliance by users regarding access to resources within the organization based on the defined policies

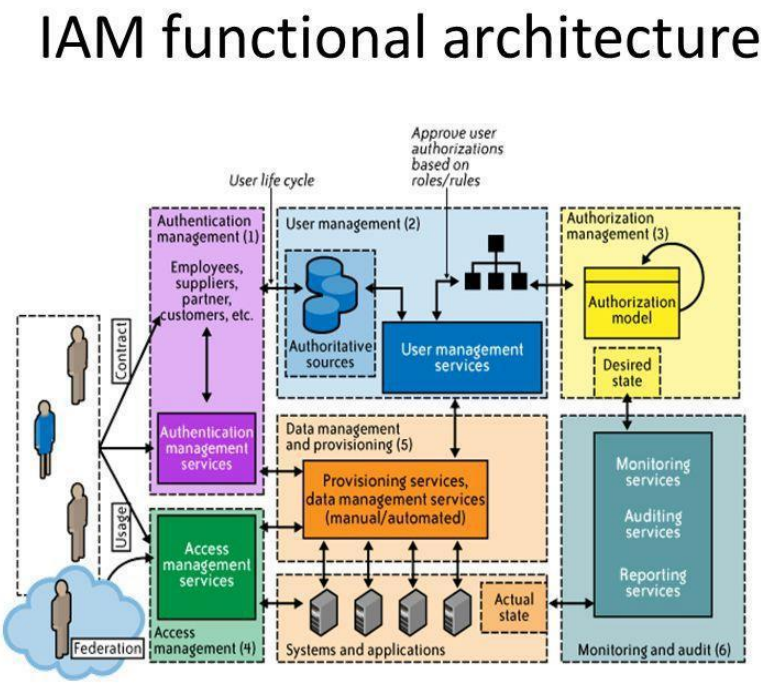


FIGURE 5-1. Enterprise IAM functional architecture

IAM processes support the following operational activities:

Provisioning

This is the process of on-boarding users to systems and applications. These processes provide users with necessary access to data and technology resources. The term typically is used in reference to enterprise-level resource management.

Provisioning can be thought of as a combination of the duties of the human resources and IT departments, where users are given access to data repositories or systems, applications, and databases based on a unique user identity. Deprovisioning works in the opposite manner, resulting in the deletion or deactivation of an identity or of privileges assigned to the user identity.

Credential and attribute management

These processes are designed to manage the life cycle of credentials and user attributes—create, issue, manage, revoke—to minimize the business risk associated with identity impersonation and inappropriate account use. Credentials are usually bound to an individual and are verified during the authentication process.

The processes include provisioning of attributes, static (e.g., standard text password) and dynamic (e.g., one-time password) credentials that comply with a password standard (e.g., passwords resistant to dictionary attacks), handling password expiration, encryption management of credentials during transit and at rest, and access policies of user attributes (privacy and handling of attributes for various regulatory reasons).

Entitlement management

Entitlements are also referred to as *authorization policies*. The processes in this domain address the provisioning and deprovisioning of privileges needed for the user to access resources including systems, applications, and databases.

Proper entitlement management ensures that users are assigned only the required privileges (least privileges) that match with their job functions. Entitlement management can be used to strengthen the security of web services, web applications, legacy applications, documents and files, and physical security systems.

Compliance management

This process implies that access rights and privileges are monitored and tracked to ensure the security of an enterprise's resources. The process also helps auditors verify compliance to various internal access control policies, and standards that include practices such as segregation of duties, access monitoring, periodic auditing, and reporting.

An example is a user certification process that allows application owners to certify that only authorized users have the privileges necessary to access business-sensitive information.

Identity federation management

Federation is the process of managing the trust relationships established beyond the internal network boundaries or administrative domain boundaries among distinct organizations.

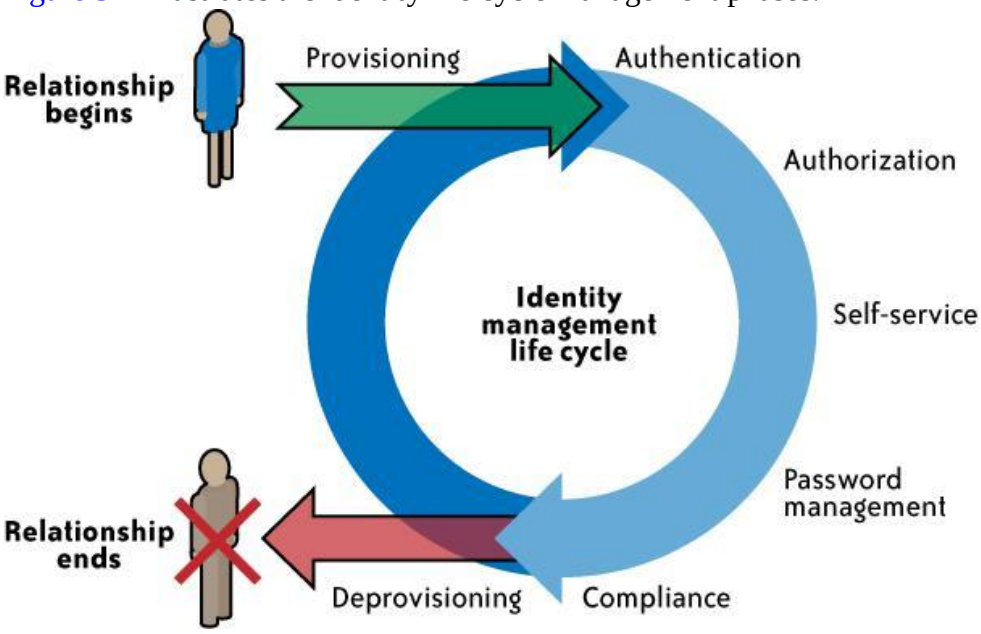
A federation is an association of organizations that come together to exchange information about their users and resources to enable collaborations and transactions (e.g., sharing user information with the organizations' benefits systems managed by a third-party provider). Federation of identities to service providers will support SSO to cloud services.

Centralization of authentication (authN) and authorization (authZ)

A central authentication and authorization infrastructure alleviates the need for application developers to build custom authentication and authorization features into their applications. Furthermore, it promotes a loose coupling architecture where applications become agnostic to the authentication methods and policies.

This approach is also called an ―externalization of authN and authZ‖ from applications.

Figure 5-2 illustrates the identity life cycle management phases.



15.Explain in detail about IAM Practices in the Cloud

When compared to the traditional applications deployment model within the enterprise, IAM practices in the cloud are still evolving. In the current state of IAM technology, standards support by CSPs (SaaS, PaaS, and IaaS) is not consistent across providers.

Although large providers such as Google, Microsoft, and Salesforce.com seem to demonstrate basic IAM capabilities, our assessment is that they still fall short of enterprise IAM requirements for managing regulatory, privacy, and data protection requirements. Table 5-2 illustrates the current maturity model, based on the authors' assessment, generalized across SPI service delivery models.

TABLE 5-2. Comparison of SPI maturity models in the context of IAM

Level	**SaaS**	**PaaS**	**IaaS**
User Management, New Users	Capable	Immature	Aware
UserManagement,User Modifications	Capable	Immature	Immature
Authentication Management	Capable	Aware	Capable
Authorization Management	Aware	Immature	Immature

The maturity model takes into account the dynamic nature of IAM users, systems, and applications in the cloud and addresses the four key components of the IAM automation process:
• User Management, New Users
• User Management, User Modifications
• Authentication Management
• Authorization Management

By matching the model's descriptions of various maturity levels with the cloud services delivery model's (SaaS, PaaS, IaaS) current state of IAM, a clear picture emerges of IAM maturity across the four IAM components. If, for example, the service delivery model (SPI) is ―immature‖ in one area but ―capable‖ or ―aware‖ in all others, the IAM maturity model can help focus attention on the area most in need of attention.

Although the principles and purported benefits of established enterprise IAM practices and processes are applicable to cloud services, they need to be adjusted to the cloud environment. Broadly speaking, user management functions in the cloud can be categorized as follows:
• Cloud identity administration
• Federation or SSO
• Authorization management
• Compliance management
We will now discuss each of the aforementioned practices in detail.

Cloud Identity Administration

Cloud identity administrative functions should focus on life cycle management of user identities in the cloud—provisioning, deprovisioning, identity federation, SSO, password or credentials management, profile management, and administrative management.Organizations that are not capable of supporting federation should explore cloud-based identity management services.

This new breed of services usually synchronizes an organization's internal directories with its directory (usually multitenant) and acts as a proxy IdP for the organization. By federating identities using either an internal Internet-facing IdP or a cloud identity management service provider, organizations can avoid duplicating identities and attributes and storing them with the CSP.

Given the inconsistent and sparse support for identity standards among CSPs, customers may have to devise custom methods to address user management functions in the cloud. Provisioning users when federation is not su pported can be complex and laborious.

It is not unusual for organizations to employ manual processes, web-based administration, outsourced (delegated) administration that involves uploading of spreadsheets, and execution of custom scripts at both the customer and CSP locations.

The latter model is not desirable as it is not scalable across multiple CSPs and will be costly to manage in the long run.

Federated Identity (SSO)

Organizations planning to implement identity federation that enables SSO for users can take one of the following two paths (architectures):
• Implement an enterprise IdP within an organization perimeter.

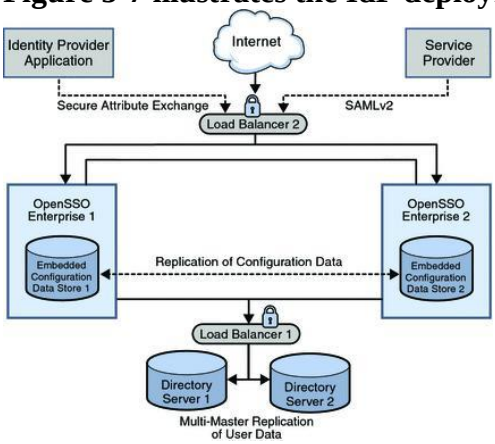
• Integrate with a trusted cloud-based identity management service provider.

Enterprise identity provider

In this architecture, cloud services will delegate authentication to an organization's IdP. In this delegated authentication architecture, the organization federates identities within a trusted circle of CSP domains. A circle of trust can be created with all the domains that are authorized to delegate authentication to the IdP.

In this deployment architecture, where the organization will provide and support an IdP, greater control can be exercised over user identities, attributes, credentials, and policies for authenticating and authorizing users to a cloud service.

Figure 5-7 illustrates the IdP deployment architecture



Here are the specific pros and cons of this approach:

Pros

Organizations can leverage the existing investment in their IAM infrastructure and extend the practices to the cloud. For example, organizations that have implemented SSO for applications within their data center exhibit the following benefits:

• They are consistent with internal policies, processes, and access management frameworks.

• They have direct oversight of the service-level agreement (SLA) and security of the IdP.

• They have an incremental investment in enhancing the existing identity architecture to support federation.

Cons

By not changing the infrastructure to support federation, new inefficiencies can result due to the addition of life cycle management for non-employees such as customers. Most organizations will likely continue to manage employee and long-term contractor identities using organically developed IAM infrastructures and practices. But they seem to prefer to outsource the management of partner and consumer identities to a trusted cloud based identity provider as a service partner.

Identity management-as-a-service

In this architecture, cloud services can delegate authentication to an identity management-asa- service (IDaaS) provider. In this model, organizations outsource the federated identity management technology and user management processes to a third-party service provider,such as Ping Identity, TriCipher's Myonelogin.com, or Symplified.com.

When federating identities to the cloud, organizations may need to manage the identity life cycle using their IAM system and processes. However, the organization might benefit from

anoutsourced multiprotocol federation gateway (identity federation service) if it has to interfacewith many different partners and cloud service federation schemes. For example, as of this writing, Salesforce.com supports SAML 1.1 and Google Apps supports SAML 2.0.

Enterprises accessing Google Apps and Salesforce.com may benefit from a multiprotocol federation gateway hosted by an identity management CSP such as Symplified or TriCipher.

In cases where credentialing is difficult and costly, an enterprise might also outsource credential issuance (and background investigations) to a service provider, such as the GSA Managed Service Organization (MSO) that issues personal identity verification (PIV) cards and, optionally, the certificates on the cards.

The GSA MSO† is offering the USAccess management end-to-end solution as a shared service to federal civilian agencies. In essence, this is a SaaS model for identity management, where the SaaS IdP stores identities in a ―trusted identity store‖ and acts as a proxy for the organization's users accessing cloud services, as illustrated in Figure 5-8.

The identity store in the cloud is kept in sync with the corporate directory through a provider proprietary scheme Once the IdP is established in the cloud, the organization should work with the CSP to delegate authentication to the cloud identity service provider. The cloud IdP will authenticate the cloud users prior to them accessing any cloud services.

Here are the specific pros and cons of this approach:

Pros

Delegating certain authentication use cases to the cloud identity management service hides the complexity of integrating with various CSPs supporting different federation standards. Case in point: Salesforce.com and Google support delegated authentication using SAML.

However, as of this writing, they support two different versions of SAML: Google Apps supports only SAML 2.0, and Salesforce.com supports only SAML 1.1. Cloudbased identity management services that support both SAML standards (multiprotocol federation gateways) can hide this integration complexity from organizations adopting cloud services.

Another benefit is that there is little need for architectural changes to support this model. Once identity synchronization between the organization directory or trusted system of record and the identity service directory in the cloud is set up, users can sign on to cloud

The identity store in the cloud is kept in sync with the corporate directory through a provider proprietary scheme (e.g., agents running on the customer's premises synchronizing a subset of an organization's identity store to the identity store in the cloud using SSL VPNs).

Once the IdP is established in the cloud, the organization should work with the CSP to delegate authentication to the cloud identity service provider.

The cloud IdP will authenticate the cloud users prior to them accessing any cloud services (this is done via browser SSO techniques that involve standard HTTP redirection techniques).

Here are the specific pros and cons of this approach

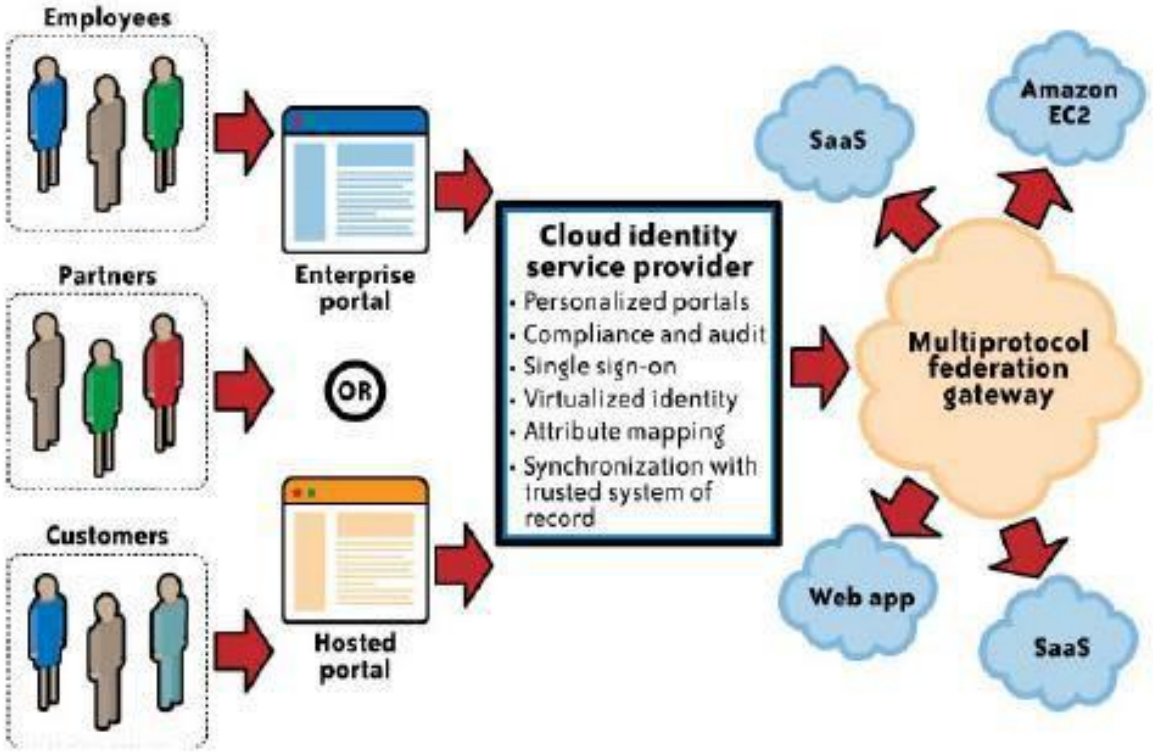


FIGURE 5-8. Identity management-as-a-service (IDaaS):

Pros

Delegating certain authentication use cases to the cloud identity management service hides the complexity of integrating with various CSPs supporting different federation standards. Case in point: Salesforce.com and Google support delegated authentication using SAML. However, as of this writing, they support two different versions of SAML:

Google Apps supports only SAML 2.0, and Salesforce.com supports only SAML 1.1.

Cloudbased identity management services that support both SAML standards (multiprotocol federation gateways) can hide this integration complexity from organizations adopting cloud services.

Another benefit is that there is little need for architectural changes to support this model. Once identity synchronization between the organization directory or trusted system of record and the identity service directory in the cloud is set up, users can sign on to cloud services using corporate identity, credentials (both static and dynamic), and authentication policies.

Cons

When you rely on a third party for an identity management service, you may have less visibility into the service, including implementation and architecture details. Hence, the availability and authentication performance of cloud applications hinges on the identity management service provider's SLA, performance management, and availability.

It is important to understand the provider's service level, architecture, service redundancy,and performance guarantees of the identity management service provider. Another drawback to this approach is that it may not be able to generate custom reports to meet internal compliance requirements.

In addition, identity attribute management can also become complex when identity attributes are not properly defined and associated with identities (e.g., definitions of attributes, both mandatory and optional).

New governance processes may be required to authorize various operations (add/modify/remove attributes) to govern user attributes that move outside the organization's trust boundary. Identity attributes will change through the life cycle of the identity itself and may get out of sync.

Although both approaches enable the identification and authentication of users to cloud services, various features and integration nuances are specific to the service delivery model— SaaS, PaaS, and IaaS—as we will discuss in the next section.

Cloud Authorization Management

Medium-size and large organizations usually have specific requirements for authorization features for their cloud users (i.e., assignment of privileges, or entitlements, to users based on their job functions). In some cases, a business application may require role-based access control (RBAC), in which case authorization is structured to suit the organization's functional role requirements. As of this writing, cloud service authorization enforcement and management capabilities are weak, and when they are available they are very coarse-grained. The services available may not meet your enterprise requirements.

Most cloud services support at least dual roles (privileges): administrator and end user. It is a normal practice among CSPs to provision the administrator role with administrative privileges. These privileges allow administrators to provision and deprovision identities, basic attribute profiles, and, in some cases, to set access control policies such as password strength and trusted networks from which connections are accepted.

As we mentioned earlier, XACML is the preferred standard for expressing and enforcing authorization and user authentication policies. As of this writing, we are not aware of any cloud services supporting XACML to express authorization policies for users.

As much as cloud IAM architecture and practices impact the efficiency of internal IT processes, they also play a major role in managing compliance within the enterprise. Properly implemented IAM practices and processes can help improve the effectiveness of the controls identified by compliance frameworks.

For example, by automating the timely provisioning and deprovisioning of users and entitlements, organizations can reduce the risk of unauthorized users accessing cloud services and meet your privacy and compliance requirements. In addition, identity and attribute management will be key areas of compliance focus for regulatory and privacy issues—proper IAM governance processes should be instituted to address these issues.

IAM practices and processes offer a centralized view of business operations and an automated process that can stop insider threats before they occur. However, given the sparse support for IAM standards such as SAML (federation), SPML (provisioning), and XACML (authorization) by the CSP, you should assess the CSP capabilities on a case-by-case basis and institute processes for managing compliance related to identity (including attribute) and access management.

Cloud Service Provider IAM Practice:

From the CSP's (SaaS, PaaS, or IaaS) perspective, IAM features should be included in the cloud service's design criteria, with the goal of delegating user authentication and authorization to the customer using user management and federation standards. Support for IAM

features has integration implications for both customers (e.g., single sign-on, user provisioning) and CSPs (e.g., billing, accounting resource utilization).

For both the customer and CSP, IAM integration considerations at the early stage of service design will help avoid costly retrofits. Hence, cloud service architects and platform application developers should be embedding IAM features at various stages of the product life cycle—architecture, design, and implementation (e.g., externalize the authentication from the application using the federation feature).

From a cloud customer perspective, the application's IAM capabilities (or lack thereof), such as identity federation, will impact the cloud service governance, integration, and user experience (e.g., barriers to adopt the cloud service). Hence, architects, designers, and developers of cloud applications should understand the IAM requirements of cloud applications and bake the features into the RFP or CSP evaluation criteria.

Enterprise IAM requirements include:
• Provisioning of cloud service accounts to users, including administrators.
• Provisioning of cloud services for service-to-service integration
• SSO support for users based on federation standards (e.g., SAML support).
• Support for internal- and regulatory-policy compliance requirements, including segregation of duties using RBAC, rules, or claims-based authentication methodology.

RBAC features promote a least-privilege-based access model where a user is granted the right number of privileges required to perform the job. Claims-based methodology enables some important privacy use cases because it allows for only the user's entitlements, not her actual identity, to flow with messages, which allows for fine-grained authorization without the requirement to actually embed the user's identity into messages.
• User activity monitoring, logging, and reporting dictated by internal policies and regulatory compliance, such as SOX, PCI, and HIPAA.

16. Brief explanation about SaaS.

SaaS

One of the primary concerns of IT and business decision makers regarding software-as-aservice applications is security management. Although most SaaS vendors have been able to demonstrate that their cloud-based applications are secure from an operational point of view, there are still access control issues that organizations need to address to ensure that their corporate data is fully secure from a corporate policies and procedures standpoint.

It is becoming particularly important to address these issues because SaaS applications aregaining popularity due to their low barrier to adoption and their pay-as-you-go service model. In some cases, business units are sidestepping IT and directly engaging with SaaS vendors, which can lead to additional IT headaches.

IT must manage risks that may come out of this loss of visibility and control, and be able to ensure that the right users have the right level of access to information hosted by SaaS vendors.
Organizations considering integrating into SaaS services should consider two major challenges for identity management:

• Is the organization ready to provision and manage the user life cycle by extending itsestablished IAM practice to the SaaS service?
• Are the SaaS provider capabilities sufficient to automate user provisioning and life cycle management without implementing a custom solution for the SaaS service?

Customer responsibilities

In SaaS services, customers have limited responsibility and available controls to secure information. In general, SaaS solutions are multitenant and are delivered to the customer via a web browser.

The only controls that are available to the customer are IAM controls such as identity provisioning, authentication policies (e.g., password strength), profile configuration, and basic authorization policies that manifest as user profiles. The following are the responsibilities of customers from an IAM perspective:

✓ ***User provisioning***

User provisioning methods are typically unique to the SaaS provider. Customers need to understand the preferred method, lag time to activate users, and user attributes that are supported by the SaaS service. Most often the provisioning process is manual and may involve uploading spreadsheets or documents in XML format.

Almost all SaaS providers support bulk upload of user identities, as that's the most common use case for provisioning users. Some SaaS providers may support just-in-time provisioning where user identities are created on the fly using a provisioning request (sometimes SPML-employed) that is usually triggered by user activity such as the user clicking on a hyperlink that is unique to the user identity.

✓ ***Profile management***

As part of the provisioning process, customers may have the ability to create user profiles that play a role in user authorization. User profiles such as *user* and *manager* are an approach to assigning entitlements to users within the SaaS application. Admittedly, these are not sophisticated features and will require customers to understand the flexibility and management of the profiles.

✓ ***SaaS IAM capability evaluation***

Customers are responsible for evaluating the support for IAM features such as SSO (using identity federation) by CSPs. SAML is the de facto standard for federating identities and is now supported by large SaaS providers (among them Google and Salesforce.com). However, not all providers are supporting SAML 2.0, and some may support only SAML 1.1.

For example, Salesforce.com supports SAML 1.1 while Google Apps supports SAML 2.0. Hence, it is important to understand what federation protocols are supported by which providers and the integration requirements to federate and support SSO.

✓ ***Investigation support***

Logs and audit trails are also often needed to investigate incidents. For example, PCI DSS requires the provider to ―provide for timely forensic investigation‖ if the service provider suffers a breach. Since the SaaS provider's logs are internal and are not necessarily accessible externally or by customers, monitoring (let alone investigation) is difficult. Since access to logs is required for PCI compliance and may be requested by auditors and regulators, make sure to negotiate access to the provider's logs as part of any service agreement.

✓ ***Compliance management***

Although the same security concerns companies already have within their own networks—securing the network, hardware, applications, and data—apply for companies outsourcing their data with SaaS, trust and transparency exacerbate the situation in cloud computing. When compliance with government regulations such as SOX, the Gramm- Leach-Bliley Act (GLBA), and HIPAA and with industry standards such as PCI DSS come into the scope of the data hosted in SaaS, it could be challenging to meet those demands.

In general, customers of SaaS services are responsible for compliance management,although the provider hosts the data. Make an effort to understand the access control, logging, reporting, and auditing capabilities offered by SaaS providers and assess whether those controls are adequate to meet compliance management requirements.

CSP responsibilities

With regard to IAM, some responsibilities belong to the CSP and some belong to the customer. Here are CSP responsibilities:

- ✓ ***Authentication services***

Unless the SaaS provider supports delegated authentication via federation, it typically authenticates the SaaS users—usually via a web form delivered over HTTPS—using a user identifier and static password. Since users can be accessing the service from anywhere on the Internet, it is up to the SaaS provider to authenticate users based on the network trust level.

For example, some CSPs can preregister the IP address or IP range of a user's location (home, office, etc.) to protect data from hackers who are stealing the user's identity and credentials using keystroke loggers (potentially installed on the user's computer in a stealthy manner). Given that authentication activity is a precursor to the actual use of the SaaS service, it is critical for the CSP to deliver and maintain a continuously available authentication service.

- ✓ ***Account management policies***

CSPs should communicate the account management policies including account lock-outs (after many login failures), account provisioning methods, and privilege account management roles.

- ✓ ***Federation***

CSPs supporting identity federation using standards such as SAML should publish the information necessary for customers to take advantage of this feature and enable SSO for their users. Such information includes the version (SAML 1.1, SAML 2.0), a use case implementation example, and implementation details of the federation using the API (e.g., support for SAML using REST and SOAP).

17. Brief explanation about PaaS

Organizations considering extending their established IAM practices to PaaS cloud providers have few options at their disposal. PaaS CSPs typically delegate authentication functions using federation to the PaaS provider's IdP (e.g., the Google App Engine delegates authentication to Google's authentication service).

In some cases, such as Salesforce.com's Force.com, there is limited support for delegated authentication and it is usually performed without the aid of SAML assertions (e.g., it is proprietary to each PaaS provider implementation). CSPs also supply software components that can be invoked using programming languages (usually PaaSspecific) to perform authentication and limited authorization.

Microsoft recently introduced the ―Geneva‖ Claims-Based Access Platform that is SAML 2.0 compliant (and is still in beta as of this writing). The project's goal is to help developers

externalize authentication, authorization, and personalization from .NET applications, and help organizations federate users using Microsoft's federation offering, Security Token Service (STS).

Microsoft developers could potentially utilize STS deployed in an enterprise, and interoperate with applications that are deployed on an Azure platform. This solution is specifically targeted toward Microsoft customers who are interested in extending their Active Directory directory into a cloud by way of federation.

Therefore, it is not apparent whether the Geneva Claims-Based Access Platform will interoperate with existing SaaS and PaaS providers who are SAML 2.0 compliant.

18. Brief explanation about IaaS

Enterprises considering extending their established IAM practices to IaaS cloud providers (computing and storage) have limited or no options. Because IaaS providers provide computing or storage-as-a-service, they do not have visibility to applications that are hosted on the IaaS platform. Almost all IaaS providers use Secure Shell (SSH) to log on and administer users and credentials; few providers, such as Amazon Web Services (AWS) EC2, offer a web console to provision users, manage user keys, and assign users to security groups that relate to the administrative functions of IaaS.

Some of the responsibilities and challenges in managing users in IaaS services are:

- ✓ ***User provisioning***

Provisioning of users (developers, administrators) on IaaS systems that are dynamic in nature. Given that hundreds of systems are provisioned for workload management, user provisioning will have to be automated at the time of image creation and should be policybased. Ideally, systems should rely on corporate directories (LDAP, Active Directory) for user management to avoid duplication of identities on systems. However, the virtual network topology in cloud and network security policies may interfere with directorybased authentication schemes and should be assessed on a per-CSP basis.

- ✓ ***Privileged user management***

Managing private keys of system administrators and protecting the keys when system administrators leave the company (e.g., SSH host keys).

- ✓ ***Customer key assignment***

Assigning IDs and keys required to access the service. These keys are used for managing access to customer accounts for billing reasons, as well as for authenticating customers to their services. For example, Amazon assigns an Access Key ID, a Secret Access Key, an X.509 certificate, and a corresponding private key to every EC2 customer. You authenticate to an AWS request using either the Access Key ID and Secret Access Key or the X.509 certificate and private key. Hence, customers are responsible for provisioning and safeguarding these keys.

- ✓ ***Developer user management***

Provisioning of developers and testers to IaaS instances, and deprovisioning the same when access is no longer required.

- ✓ ***End user management***

Provisioning users who need access to applications hosted on IaaS. Currently, there is no automated way to synchronize an organizational LDAP or Active Directory directory with IaaS providers to avoid a redundant user database at each of the IaaS clouds. Some third-party identity management service providers claim to have developed adapters for EC2 user provisioning and management, however

What Is the Data Life Cycle?

- ✓ Personal information should be managed as part of the data used by the organization. It should be managed from the time the information is conceived through to its final disposition.
- ✓ Protection of personal information should consider the impact of the cloud on each of the following phases as detailed in Figure 7-1.

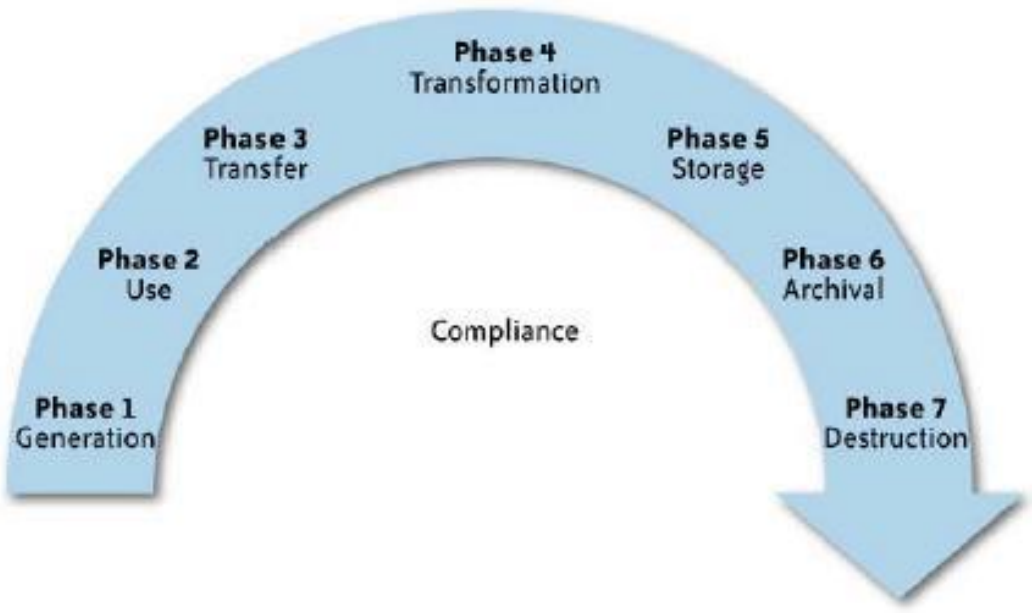


FIGURE 7-1. KPMG data life cycle

The components within each of these phases are:

Generation of the information

- ✓ ***Ownership***: Who in the organization owns PII, and how is the ownership maintained if the organization uses cloud computing?
- ✓ ***Classification*:** How and when is PII classified? Are there limitations on the use of cloud computing for specific data classes?
- ✓ ***Governance*:** Is there a governance structure to ensure that PII is managed and protected through its life cycle, even when it is stored or processed in a cloud computing environment?

Use

- ✓ ***Internal versus external***: Is PII used only within the collecting organization, or is it used outside the organization (e.g., in a public cloud)?
- ✓ ***Third party***: Is the information shared with third parties (e.g., subcontractors or CSPs)?

- ***Appropriateness*:** Is the use of the information consistent with the purpose for which it was collected? Is the use within the cloud appropriate based on the commitments the organization made to the data ubjects?
- ***Discovery/subpoena***: Is the information managed in the cloud in a way that will enable the organization to comply with legal requirements in case of legal proceedings?

Transfer

- ***Public versus private networks***: When information is transferred to a cloud is the organization using public networks, and is it protected appropriately? (PII should always be protected to address the risk level and legal requirements.)
- ***Encryption requirements***: Is the PII encrypted? Some laws require that PII will be encrypted when transmitted via a public network (and this will be the case when the organization is using a public cloud).
- ***Access control***: Are there appropriate access controls over PII when it is in the cloud?

Transformation

- ***Derivation*:** Are the original protection and use limitations maintained when data is transformed or further processed in the cloud?
- ***Aggregation*:** Is data in the cloud aggregated so that it is no longer related to an identifiable individual (and hence is no longer considered PII)?
- ***Integrity***: Is the integrity of PII maintained when it is in the cloud?

Storage

- ***Access control***: Are there appropriate controls over access to PII when stored in the cloud so that only individuals with a need to know will be able to access it?
- ***Structured versus unstructured***: How is the data stored to enable the organization to access and manage the data in the future?
- ***Integrity/availability/confidentiality*:** How are data integrity, availability, and confidentiality maintained in the cloud?
- ***Encryption*:** Several laws and regulations require that certain types of PII should be stored only when encrypted. Is this requirement supported by the CSP?

Archival

- ***Legal and compliance***: PII may have specific requirements that dictate how long it should be stored and archived. Are these requirements supported by the CSP?
- ***Off-site considerations***: Does the CSP provide the ability for long-term off-site storage that supports archival requirements?
- ***Media concerns***: Is the information stored on media that will be accessible in the future? Is the information stored on portable media that may be more susceptible to loss? Who controls the media and what is the organization's ability to recover such media from the CSP if needed?
- ***Retention*:** For how long will the data be retained by the CSP? Is the retention period consistent with the organization's retention period?

Destructio*n*

- ***Secure***: Does the CSP destroy PII obtained by customers in a secure manner to avoid potential breach of the information?
- ***Complete*:** Is the information completely destroyed? Does the destruction completely erase the data, or can it be recovered?

The impact differs based on the specific cloud model used by the organization, the phase (Figure 7-1, shown earlier) of personal information in the cloud, and the nature of the organization.

19. What Are the Key Privacy Concerns in the Cloud?and give the concept in detail

Privacy advocates have raised many concerns about cloud computing. These concerns typically mix security and privacy. Here are some additional considerations to be aware of:

- ✓ ***Access***

Data subjects have a right to know what personal information is held and, in some cases, can make a request to stop processing it. This is especially important with regard to marketing activities; in some jurisdictions, marketing activities are subject to additional regulations and are almost always addressed in the end user privacy policy for applicable organizations.

In the cloud, the main concern is the organization's ability to provide the individual with access to all personal information, and to comply with stated requests. If a data subject exercises this right to ask the organization to delete his data, will it be possible to ensure that all of his information has been deleted in the cloud?

- ✓ ***Compliance***

What are the privacy compliance requirements in the cloud? What are the applicable laws, regulations, standards, and contractual commitments that govern this information, and who is responsible for maintaining the compliance? How are existing privacy compliance requirements impacted by the move to the cloud? Clouds can cross multiple jurisdictions; for ***example***, data may be stored in multiple countries, or in multiple states within the United States.

- ✓ ***Storage***

Where is the data in the cloud stored? Was it transferred to another data center in another country? Is it commingled with information from other organizations that use the same CSP? Privacy laws in various countries place limitations on the ability of organizations to transfer some types of personal information to other countries.

When the data is stored in the cloud, such a transfer may occur without the knowledge of the organization, resulting in a potential violation of the local law.

- ✓ ***Retention***

How long is personal information (that is transferred to the cloud) retained? Which retention policy governs the data? Does the organization own the data, or the CSP? Who enforces the retention policy in the cloud, and how are exceptions to this policy (such as litigation holds) managed?

- ✓ ***Destruction***

How does the cloud provider destroy PII at the end of the retention period? How do organizations ensure that their PII is destroyed by the CSP at the right point and is not available to other cloud users? How do they know that the CSP didn't retain additional copies? Cloud storage providers usually replicate the data across multiple systems and sites—increased availability is one of the benefits they provide.

This benefit turns into a challenge when the organization tries to destroy the data—can you truly destroy information once it is in the cloud? Did the CSP really destroy the data, or just make it inaccessible to the organization? Is the CSP keeping the information longer than necessary so that it can mine the data for its own use?

- ✓ ***Audit and monitoring***

How can organizations monitor their CSP and provide assurance to relevant stakeholders that privacy requirements are met when their PII is in the cloud?

- ✓ ***Privacy breaches***

How do you know that a breach has occurred, how do you ensure that the CSP notifies you when a breach occurs, and who is responsible for managing the breach notification process (and costs associated with the process)?

If contracts include liability for breaches resulting from negligence of the CSP, how is the contract enforced and how is it determined who is at fault? Many of these concerns are not specific to personal information, but to all types of information and a broader set of compliance requirements.

Who Is Responsible for Protecting Privacy?

There are conflicting opinions regarding who is responsible for security and privacy. Some publications assign it to providers; but although it may be possible to transfer liability via contractual agreements, it is never possible to transfer accountability.

Ultimately, in the eyes of the public and the law, the onus for data security and privacy falls on the organization that collected the information in the first place—the user organization. This is true even if the user organization has no technical capability to ensure that the contractual requirements with the CSP are met.

History and experience have proven that data breaches have a cascading effect. When an organization loses control of users' personal information, the users are responsible (directly or indirectly) for subsequent damages resulting from the loss. Identity theft is only one of the possible effects; others may include invasion of privacy or unwelcome solicitation.

When an affected individual is dealing with the fallout, he will likely blame the one who made the decision to use the service, as opposed to the provider of the service. Full reliance on a third party to protect personal data is irresponsible and will inevitably lead to negative consequences.

Responsible data stewardship requires an in-depth understanding of the technology underlying cloud computing and the legal requirements and implications. As such, a cross functional team is critical to adequately maintain security and privacy. The accountability model is similar to discussions around privacy in outsourcing or subcontracting relationships, and the conclusion is similar:

- Organizations can transfer liability, but not accountability.
- Risk assessment and mitigation throughout the data life cycle is critical.
- Knowledge about legal obligations and contractual agreements or commitments is imperative.

Important questions

PART-A

1. What is the Authorization for Access Control?
2. How can we classify the authority?
3. List the Three Authorization Models?
4. List the four distinct functions?
5. What is Transport-Level Security?
6. What are the message-level securities for message protection for SOAP messages provided by GSI?

7. What are the four primary pieces of information the certificate includes in GSI authentication?
8. What are the four significant risk factors Network Level infrastructure security?
9. What does Denial-of-Service Attack (DoS) mean?
10. What are the basic types of DoS attack ?
11. What are the problems caused by the DOS attacks?
12. What is Distributed Denial-of –service (DDoS)?
13. Lists security controls at the network level?
14. What Is Privacy?
15. What are the Enterprise IAM requirement?
16. List the user management function in the cloud?
17. List the new host security threats in public IaaS?
18. What is Clearing?
19. What is mean by Sanitization?
20. What Are the Key Privacy Concerns in the Cloud?and give the concept in detail
21. What Is the Data Life Cycle?
22. Define Cloud Identity Administration
23. List the four key components of the IAM automation process:
24. Give The IAM processes to support the business can be broadly categorized as follows:

PART-B

1. Replacing the Established Model of Network Zones and Tiers with Domains?
2. Explain in detail Infrastructure Security and its types?
3. Explain the GSI Functional Layers with neat diagram?
4. Explain in detail the Authentication and Authorization Methods?
5. Write short notes on a Fuzzy-Trust Model ?
6. Write short notes on Reputation-Based Trust Model ?
7. Write short notes on Generalized Trust Model?
8. Explain in detail the Trust Models for Grid Security Enforcement?
9. Explain in detail Application Level Infrastructure Security?
10. Brief explanation about Application-Level Security Threats in cloud infra satructure
11. Brief explanation about Aspects of Data Security in cloud infra structure
12. Explain in detail about Identity and Access Management
13. Explain in detail about IAM Practices in the Cloud
14. Brief explanation about SaaS.
15. Brief explanation about PaaS
16. Brief explanation about IaaS
17. What Are the Key Privacy Concerns in the Cloud?and give the concept in detail