

UNIT- I

CLOUD ARCHITECTURE AND MODEL

⇒ SCALABLE COMPUTING OVER THE INTERNET:

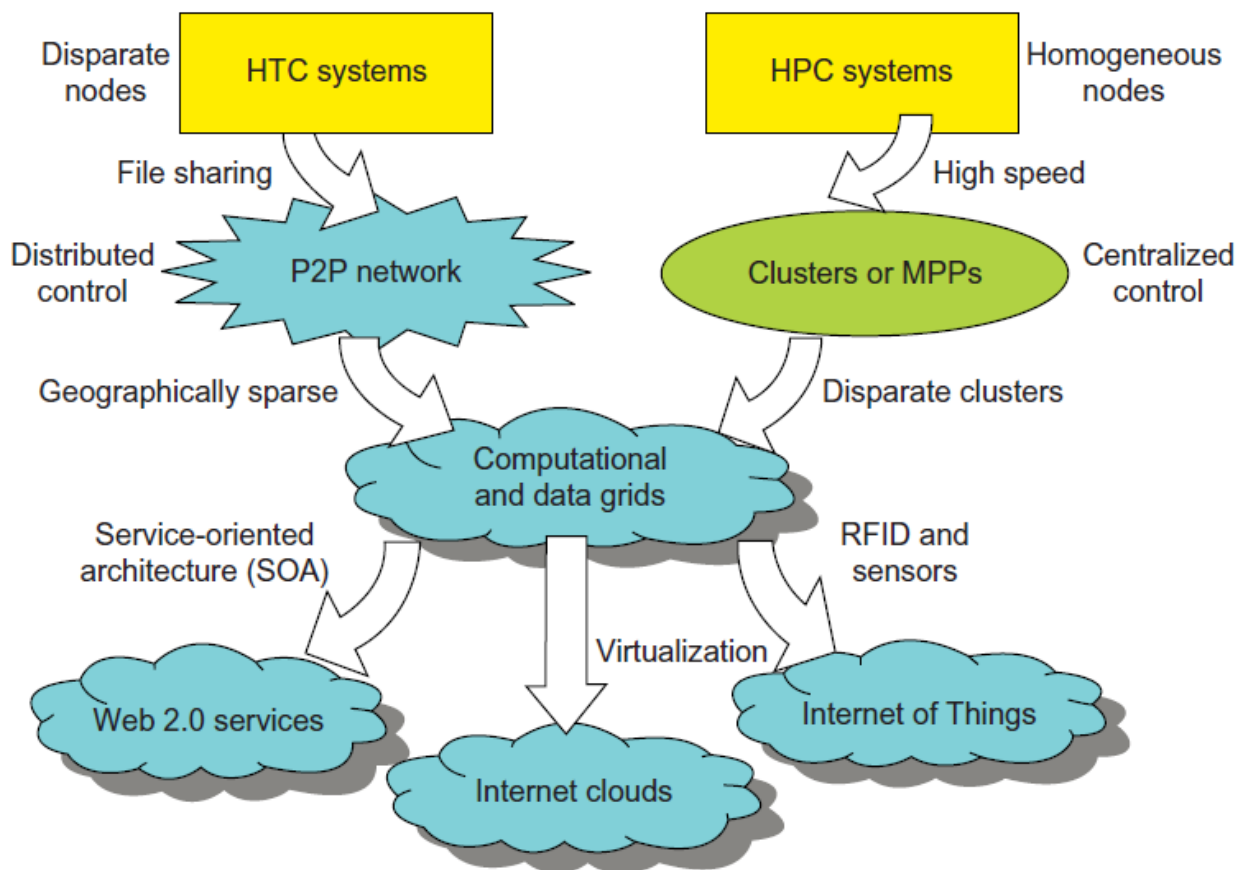
Over the past 60 years, computing technology has undergone a series of platform and environment changes. Instead of using a centralized computer to solve computational problems, a parallel and distributed computing system uses multiple computers to solve large-scale problems over the Internet. Thus, distributed computing becomes data-intensive and network-centric. These large-scale Internet applications have significantly enhanced the quality of life and information services in society today.

➤ **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 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.

➤ **The Platform Evolution**

- 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. Figure 1.1 illustrates the evolution of HPC and HTC systems.
- 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 clients. 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.



➤ High-Performance Computing

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.

➤ High-Throughput Computing

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.

➤ Three New Computing Paradigms

As Figure 1.1 illustrates, with the introduction of SOA, Web 2.0 services become available. Advances in virtualization make it possible to see the growth of Internet clouds as a new computing paradigm. The maturity of radio-frequency identification (RFID), Global Positioning System (GPS), and sensor technologies has triggered the development of the Internet of Things (IoT).

➤ 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** This 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. Some authors refer to this discipline as parallel processing. 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. Some authors consider cloud computing to be a form of utility computing or service computing.

➤ **Distributed System Families**

Technologies for building P2P networks and networks of clusters have been consolidated into many national projects designed to establish wide area computing infrastructures, known as computational grids or data grids.

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. Meeting these goals requires to yield the following design objectives:

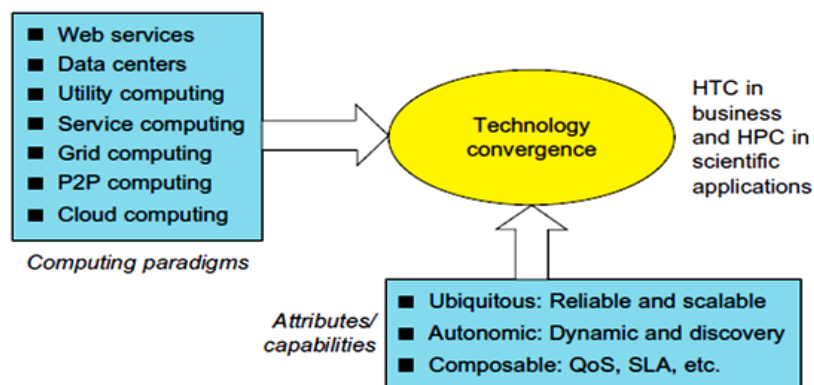
- ▶ **Efficiency** measures the utilization rate of resources in an execution model by exploiting massive parallelism in HPC. For HTC, efficiency is more closely related to job throughput, data access, storage, and power efficiency.
- ▶ **Dependability** measures the reliability and self-management from 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.

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.

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.

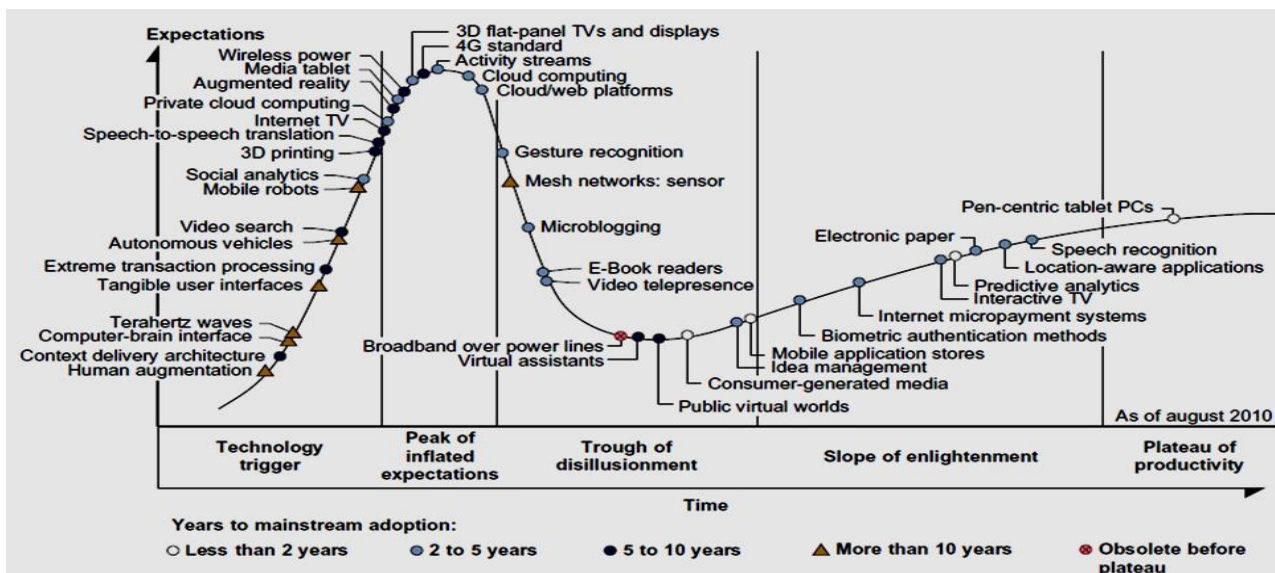
➤ The Trend toward Utility Computing

Figure identifies major computing paradigms to facilitate the study of distributed systems and their applications. 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.



➤ The Hype Cycle of New Technologies

Any new and emerging computing and information technology may go through a hype 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.



➤ Cyber-Physical Systems

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.

⇒ TECHNOLOGIES FOR NETWORK-BASED SYSTEMS

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.

➤ Multicore CPUs and Multithreading Technologies

Consider the growth of component and network technologies over the past 30 years. They are crucial to the development of HPC and HTC systems. The 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.

➤ 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.

➤ Multithreading Technology

Consider in **Figure** 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.

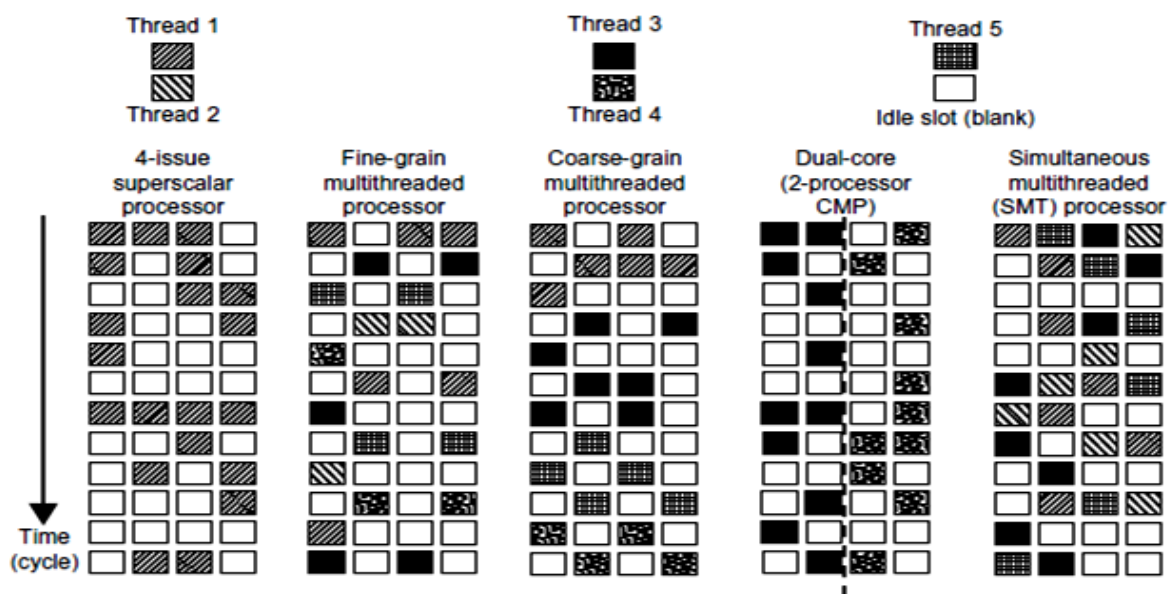
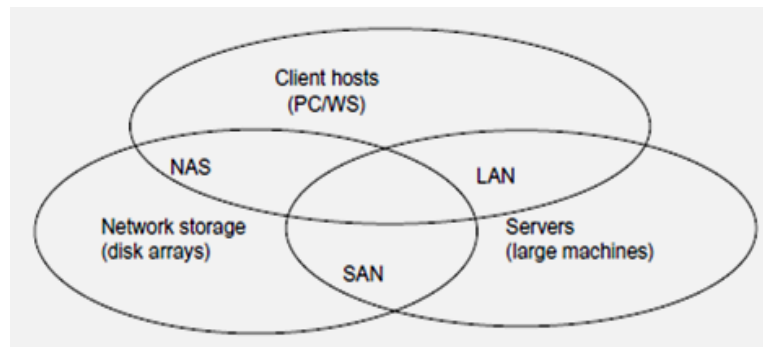


FIGURE 1.6

Five micro-architectures in modern CPU processors, that exploit ILP and TLP supported by multicore and multithreading technologies.

➤ Memory, Storage, and Wide-Area Networking

The growth of DRAM chip capacity from 16 KB in 1976 to 64 GB in 2011. For hard drives, capacity increased from 260 MB in 1981 to 250 GB in 2004. The disks or disk arrays have exceeded 3 TB in capacity. 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.



➤ Virtual Machines

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). Virtual machines (VMs) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines.

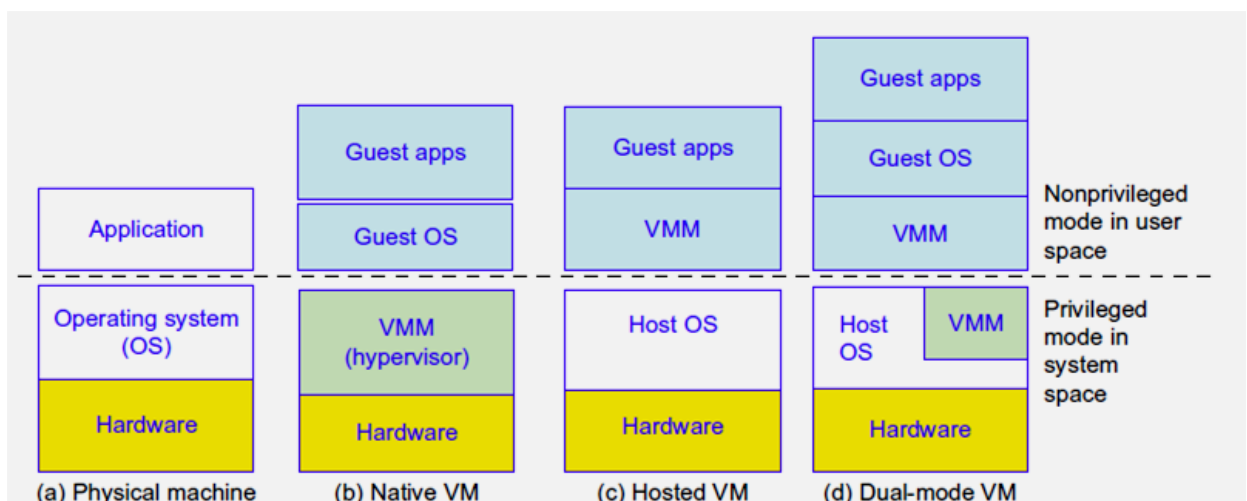


FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

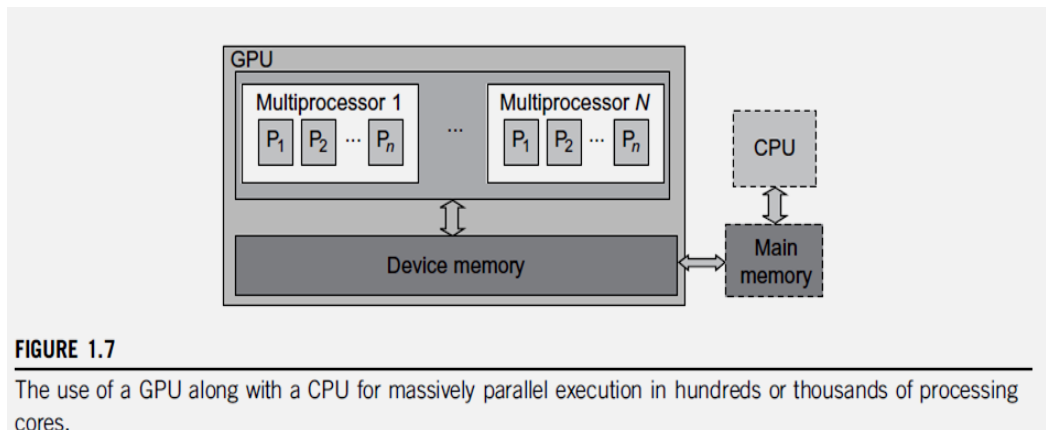
➤ GPU Computing to Exascale and Beyond

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.

➤ GPU Programming Model

Figure 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.



➤ Data Center Virtualization for Cloud Computing

- 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.
- 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.
- 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.

⇒ SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING

- Distributed and cloud computing systems are built over a large number of autonomous computer nodes. These node machines are interconnected by SANs, LANs, or WANs in a hierarchical manner. With today's networking technology, a few LAN switches can easily connect hundreds of machines as a working cluster. A WAN can connect many local clusters to form a very large cluster of clusters. In this sense, one can build a massive system with millions of computers connected to edge networks.

Functionality, Applications	Computer Clusters [10,28,38]	Peer-to-Peer Networks [34,46]	Data/ Computational Grids [6,18,51]	Cloud Platforms [1,9,11,12,30]
Architecture, Network Connectivity, and Size	Network of compute nodes interconnected by SAN, LAN, or WAN hierarchically	Flexible network of client machines logically connected by an overlay network	Heterogeneous clusters interconnected by high-speed network links over selected resource sites	Virtualized cluster of servers over data centers via SLA
Control and Resources Management	Homogeneous nodes with distributed control, running UNIX or Linux	Autonomous client nodes, free in and out, with self-organization	Centralized control, server-oriented with authenticated security	Dynamic resource provisioning of servers, storage, and networks
Applications and Network-centric Services	High-performance computing, search engines, and web services, etc.	Most appealing to business file sharing, content delivery, and social networking	Distributed supercomputing, global problem solving, and data center services	Upgraded web search, utility computing, and outsourced computing services
Representative Operational Systems	Google search engine, SunBlade, IBM Road Runner, Cray XT4, etc.	Gnutella, eMule, BitTorrent, Napster, KaZaA, Skype, JXTA	TeraGrid, GriPhyN, UK EGEE, D-Grid, ChinaGrid, etc.	Google App Engine, IBM Bluecloud, AWS, and Microsoft Azure

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

- **Clusters of Cooperative Computers**

A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource.

- **Cluster Architecture**

Figure shows the architecture of a typical server cluster built around a low-latency, high bandwidth 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.

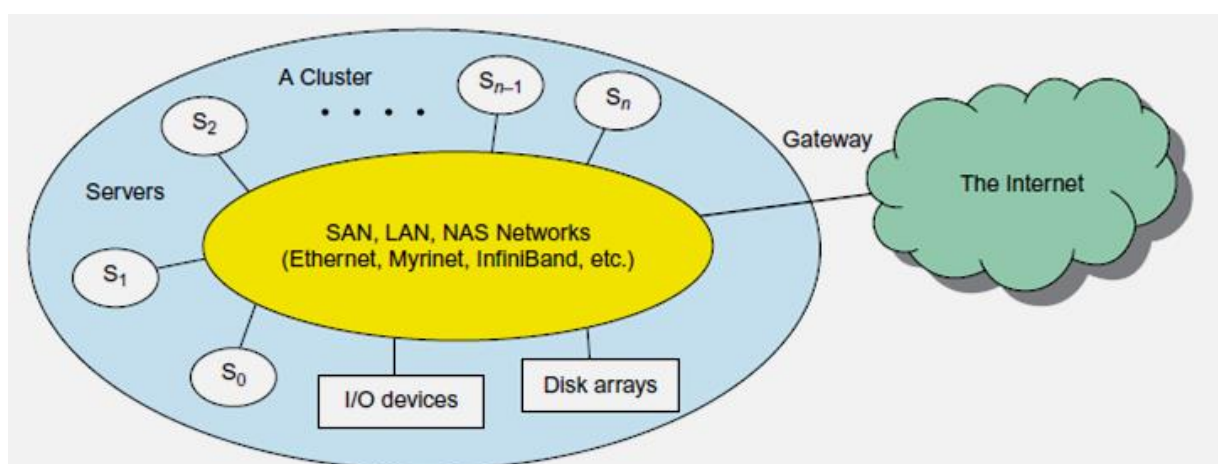


FIGURE 1.15

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**

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.

A cluster-wide OS for complete resource sharing is not available yet. 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.

- **Grid Computing Infrastructures**

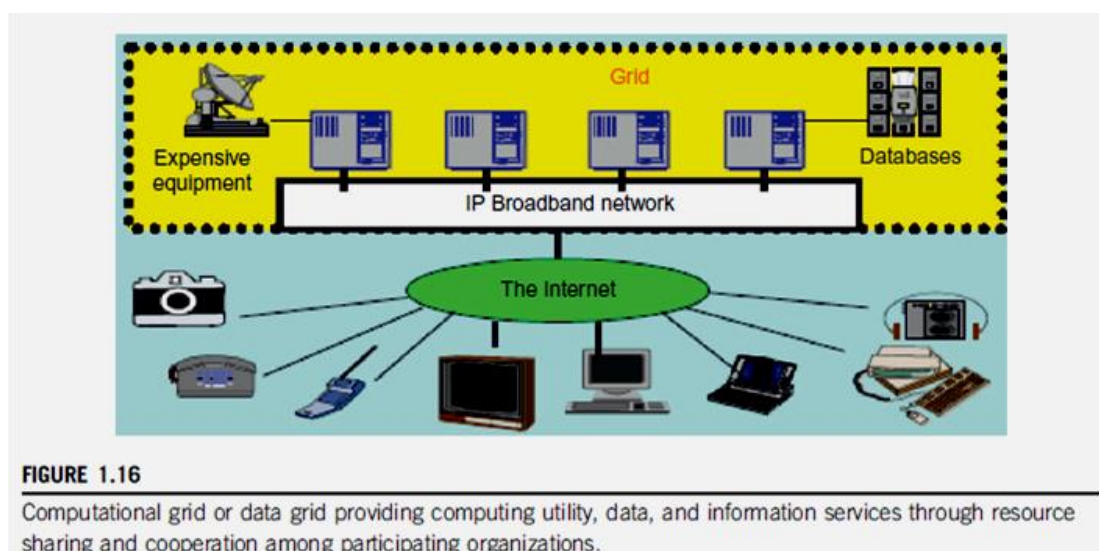
In the past 30 years, users have experienced a natural growth path from Internet to web and grid computing services. Internet services such as the Telnet command enables a local computer to connect to a remote computer. A web service such as HTTP enables remote access of remote web pages. Grid computing is envisioned to allow close interaction among applications running on distant computers simultaneously.

- **Computational Grids**

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.

- **Grid Families**

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 below Table, 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.



- **Peer-to-Peer Network Families**

An example of a well-established distributed system is the client-server architecture. The P2P architecture offers a distributed model of networked systems. First, a P2P network is client-oriented instead of server-oriented. In a P2P system, every node acts as both a client and a server, providing part of the system resources. Peer machines are simply client computers connected to the Internet.

Figure 1.17 shows the architecture of a P2P network at two abstraction levels. Each peer machine joins or leaves the P2P network voluntarily. Only the participating peers form the physical network at any time. Data items or files are distributed in the participating peers. Based on communication or file-sharing needs, the peer IDs form an overlay network at the logical level. This overlay is a virtual network formed by mapping each physical machine with its ID, logically, through a virtual mapping as shown in Figure 1.17.

There are two types of overlay networks: unstructured and structured. An unstructured overlay network is characterized by a random graph. There is no fixed route to send messages or files among the nodes. Structured overlay networks follow certain connectivity topology and rules for inserting and removing nodes (peer IDs) from the overlay graph.

Based on application, P2P networks are classified into four groups, as shown in Table 1.5.

System Features	Distributed File Sharing	Collaborative Platform	Distributed P2P Computing	P2P Platform
Attractive Applications	Content distribution of MP3 music, video, open software, etc.	Instant messaging, collaborative design and gaming	Scientific exploration and social networking	Open networks for public resources
Operational Problems	Loose security and serious online copyright violations	Lack of trust, disturbed by spam, privacy, and peer collusion	Security holes, selfish partners, and peer collusion	Lack of standards or protection protocols
Example Systems	Gnutella, Napster, eMule, BitTorrent, Aimster, KaZaA, etc.	ICQ, AIM, Groove, Magi, Multiplayer Games, Skype, etc.	SETI@home, Geonome@home, etc.	JXTA, .NET, FightingAid@home, etc.

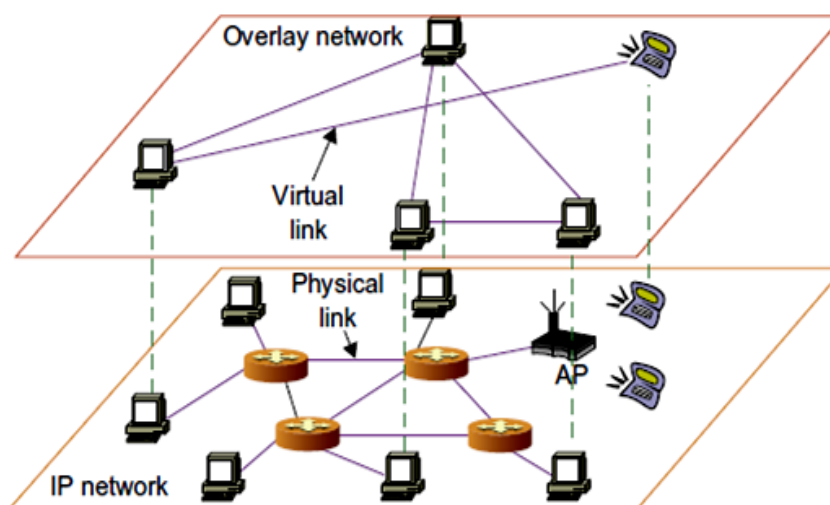


FIGURE 1.17

The structure of a P2P system by mapping a physical IP network to an overlay network built with virtual links.

- **Cloud Computing over the Internet**

In the future, working with large data sets will typically mean sending the computations (programs) to the data, rather than copying the data to the workstations. This reflects the trend in IT of moving computing and data from desktops to large data centers, where there is on-demand provision of

software, hardware, and data as a service. This data explosion has promoted the idea of cloud computing.

“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.”

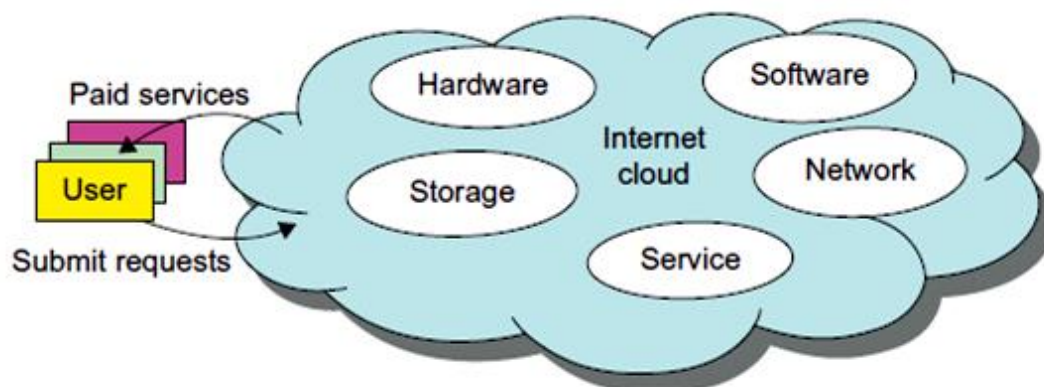


FIGURE Virtualized resources from data centers to form an Internet cloud, provisioned with hardware, software, storage, network, and services for paid users to run their applications.

The Cloud Landscape: Figure 1.19 depicts the cloud landscape and major cloud players, based on three cloud service models.

- **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.
- **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.

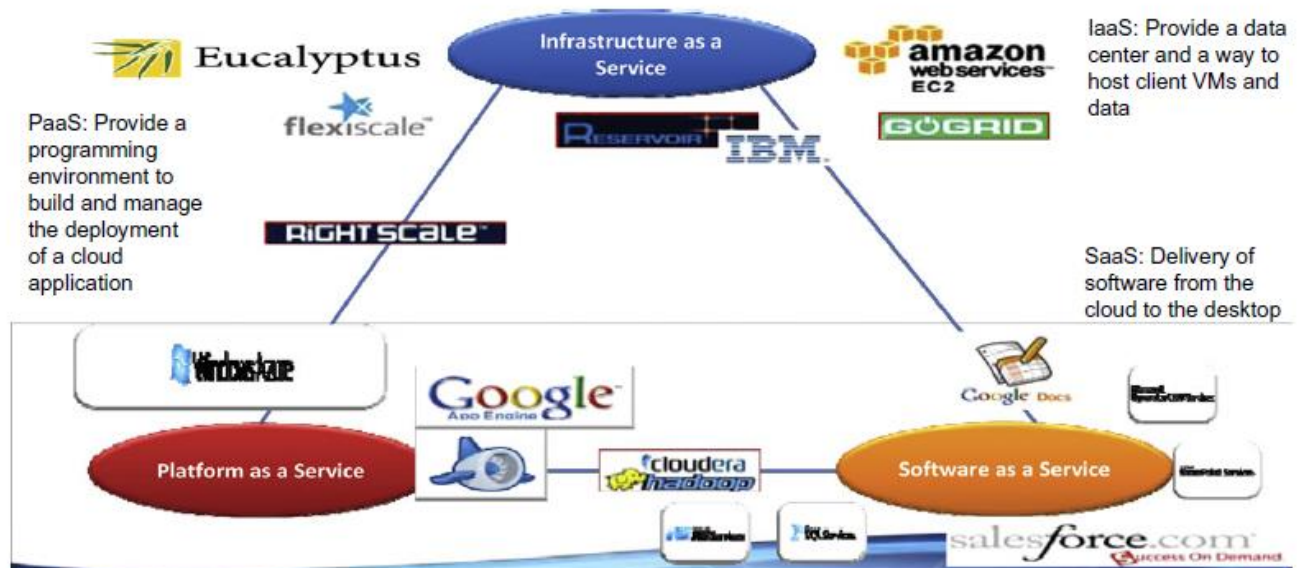


FIGURE 1.19

Three cloud service models in a cloud landscape of major providers.

⇒ **NIST CLOUD COMPUTING REFERENCE ARCHITECTURE**

- ✓ **The National Institute of Standards and Technology (NIST)** has been designated by the Federal Chief Information Officer (CIO) to accelerate the federal government's secure adoption of cloud computing by leading efforts to identify existing standards and guidelines.
- ✓ Consistent with NIST's mission, the NIST Cloud Computing Program has developed a USG Cloud Computing Technology Roadmap, as one of many mechanisms in support of United States Government (USG) secure and effective adoption of the Cloud Computing model² to reduce costs and improve services.
- ✓ The NIST Cloud Computing Standards Roadmap Working Group has surveyed the existing standards landscape for interoperability, performance, portability, security, and accessibility standards/models/studies/use cases/conformity assessment programs, etc., relevant to cloud computing.
- ✓ The NIST Definition of Cloud Computing identified cloud computing as a model for enabling ubiquitous, 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.

Using the taxonomy developed by the NIST Cloud Computing Reference Architecture and Taxonomy Working Group, cloud computing relevant standards have been mapped to the requirements of accessibility, interoperability, performance, portability, and security.

Present areas with standardization gaps include: SaaS (Software as a Service) functional interfaces; SaaS self-service management interfaces; PaaS (Platform as a Service) functional interfaces; business support / provisioning / configuration; security; and privacy. Present standardization areas of priority to the federal government include: security auditing and compliance; identity and access management; SaaS application specific data and metadata; and resource description and discovery.

At the beginning of 2011, NIST created the following public working groups in order to provide a technically oriented strategy and standards-based guidance for the federal cloud computing implementation effort:

- Cloud Computing Reference Architecture and Taxonomy Working Group
- Cloud Computing Standards Acceleration to Jumpstart Adoption of Cloud Computing
- (SAJACC) Working Group Cloud Computing Security Working Group
- Cloud Computing Standards Roadmap Working Group
- Cloud Computing Target Business Use Cases Working Group

➤ **THE NIST DEFINITION OF CLOUD COMPUTING**

“Cloud computing is a model for enabling ubiquitous, 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.” This cloud model is composed of **five essential characteristics, three service models, and four deployment models.**

▶ **Essential Characteristics:**

- **On-demand self-service** - A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
- **Broad network access** - Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- **Resource pooling** - The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.
- **Rapid elasticity** - Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
- **Measured service** - Cloud systems automatically control and optimize resource use by leveraging a metering capability¹⁰ at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, active user accounts). Resource usage can be monitored, controlled, audited, and reported, providing transparency for both the provider and consumer of the utilized service.

▶ **Service Models:**

- **Software as a Service (SaaS).**

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email) or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited userspecific application configuration settings.

- **Platform as a Service (PaaS).**

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. 12 The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

- **Infrastructure as a Service (IaaS).**

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

- ▶ **Deployment Models:**

- **Private cloud.**

The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

- **Community cloud.**

The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

- **Public cloud.**

The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

- **Hybrid cloud.**

The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

➤ CLOUD COMPUTING REFERENCE ARCHITECTURE

The NIST cloud computing reference architecture is a generic high-level conceptual model that is a powerful tool for discussing the requirements, structures, and operations of cloud computing. It defines a set of actors, activities, and functions that can be used in the process of developing cloud computing architectures, and relates to a companion cloud computing taxonomy.

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

For example, a **Cloud Consumer** is an individual or organization that acquires and uses cloud products and services. The purveyor of products and services is the **Cloud Provider**. Because of the possible service offerings (Software, Platform or Infrastructure) allowed for by the cloud provider, there will be a shift in the level of responsibilities for some aspects of the scope of control, security and configuration.

The **Cloud Broker** acts as the intermediary between consumer and provider and will help consumers through the complexity of cloud service offerings and may also create value-added cloud services. The **Cloud Auditor** provides a valuable inherent function for the government by conducting the independent performance and security monitoring of cloud services. The **Cloud Carrier** is the organization which has the responsibility of transferring the data, somewhat akin to the power distributor for the electric grid.

Figure 1 – Cloud Actors briefly lists the five major actors defined in the NIST cloud computing reference architecture.

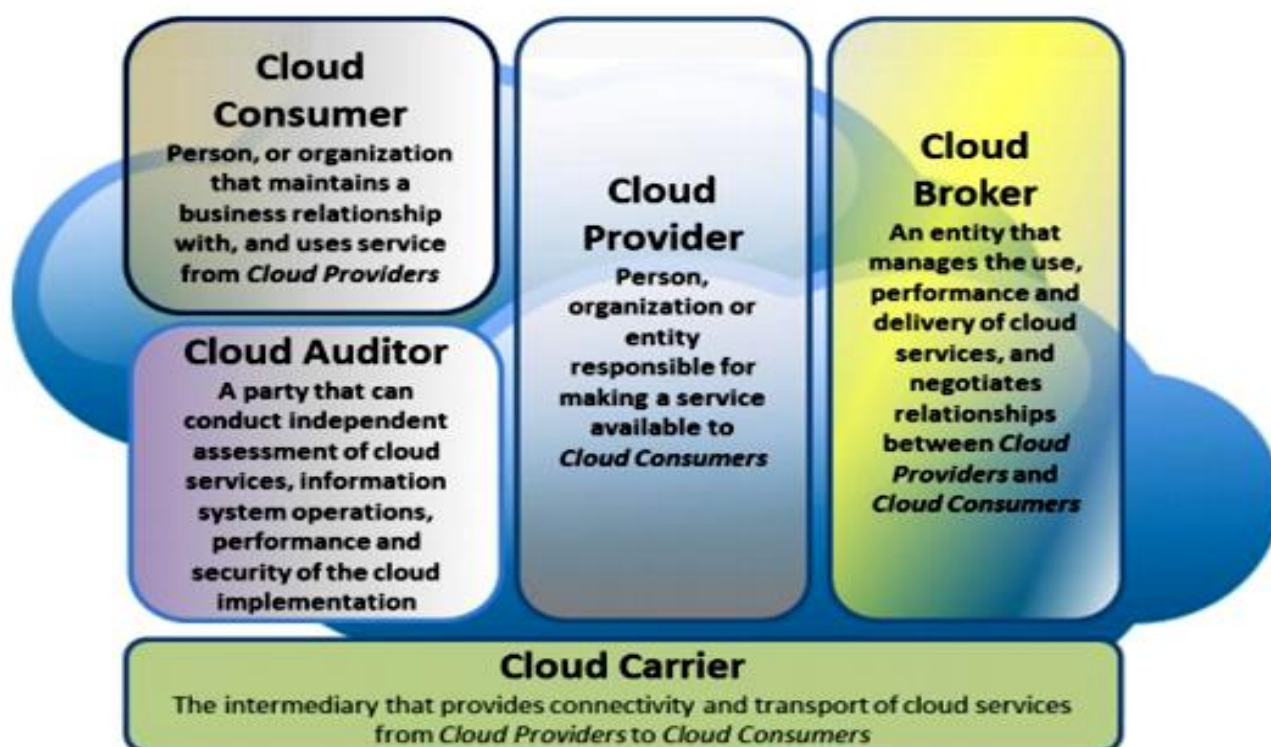
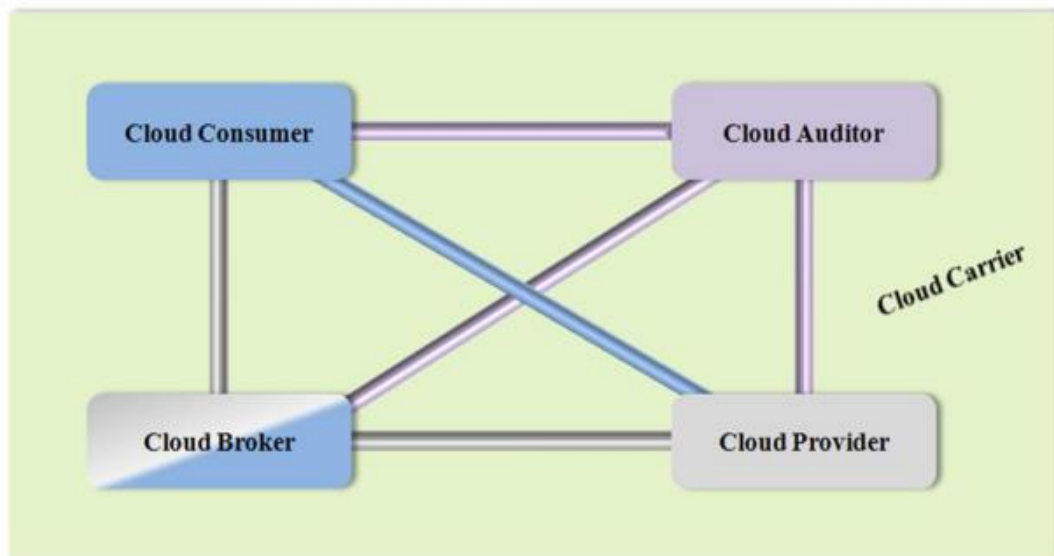


Figure 2 – Interactions between the Actors in Cloud Computing shows the interactions among the actors in the NIST cloud computing reference architecture. A cloud consumer may request cloud services from a cloud provider directly or via a cloud broker. A cloud auditor conducts independent

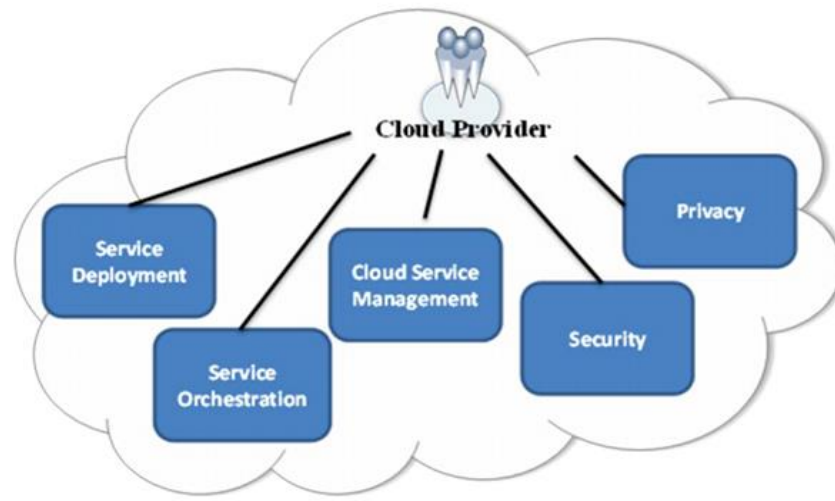
audits and may contact the others to collect necessary information. The details will be discussed in the following sections and be presented as successive diagrams in increasing levels of detail.



1) CLOUD CONSUMER - The cloud consumer is the ultimate stakeholder that the cloud computing service is created to support. A cloud consumer represents a person or organization that maintains a business relationship with, and uses the service from, a cloud provider.

Service Models	Consumer Activities	Provider Activities
SaaS	Uses application/service for business process operations.	Installs, manages, maintains, and supports the software application on a cloud infrastructure.
PaaS	Develops, tests, deploys, and manages applications hosted in a cloud system.	Provisions and manages cloud infrastructure and middleware for the platform consumers; provides development, deployment, and administration tools to platform consumers.
IaaS	Creates/installs, manages, and monitors services for IT infrastructure operations.	Provisions and manages the physical processing, storage, networking, and the hosting environment and cloud infrastructure for IaaS consumers.

2) CLOUD PROVIDER - A cloud provider can be a person, an organization, or an entity responsible for making a service available to cloud consumers. A cloud provider builds the requested software/platform/ infrastructure services, manages the technical infrastructure required for providing the services, provisions the services at agreed-upon service levels, and protects the security and privacy of the services.



For SaaS, the cloud provider deploys, configures, maintains, and updates the operation of the software applications on a cloud infrastructure so that the services are provisioned at the expected service levels to cloud consumers.

For PaaS, the cloud provider manages the cloud infrastructure for the platform, and provisions tools and execution resources for the platform consumers to develop, test, deploy, and administer applications.

For IaaS, the cloud provider provisions the physical processing, storage, networking, and other fundamental computing resources, as well as manages the hosting environment and cloud infrastructure for IaaS consumers.

The activities of cloud providers can be discussed in greater detail from the perspectives of Service Deployment, Service Orchestration, Cloud Service Management, Security and Privacy.

➤ SERVICE DEPLOYMENT

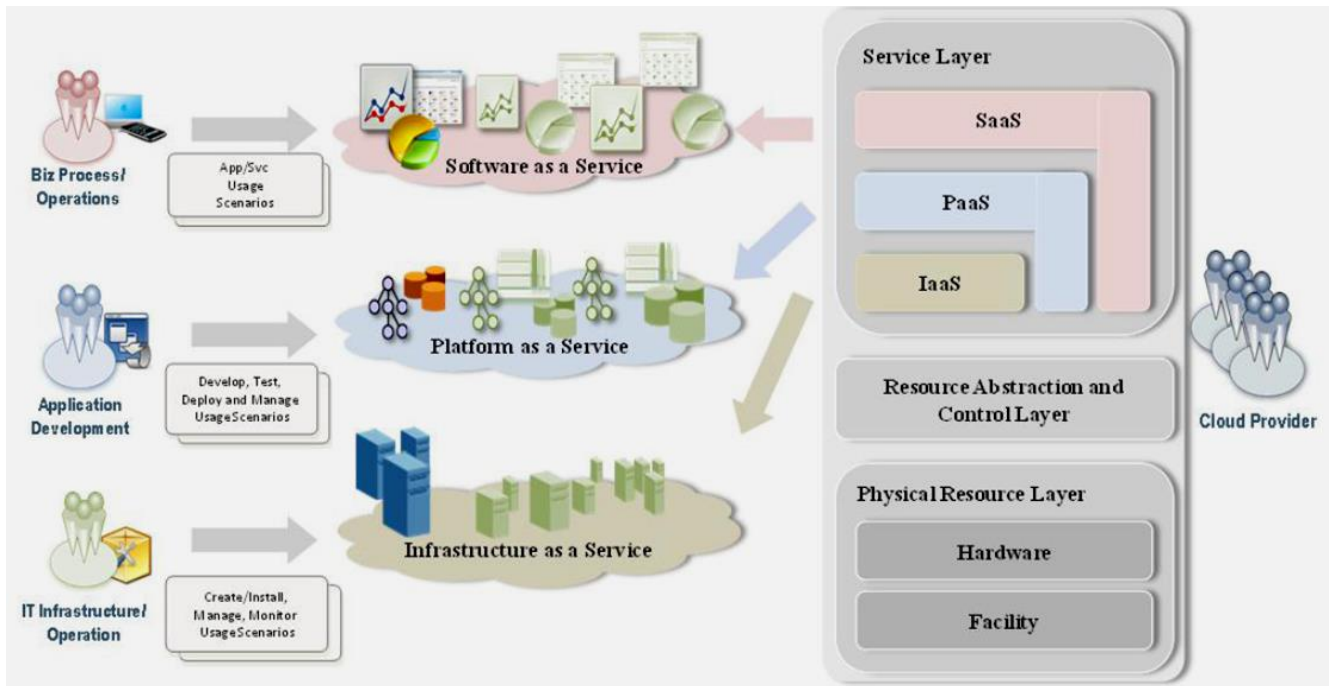
As identified in the NIST cloud computing definition, a cloud infrastructure may be operated in one of the following deployment models: public cloud, private cloud, community cloud, or hybrid cloud.

- A **public cloud** is one in which the cloud infrastructure and computing resources are made available to the general public over a public network. A public cloud is owned by an organization selling cloud services and serves a diverse pool of clients.
- For **private clouds**, the cloud infrastructure is operated exclusively for a single organization. A private cloud gives the organization exclusive access to and usage of the infrastructure and computational resources. It may be managed either by the organization or by a third party, and may be implemented at the organization's premise (i.e., on-site private clouds) or outsourced to a hosting company (i.e., outsourced private clouds).
- Similar to private clouds, a **community cloud** may be managed by the organizations or by a third party, and may be implemented at the customer's location (i.e., on-site community cloud) or outsourced to a hosting company (i.e., outsourced community cloud). However, a community cloud serves a set of organizations that have common security, privacy, and compliance considerations, rather than serving a single organization as does a private cloud.

- A hybrid cloud is a composition of two or more cloud deployment models (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability. As discussed in this section, both private clouds and community clouds can be either implemented on-site or outsourced to a third party. Therefore, each constituent cloud of a hybrid cloud can be one of the five variants.

➤ SERVICE ORCHESTRATION

Service orchestration refers to the arrangement, coordination, and management of cloud infrastructure to provide the optimizing capabilities of cloud services, as a cost-effective way of managing IT resources, as dictated by strategic business requirements. Figure 5 shows the general requirements and processes for cloud providers to build each of the three service models.

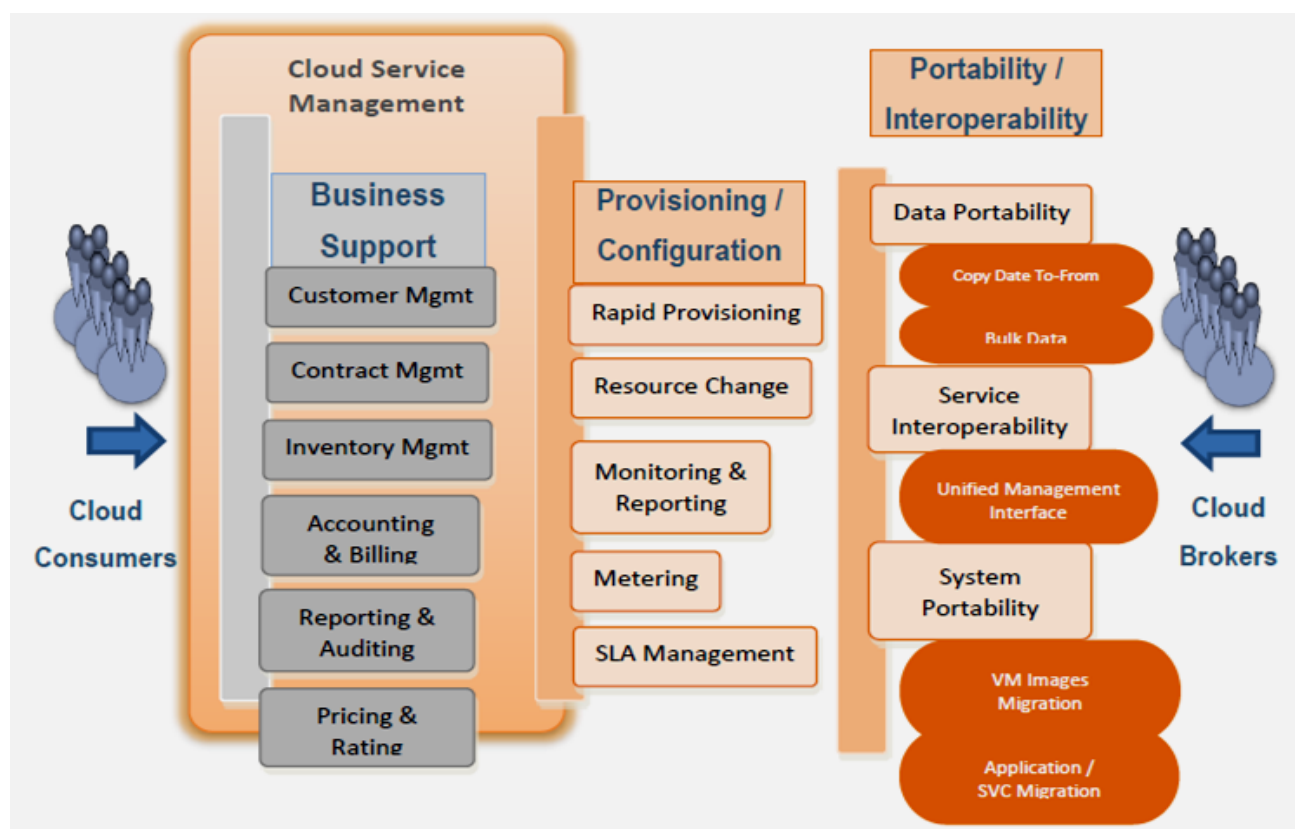


- A three-layered framework is identified for a generalized cloud system in Figure 5. The top layer is the service layer, where a cloud provider defines and provisions each of the three service models.
- The middle layer is the resource abstraction and control layer. This layer contains the system components that a cloud provider uses to provide and manage access to the physical computing resources through software abstraction.
- The lowest layer in the framework is the physical resource layer, which includes all the physical computing resources. This layer includes hardware resources, such as computers (CPU and memory), networks (routers, firewalls, switches, network links, and interfaces), storage components (hard disks), and other physical computing infrastructure elements.

➤ 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. As illustrated in Figure 6, cloud service management can be described from the perspective of *business support*, *provisioning and configuration*, and from the perspective of *portability and interoperability* requirements.

Figure 6 – Cloud Provider: Cloud Service Management



➤ SECURITY

“As the Federal Government moves to the cloud, it must be vigilant to ensure the security and proper management of government information to protect the privacy of citizens and national security”.

Security is a cross-cutting function that spans all layers of the reference architecture involving end-to-end security that ranges from physical security to application security, and in general, the responsibility is shared between cloud provider and federal cloud consumer.

Cloud Providers should ensure that the facility hosting cloud services is secure and that the staff has proper background checks. When data or applications are moved to a cloud, Cloud Consumers ensure that the cloud offering satisfies the security requirements and enforces the compliance rules.

➤ PRIVACY

Cloud providers should protect the assured, proper, and consistent collection, processing, communication, use, and disposition of personal information (PI) and personally identifiable information (PII) in the cloud system.

3) CLOUD AUDITOR - A cloud auditor is a party that can conduct independent assessment of cloud services, information system operations, performance, and the security of a cloud computing implementation. A cloud auditor can evaluate the services provided by a cloud provider in terms of security controls, privacy impact, performance, and adherence to service level agreement parameters.

4) CLOUD BROKER - The NIST Reference Architecture defines a Cloud Broker as an entity that manages the use, performance, and delivery of cloud services, and negotiates relationships between

Cloud Providers and Cloud Consumers. As cloud computing evolves, the integration of cloud services may become too complex for cloud Consumers to manage. In such cases, a Cloud Consumer may request cloud services from a Cloud Broker instead of directly contacting a Cloud Provider. Cloud Brokers provide a single point of entry for managing multiple cloud services. In general, Cloud Brokers provide services in three categories:

Intermediation: A Cloud Broker enhances a given service by improving some specific capability and providing value-added services to cloud Consumers. The improvement can be managing access to cloud services, identity management, performance reporting, enhanced security, etc.

Aggregation: A Cloud Broker combines and integrates multiple services into one or more new services. The Broker provides data and service integration and ensures the secure data movement between the cloud Consumer and multiple cloud Providers.

Arbitrage: Service arbitrage is similar to service aggregation except that the services being combined/consolidated are not fixed. Service arbitrage means a Broker has the flexibility to choose services from multiple service Providers.

5) CLOUD CARRIER - A cloud carrier acts as an intermediary that provides connectivity and transport of cloud services between cloud consumers and cloud providers. Cloud carriers provide access to consumers through network, telecommunication, and other access devices. For example, cloud consumers can obtain cloud services through network access devices, such as computers, laptops, mobile phones, mobile Internet devices (MIDs), etc.

The distribution of cloud services is normally provided by network and telecommunication carriers or a transport agent, where a transport agent refers to a business organization that provides physical transport of storage media such as high-capacity hard drives. Note that a cloud provider will set up service level agreements (SLAs) with a cloud carrier to provide services consistent with the level of SLAs offered to cloud consumers, and may require the cloud carrier to provide dedicated and encrypted connections between cloud consumers and cloud providers.

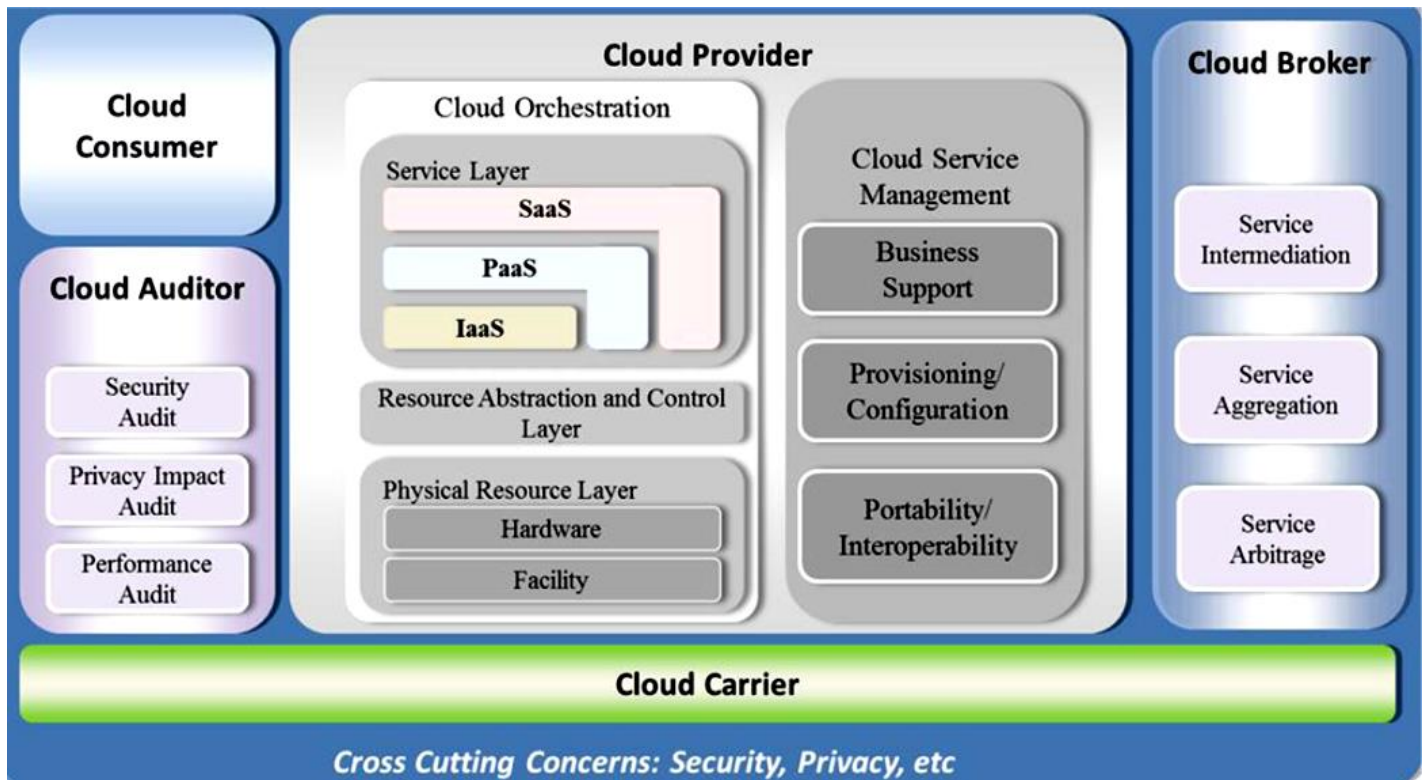


Figure – The Combined Conceptual Reference Diagram

⇒ CLOUD MODELS

Cloud computing is a model for enabling ubiquitous, 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. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

➤ The 5 Essential Characteristics of Cloud Computing:

1) *On-demand self-service.*

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

2) *Broad network access.*

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

3) *Resource pooling.*

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no

control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

4) *Rapid elasticity.*

Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

5) *Measured service.*

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Service Models:

1) Software as a service (SaaS)

A service provider delivers software and applications through the internet. Users subscribe to the software and access it via the web or vendor APIs.

Software as a service (SaaS) is a cloud computing offering that provides users with access to a vendor's cloud-based software. Users do not install applications on their local devices. Instead, the applications reside on a remote cloud network accessed through the web or an API. Through the application, users can store and analyze data and collaborate on projects.

- SaaS vendors provide users with software and applications via a subscription model.
- Users do not have to manage, install or upgrade software; SaaS providers manage this.
- Data is secure in the cloud; equipment failure does not result in loss of data.
- Use of resources can be scaled depending on service needs.
- Applications are accessible from almost any internet-connected device, from virtually anywhere in the world.

2) Platform as a service (PaaS)

A service provider offers access to a cloud-based environment in which users can build and deliver applications. The provider supplies underlying infrastructure.

Platform as a service (PaaS) is a cloud computing offering that provides users with a cloud environment in which they can develop, manage and deliver applications. In addition to storage and other computing resources, users are able to use a suite of prebuilt tools to develop, customize and test their own applications.

Key features

- PaaS provides a platform with tools to test, develop and host applications in the same environment.
- Enables organizations to focus on development without having to worry about underlying

infrastructure.

- Providers manage security, operating systems, server software and backups.
- Facilitates collaborative work even if teams work remotely.

3) Infrastructure as a service (IaaS)

A vendor provides clients pay-as-you-go access to storage, networking, servers and other computing resources in the cloud.

Infrastructure as a service (IaaS) is a cloud computing offering in which a vendor provides users access to computing resources such as servers, storage and networking. Organizations use their own platforms and applications within a service provider's infrastructure.

Key features:

- Instead of purchasing hardware outright, users pay for IaaS on demand.
- Infrastructure is scalable depending on processing and storage needs.
- Saves enterprises the costs of buying and maintaining their own hardware.
- Because data is on the cloud, there can be no single point of failure.
- Enables the virtualization of administrative tasks, freeing up time for other work.

Service Models		Cloud Stack	Stack Components		Who is Responsible
SaaS PaaS IaaS		User	Login		Customer
			Registration		
			Administration		
		Application	Authentication	Authorization	
			User Interface	Transactions	
			Reports	Dashboard	
		Application Stack	OS	Prog Lang	
			App Server	Middleware	
			Database	Monitoring	
		Infrastructure	Data Center	Disk Storage	
			Servers	Firewall	
			Network	Load Balancer	
				Vendor	

⇒ CLOUD MODELS: Public Vs Private Cloud

❖ Public Cloud

The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

- Available to general public
- Owned and provisioned by an organization selling cloud services
- NIST definition

- * Cloud infrastructure provisioned for open use by general public
- * May be owned, managed, and operated by some combination of a business, academic, or government organization
 - Multitenant environment
 - Resources used from a shared grid of commodity resources
 - Users unaware of physical location of resources or data center
 - * Users access resources through an abstraction layer on top of the physical hardware
 - * Virtual compute resources created by APIs in the abstraction
- Advantages of public clouds
 - * Utility pricing
 - Pay for the resources consumed
 - Scale up or scale down as per need
 - No procurement of physical hardware, except for the hardware to connect to the cloud
 - No wasted compute cycles
 - * Elasticity
 - Endless pool of resources
 - Configure software solutions to dynamically increase/decrease resources to handle peak loads
 - React to traffic spikes in real-time
 - * Core competency
 - Outsourced data center and infrastructure management
 - More time on core competence
- Risks of public clouds
 - * Control
 - Reliance on vendor for performance and uptime
 - Outage at cloud vendor could adversely affect services
 - * Regulatory issues
 - PCI DSS – Payment Card Industry Data Security Standard
 - HIPAA – Health Information Portability and Accountability Act
 - Data privacy issues
 - May be solved by leveraging certified SaaS solutions for components that are hard to audit in public cloud
 - * Limited configurations

- May not be able to access specific hardware to solve intensive computational problems

❖ **Private Cloud**

The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

- Hosted within an organization's firewall, managed by the organization
- Created and controlled by the enterprise
- NIST definition
 - * Cloud infrastructure provisioned for exclusive use by a single organization with multiple consumers or business units
 - * May be owned, managed, and operated by the organization, a third party, or some combination
 - * May exist on or off premises
- Deploy in a single-tenant environment and not comingled with other customers
- Costs more than sharing in public cloud environment, but more control and security
- Reduce regulatory risks

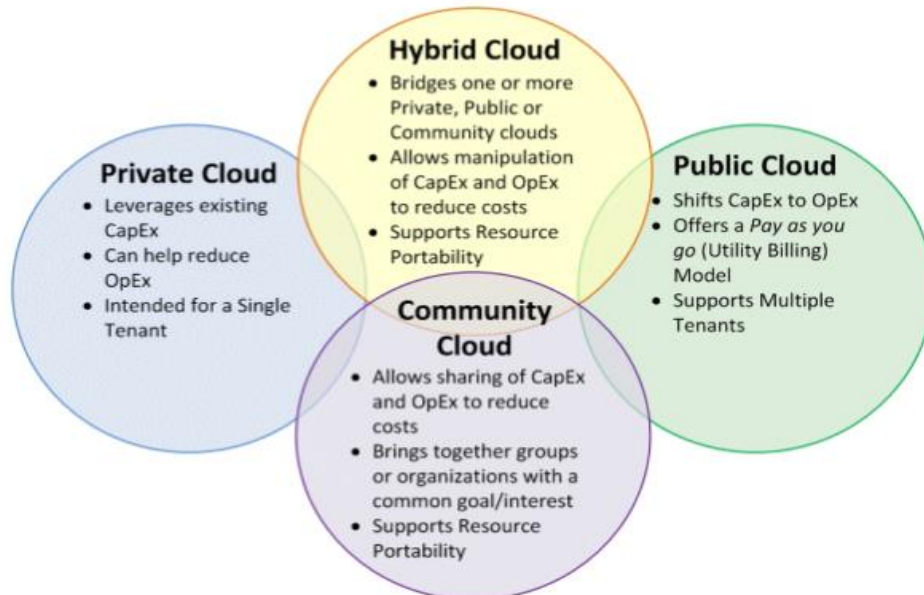
❖ **Hybrid Cloud**

The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

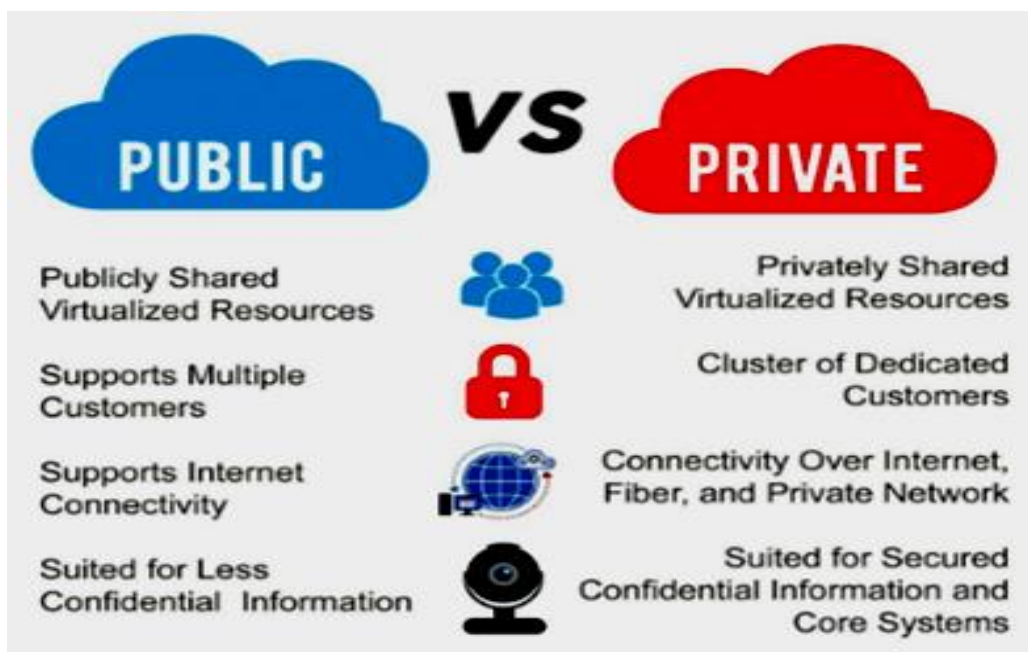
- Combination of public and private clouds
- Management responsibilities divided between public cloud provider and the business renting it
- NIST definition
 - * Composition of two or more distinct cloud infrastructures (private, community, or public)
 - * The clouds remain unique entities but bound together by standardized or proprietary technologies to enable data/application portability
 - * Cloud bursting for load balancing between clouds

❖ Community Cloud.

The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the

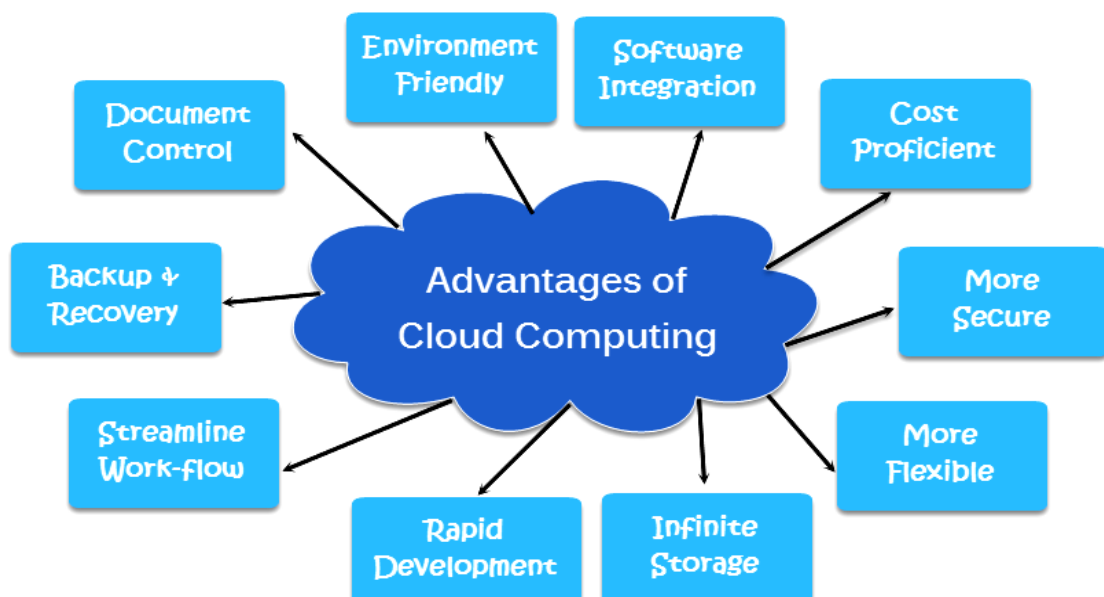


community, a third party, or some combination of them, and it may exist on or off premises.



⇒ CLOUD SOLUTIONS

- A **cloud based solution** refers to on-demand services, computer networks, storage, applications or resources accessed via the internet and through another provider's shared **cloud** computing infrastructure.
- The benefits of cloud based solutions to end users and businesses include increased capacity, scalability, functionality, and reduced maintenance and cost for computer infrastructure or in-house staff. Additionally, cloud-based solutions can enable companies to focus on revenue driving initiatives rather than time-consuming, non-core business tasks.
- The ability to access cloud-based solutions from anywhere with an internet connection paired with the widespread adoption of smart phones and faster mobile networks have given users the ability to access cloud-based solutions from anywhere and anytime.



- Sparkle Cloud Solutions allow customers to enjoy computing resources on a pay as you grow model. They can access data, applications and services from any location.

⇒ **CLOUD ECOSYSTEM**

- ✓ A **cloud ecosystem** is a complex system of interdependent components that all work together to enable **cloud** services. ... In **cloud** computing, the **ecosystem** consists of hardware and software as well as **cloud** customers, **cloud** engineers, consultants, integrators and partners.

➤ **The concept of a cloud ecosystem**

Cloud is motivating business and IT to emerge from their respective silos and forge partnerships that apply new technologies in innovative ways. But converting cloud's considerable benefits into business opportunities requires an astute understanding of your cloud ecosystem, including:

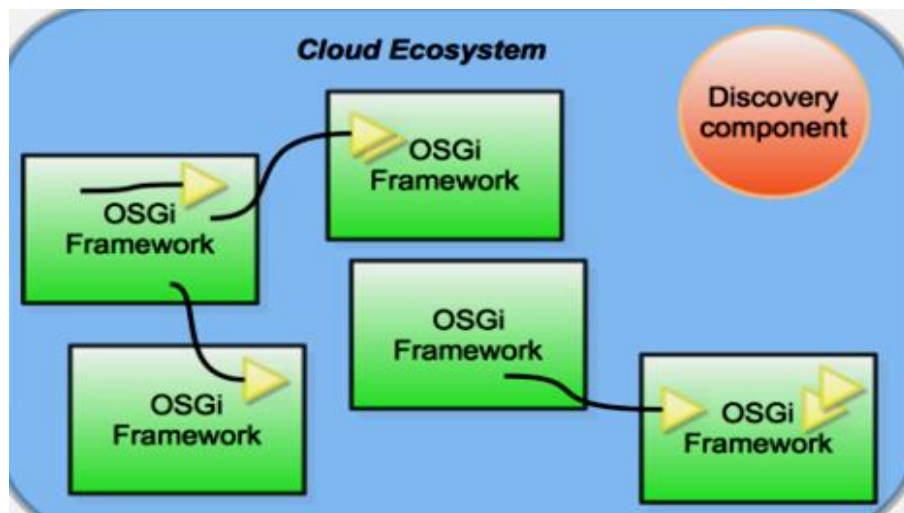
- Determining the type of cloud environment (public, private or hybrid) best suited for your organization.
- Developing your cloud adoption vision, including governance strategy, business outcomes, and project benefits.
- Establishing use cases and a detailed plan.
- Understanding the implications of adopting specific Cloud Service Layers, including business process as a service (BPaaS), software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS).

➤ **The benefits of a Cloud Ecosystem**

- ⌚ Companies can use a cloud ecosystem to build new business models. It becomes relatively easy for a medical device manufacturer, for example, to launch a heart-monitoring service on its cloud service provider's cloud infrastructure and then sell the service alongside its main business of manufacturing heart monitors for hospitals.
- ⌚ In a cloud ecosystem, it is also easier to aggregate data and analyze how each part of the system affects the other parts. For example, if an ecosystem consists of patient records, smart device logs and healthcare provider records, it becomes possible to analyze patterns across an entire patient population.
- ⌚ Data Migration provides an automated mechanism to transfer data from one **cloud** environment to another or to other computing systems. Data Synchronization is the mechanism to ensure consistency across the **Cloud Ecosystem** to a single set of source data for all duplicated target data storage and vice versa.



- ✓ The center of a cloud ecosystem is a public cloud provider. It might be an IaaS provider such as Amazon Web Services (AWS) or a SaaS vendor such as Sales force. ... For example, AWS is the center of its own ecosystem, but it's also a part of the Sales force ecosystem.

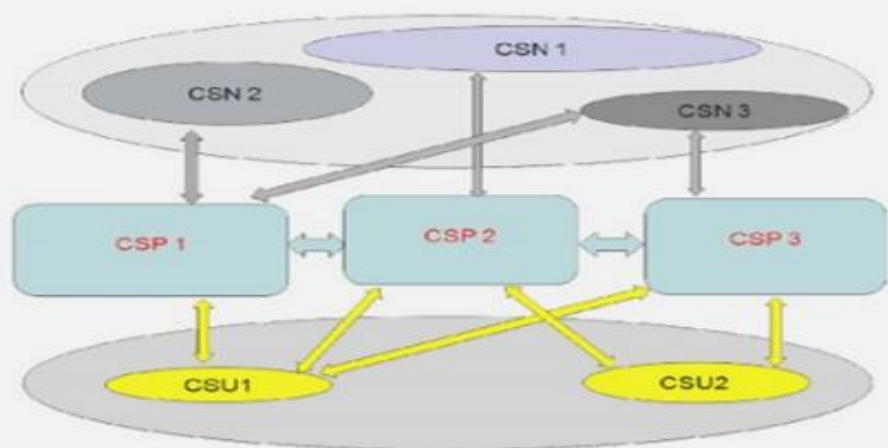


The three actors of cloud ecosystem

Cloud Service Partner (CSN)

Cloud Service Provider (CSP)

Cloud service User (CSU)



⇒ CLOUD SERVICE MANAGEMENT

- Cloud service management and operations refer to all the activities that an organization does to plan, design, deliver, operate, and control the IT and cloud services that it offers to customers.
- Service management includes the operational aspects of your applications and services. After an application is pushed to production, it must be managed. Applications are monitored to ensure availability and performance according to service level agreements (SLAs) or service level objectives (SLOs).
- The delivery of dynamic, cloud-based infrastructure, platform and application services doesn't occur in a vacuum. In addition to best practices for effective administration of all the elements associated with cloud service delivery, cloud service management and cloud monitoring tools enable providers to keep up with the continually shifting capacity demands of a highly elastic environment.
- Cloud monitoring and cloud service management tools allow cloud providers to ensure optimal performance, continuity and efficiency in virtualized, on-demand environments. These tools -- software that manages and monitors networks, systems and applications -- enable cloud providers not just to guarantee performance, but also to better orchestrate and automate provisioning of resources.
- Cloud monitoring tools, specifically, enable cloud providers to track the performance, continuity and security of all of the components that support service delivery: the hardware, software and services in the data center and throughout the network infrastructure.

Aspects of service management

- Cloud service management and operations is redefining traditional service management to better fit the needs for cloud and DevOps patterns. At the same time, it bridges traditional approaches to service management, such as the IT Infrastructure Library (ITIL).
- The service management reference architecture includes incident management, problem management, change management, and operations.

► Incident management

Incident management aims to restore the service as quickly as possible by using a first-responder team that is equipped with automation and well-defined runbooks. To maintain the best possible levels of service quality and availability, the incident management team performs sophisticated monitoring to detect issues early, before the service is affected.

► Problem management

Problem management aims to resolve the root causes of incidents to minimize their adverse impact and prevent recurrence. Capabilities include root-cause analysis, incident analysis, dashboards, and collaboration.

► Change management

The purpose of change management is to achieve the successful introduction of changes to an IT system or environment. Success is measured as a balance of the timeliness and completeness of change implementation, the cost of implementation, and the minimization of disruption that is caused in the target system or environment.

► Operations

Operations describes the activities to deliver the right quantity of the right set of services at competitive costs for customers. IT operations management runs daily routine tasks that are related to the operation of infrastructure components and applications. These tasks include application and systems availability, health checks, compliance checks, performance monitoring, backups, and capacity monitoring and management.

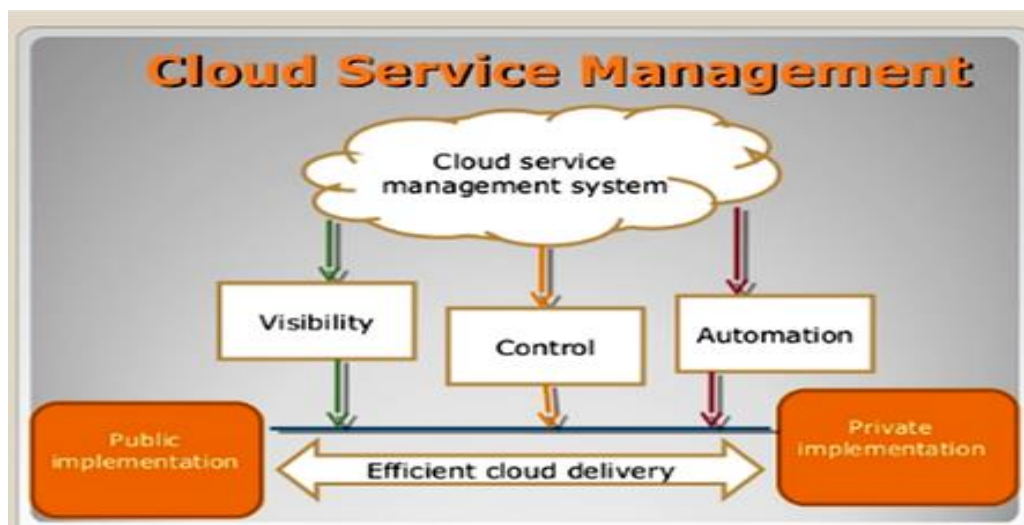
➤ What are the key processes associated with cloud service management?

- ✓ Cloud service management shares some basic principles with traditional IT service management (ITSM). Cloud management tools help providers administrate the systems and applications that facilitate the on-demand service delivery model. The goal of these practices is to improve the efficiency of the cloud environment and achieve a high level of customer satisfaction.
- ✓ Essentially, cloud service management takes the customer perspective as the measure of service assurance and manages all the individual IT resources in a way that will support that. This involves adjusting the operations and policies, as necessary, of all the assets in the virtual environment that support and affect the on-demand service delivery model. Such assets include servers, software and services that provide access and connectivity to these cloud services.
- ✓ The core elements of cloud service management mirror those of traditional ITSM -- including cloud service-level agreement (SLA) management, cloud capacity management, availability management and billing -- and are applied to administrate a cloud delivery environment in a systemic way. These processes are supported with tools that track provisioning and change management, configuration management, release management, incident management, performance management and service continuity.

Cloud service management platforms: Build or buy?

Although vendors have developed many cloud service management and monitoring tools for enterprises that build and manage their own private clouds, there are far fewer tools that meet the scale,

security and performance requirements of cloud providers. Beyond that, there are even fewer solutions that provide the comprehensive capabilities associated with the entire ITSM process for cloud providers, namely orchestration.



⇒ COMPUTING ON DEMAND

- ❖ **On-demand computing** is a business **computing** model in which **computing** resources are made available to the user on an “as needed” basis. Rather than all at once, **on-demand computing** allows **cloud** hosting companies to provide their clients with access to **computing** resources as they become necessary.
- ❖ **On-demand computing** is a delivery 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 cloud service provider.
- ❖ On-demand computing (ODC) is an enterprise-level model of technology and computing in which resources are provided on an as-needed and when-needed basis. ODC make computing resources such as storage capacity, computational speed and software applications available to users as and when needed for specific temporary projects, known or unexpected workloads, routine work, or long-term technological and computing requirements. Web services and other specialized tasks are sometimes referenced as types of ODC.

On-Demand Computing		
Definition of On-Demand • Remote resources • Self-provisioned • User-operated		
Comparison		
In-house	On-Demand	Outsource
local resources	remote resources	remote resources
IT-provisioned	self-provisioned	vendor-provisioned
user-operated	user-operated	vendor-operated

Advantages of On-Demand Computing:

The on-demand computing model was developed to overcome the common challenge that enterprises encountered of not being able to meet unpredictable, fluctuating computing demands in an efficient manner. Businesses today need to be agile and need the ability to scale resources easily and quickly based on rapidly changing market needs.

Industry experts predict on-demand computing to soon be the most widely used computing model for enterprises.

What is Cloud Computing?

“Cloud refers to Internet or Network”. “Cloud is an off-premise form of computing that stores data on the Internet.” Cloud can provide services over network i.e. public or private networks such as LAN, WAN, VPN.

- ❖ Cloud computing is defined to be Internet based computing technology, where the term 'cloud' simply means Internet – and cloud computing refers to services that are accessed directly over the

Internet. A 'cloud' data server or cluster is a collection of computer servers maintained by a cloud provider to provide computing services on a massive scale.

- ❖ Cloud computing is the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer. This is an **internet** definition of **Cloud computing**.
- ❖ Cloud computing is a general term for the delivery of hosted services over the internet.
- ❖ Cloud computing enables companies to consume a compute resource, such as a virtual machine (VM), storage or an application, as a utility -- just like electricity -- rather than having to build and maintain computing infrastructures in house.
- ❖ **Cloud computing** is the use of various services, such as software development platforms, servers, storage and software, over the internet, often referred to as the "**cloud**.. A user only pays for services used (memory, processing time and bandwidth, etc.).

1) **Cloud computing** is a computing term or metaphor that evolved in the late 2000s, based on utility and consumption of computing resources. Cloud computing involves deploying groups of remote servers and software networks that allow centralized data storage and online access to computer services or resources;

2) **Cloud computing** a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (**IaaS**), Platform-as-a-Service (**PaaS**) and Software-as-a-Service (**SaaS**).

3) **Gartner group** 'A forecaster of Information technology' defines **cloud computing** as a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service using Internet technologies.

4) **National Institute of Standards and Technology (NIST)**, which defines the standard, defines **cloud computing** as, a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.

UNIT- II**VIRTUALIZATION**

- ⊗ Basics of Virtualization
- ⊗ Types of Virtualization
- ⊗ Implementation Levels of Virtualization
- ⊗ Virtualization Structures / Tools and Mechanisms
- ⊗ Virtualization of CPU, Memory, I/O Devices
- ⊗ Virtual Clusters and Resource management
- ⊗ Virtualization for Data-center Automation

⇒ **Basics of Virtualization**

- ▶ In computing, **virtualization** refers to the act of creating a virtual (**rather than actual**) version of something, including virtual computer hardware platforms, storage devices, and computer network resources.
- ▶ Virtualization technology is one of the fundamental components of cloud computing, especially in regard to infrastructure based service. **“Virtualization allows the creation of a secure, customizable, and isolated execution environment for running applications, even if they are un-trusted, without affecting other user’s applications”.**
- ▶ The term virtualization is often synonymous with hardware virtualization, which plays a fundamental role in efficiently delivering Infrastructure-as-a-Service (IaaS) solutions for cloud computing.

➤ **Why Virtualization needed?**

- ✓ Increased performance and computing capacity.
- ✓ Underutilized hardware and software resources.
- ✓ Lack of space.
- ✓ Greening initiatives.
- ✓ Rise of administrative costs.

➤ **Characteristics of virtualized environments**

“Virtualization is a broad concept that refers to the creation of a virtual version of something, whether hardware, a software environment, storage, or a network”. In a virtualized environment there are three major components: **guest, host, and virtualization layer**. The guest represents the system component that interacts with the virtualization layer rather than with the host, as would normally happen. The host represents the original environment where the guest is supposed to be managed. The virtualization layer is responsible for recreating the same or a different environment where the guest will operate (see Figure 3.1).

- **Increased security:** The ability to control the execution of a guest in a completely transparent manner opens new possibilities for delivering a secure, controlled execution environment.

Managed execution: Virtualization of the execution environment not only allows increased security, but a wide range of features also can be implemented. In particular, sharing, aggregation, emulation, and isolation are the most relevant features (see Figure 3.2).

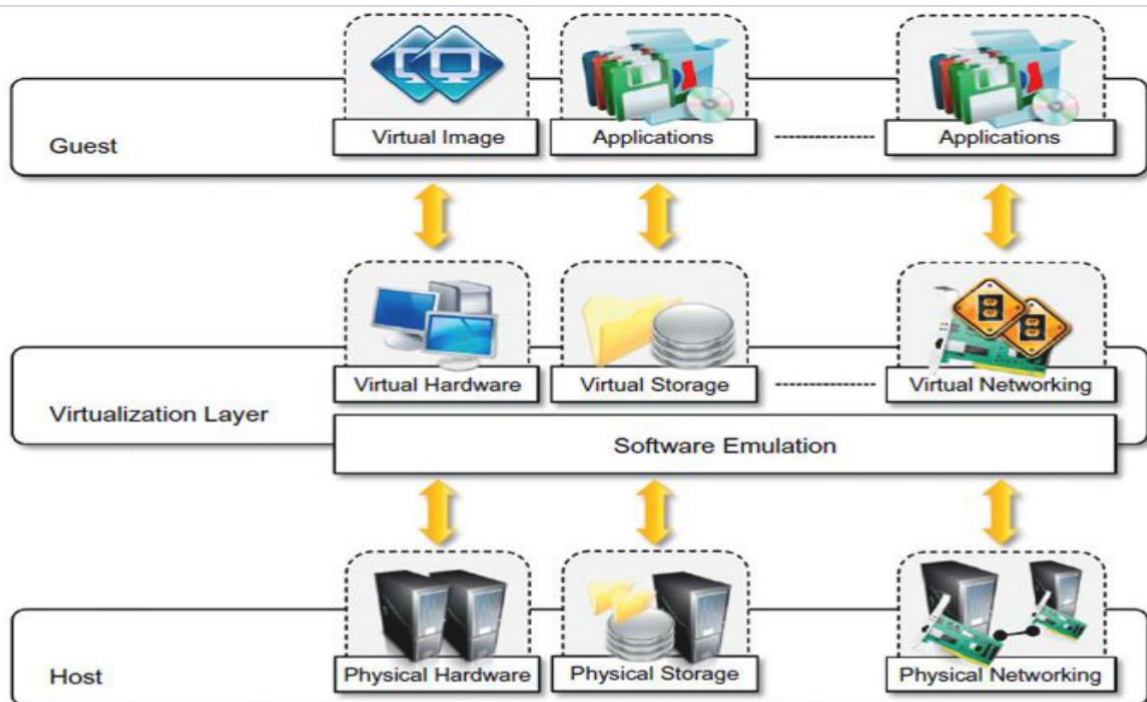


FIGURE 3.1

The virtualization reference model.

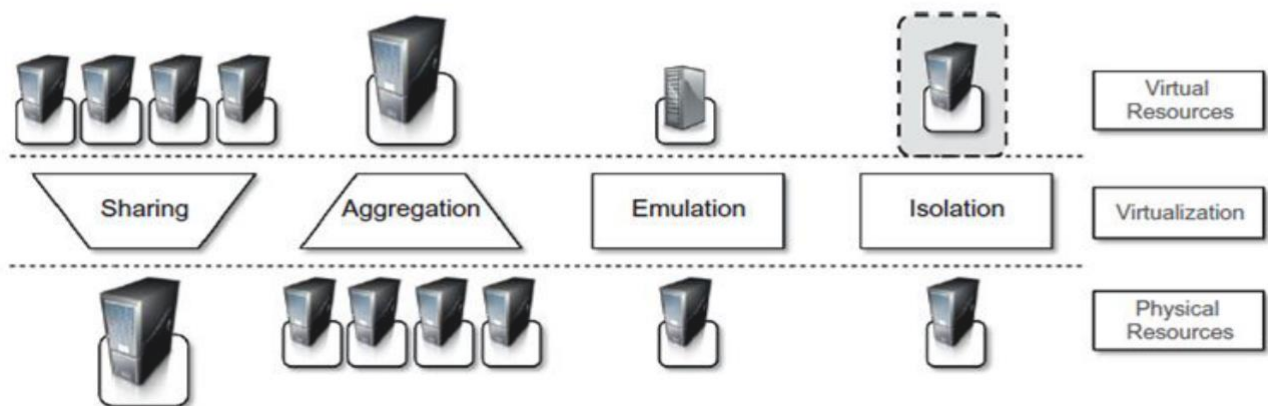


FIGURE 3.2

Functions enabled by managed execution.

Portability: The concept of portability applies in different ways according to the specific type of virtualization considered. In the case of a hardware virtualization solution, the guest is packaged in to a virtual image that, in most cases, can be safely moved and executed on top of different virtual machines.

➤ Benefits of Virtualization

1. More flexible and efficient allocation of resources.
2. Enhance development productivity.

3. It lowers the cost of IT infrastructure.
4. Remote access and rapid scalability.
5. High availability and disaster recovery.
6. Pay per use of the IT infrastructure on demand.
7. Enables running multiple operating system.

⇒ **Types of Virtualization**

1) Application Virtualization:

Application virtualization helps user to have a remote access of an application from a server. The server stores all personal information and other characteristics of the application, but can still run on a local workstation through internet. Example of this would be a user who needs to run two different versions of the same software. Application virtualization is abstracting the application layer away from the operating system. This way the application can run in an encapsulated form without being depended upon on the operating system underneath. This can allow a Windows application to run on Linux and vice versa, in addition to adding a level of isolation.

2) Network Virtualization:

The ability to run multiple virtual networks that each has a separate control and data plan. It co-exists together on top of one physical network. It can be managed by individual parties that potentially confidential to each other. Network virtualization, provides a facility to create and provision virtual networks logical switches, routers, firewalls, load balancer, Virtual Private Network (VPN), and workload security within days or even in weeks.

Network virtualization is a method of combining the available resources in a network by splitting up the available bandwidth into channels, each of which is independent from the others and can be assigned -- or reassigned -- to a particular server or device in real time.

3) Desktop Virtualization:

Desktop virtualization allows the users' OS to be remotely stored on a server in the data center. It allows the user to access their desktop virtually, from any location by different machine. Users who want specific operating systems other than Windows Server will need to have a virtual desktop. Main benefits of desktop virtualization are user mobility, portability, easy management of software installation, updates and patches.

4) Storage Virtualization:

Storage virtualization is an array of servers that are managed by a virtual storage system. The servers aren't aware of exactly where their data is stored, and instead function more like worker bees in a hive. It makes managing storage from multiple sources to be managed and utilized as a single repository. Storage virtualization software maintains smooth operations, consistent performance and a continuous suite of advanced functions despite changes, breaks down and differences in the underlying equipment. Storage virtualization is commonly used in storage area networks.

5) Server virtualization:

It is the **masking of server resources** -- including the number and identity of individual physical servers, processors and operating systems from server users. The intention is to spare the user from having to

understand and manage complicated details of server resources while increasing resource sharing and utilization and maintaining the capacity to expand later.

6) Data virtualization:

It is abstracting the traditional technical details of data and data management, such as location, performance or format, in favor of broader access and more resiliencies tied to business needs.

⇒ Implementation Levels of Virtualization

- ✓ Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.
- ✓ 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.

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.

Common virtualization layers include the instruction set architecture (ISA) level, hardware level, operating system level, library support level, and application level (see Figure 3.2).

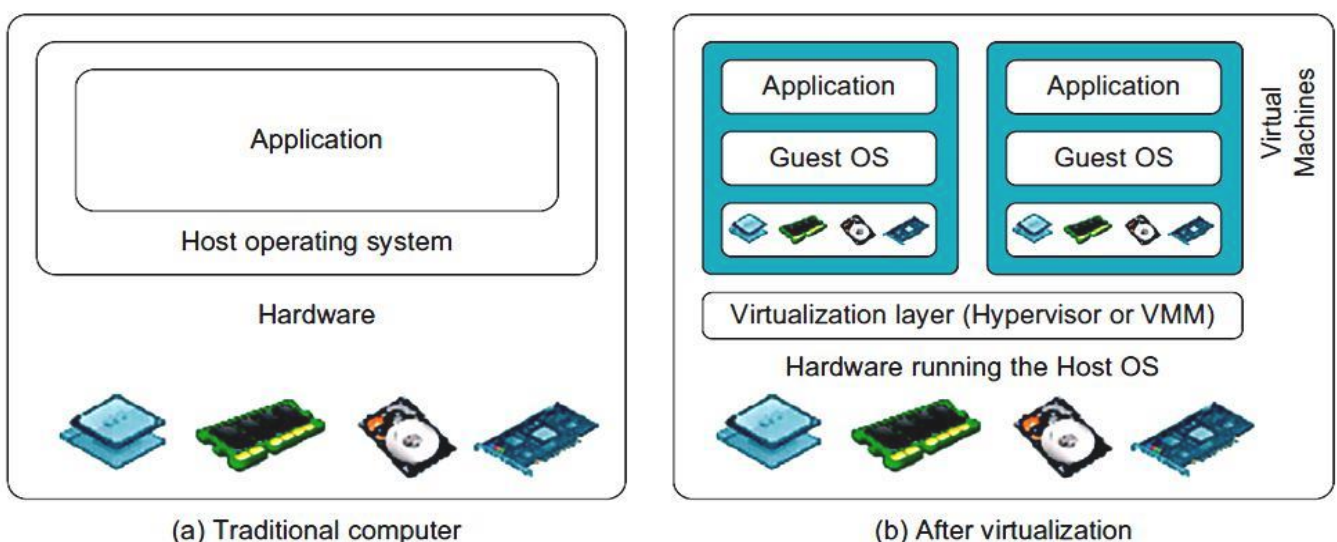
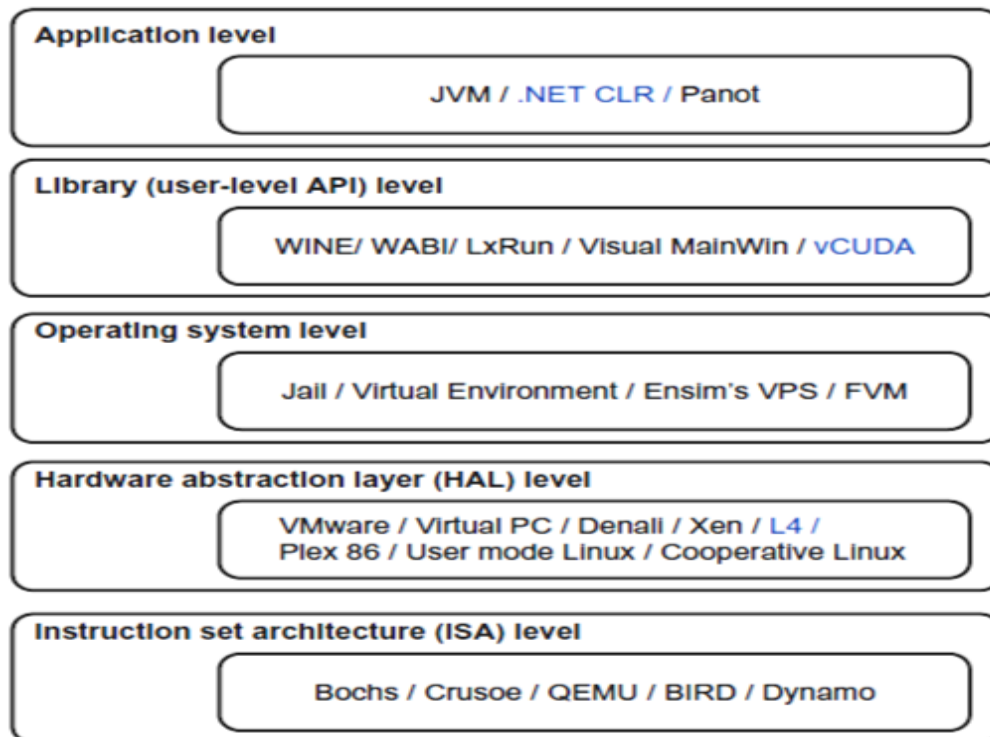


FIGURE 3.1

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



1. Instruction Set Architecture Level

At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. 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.

2. Hardware Abstraction Level

Hardware-level virtualization is performed right on top of the bare hardware. The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices. Hardware-level virtualization inserts a layer between real hardware and traditional operating systems. This layer is commonly called the Virtual Machine Monitor (VMM) and it manages the hardware resources of a computing system.

3. Operating System Level

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.

4. 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.

5. 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. 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.

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

⇒ Virtualization Structures / Tools and Mechanisms

- 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.
- 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, Para virtualization, and Host-based virtualization**. The hypervisor is also known as the VMM (Virtual Machine Monitor).

➤ Hypervisor and Xen Architecture

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. 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.

➤ 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 provides a virtual environment located between the hardware and the OS. The core components of a Xen system are the hypervisor, kernel, and applications.

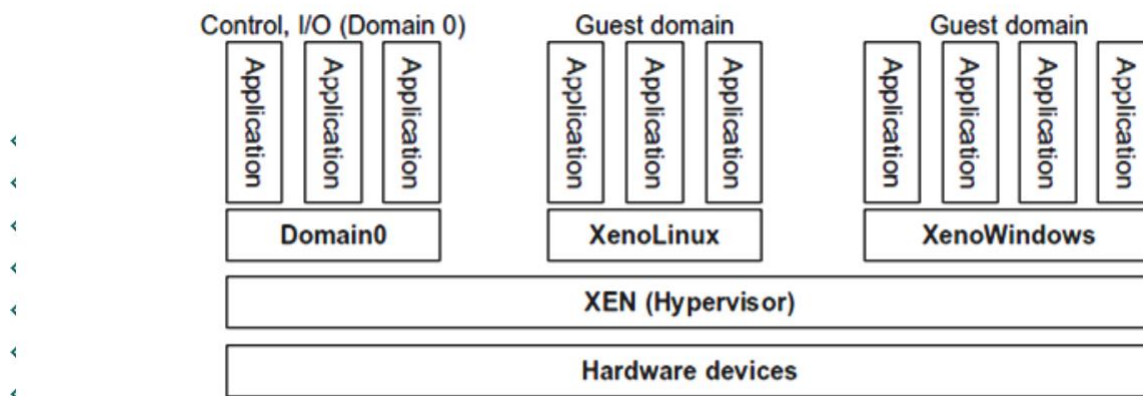


FIGURE 3.5

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

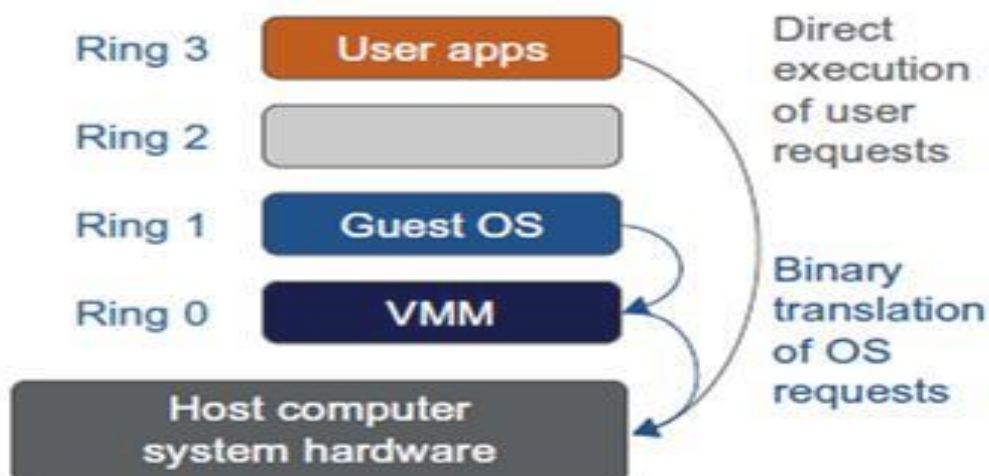
◆ Binary Translation with Full Virtualization

Depending on implementation technologies, 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, non virtualizable instructions. 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.

➤ Host-Based Virtualization

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. This host OS is still responsible for managing the hardware. The guest OSs are installed and run on top of the virtualization layer. Dedicated applications may run on the VMs.



➤ Para-Virtualization with Compiler Support

Para-virtualization needs to modify the guest operating systems. A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications.

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 non virtualizable OS instructions by hyper calls 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.

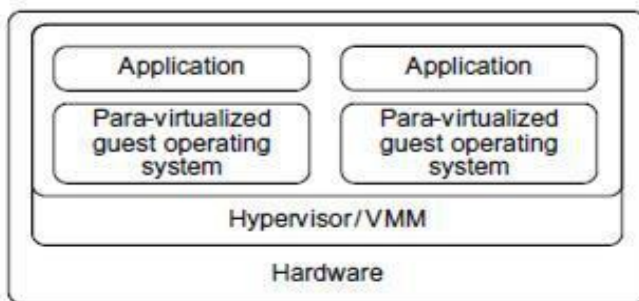


FIGURE 3.7

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.)

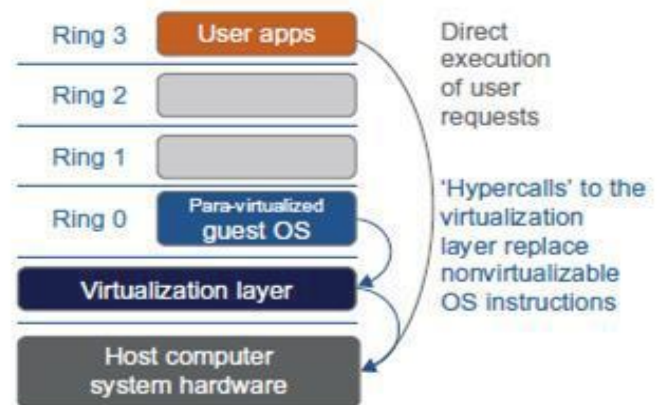


FIGURE 3.8

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

(Courtesy of VMWare [71])

◆ 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 OSs such as Windows, Linux, Solaris, and other UNIX variants.

Unlike the full virtualization architecture which intercepts and emulates privileged and sensitive instructions at runtime, para-virtualization handles these instructions at compile time.

⇒ 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.

➤ 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.

Example: Hardware Support for Virtualization in the Intel x86 Processor.

➤ 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. Other critical instructions should be handled carefully for correctness and stability. The critical instructions are divided into three categories: privileged instructions, control sensitive instructions, and behavior-sensitive instructions.

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.

Example: Intel Hardware-Assisted CPU Virtualization

➤ Memory Virtualization

Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. 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.

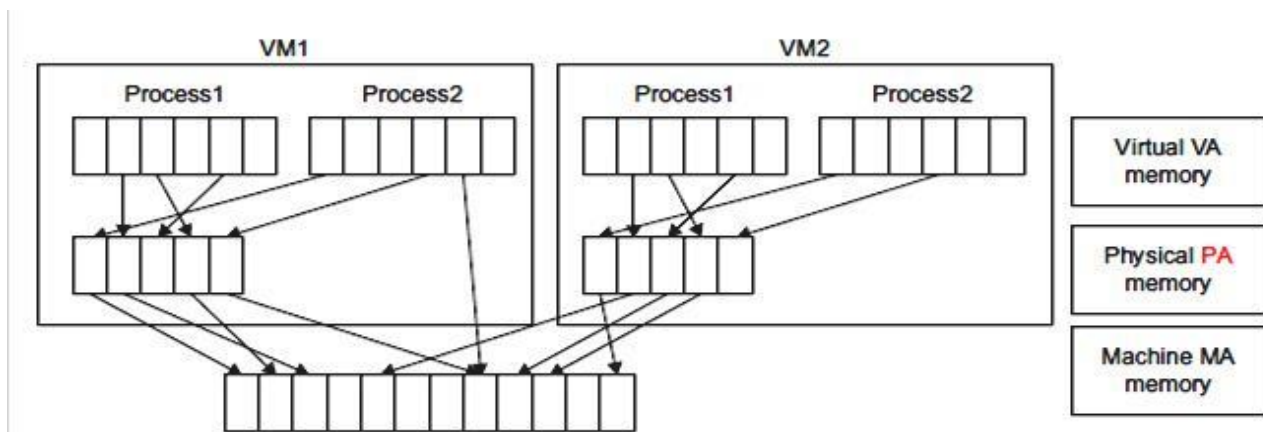


FIGURE 3.12

Two-level memory mapping procedure.

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.

➤ I/O Virtualization

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:

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

Full device emulation is the first approach for I/O 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. **Direct I/O virtualization** lets the VM access devices directly. It can achieve close-to-native performance without high CPU costs.

➤ **Virtualization in Multi-Core Processors**

Virtualizing a multi-core processor is relatively more complicated than virtualizing a uni-core processor. Though multi core processors are claimed to have higher performance by integrating multiple processor cores in a single chip, multi-core virtualization has raised some new challenges to computer architects, compiler constructors, system designers, and application programmers.

⇒ **Virtual Clusters and Resource Management**

- A physical cluster is a collection of servers (physical machines) interconnected by a physical network such as a LAN.
- Here, we introduce virtual clusters and study its properties as well as explore their potential applications. In this section, we will study three critical design issues of virtual clusters: live migration of VMs, memory and file migrations, and dynamic deployment of virtual clusters.

➤ **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 OSs 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.

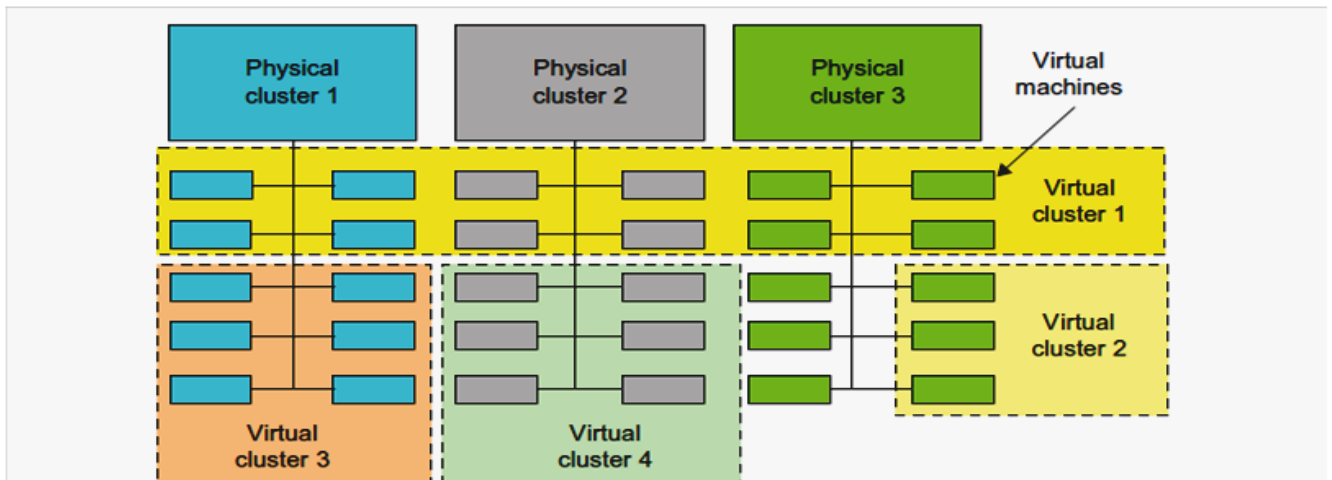


FIGURE 3.18

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.

➤ Fast Deployment and Effective Scheduling

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

- To construct and distribute software stacks (OS, libraries, applications) to a physical node inside clusters as fast as possible, and
- To quickly switch runtime environments from one user's virtual cluster to another user's virtual cluster.

➤ High-Performance Virtual Storage

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.

➤ Live VM Migration Steps and Performance Effects

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. It provides desirable features to satisfy requirements for computing resources in modern computing systems, including server consolidation, performance isolation, and ease of management.

Figure shows the process of live 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.

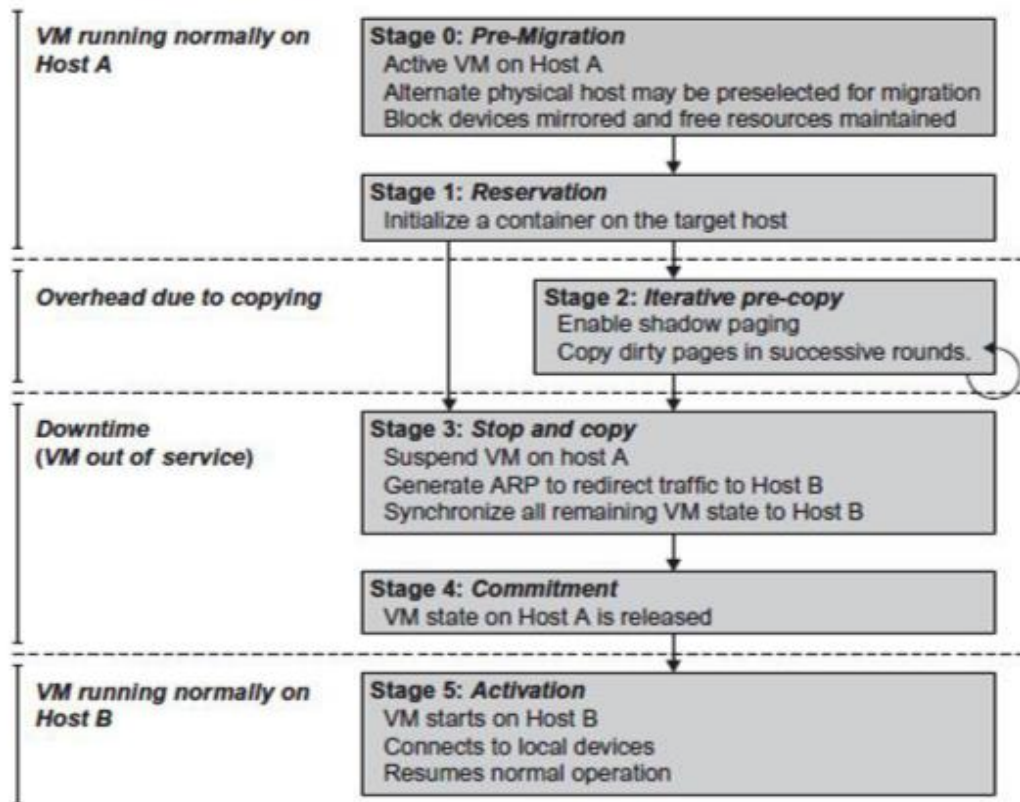


FIGURE 3.20

Live migration process of a VM from one host to another.

➤ 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. 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.

➤ 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.

➤ 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.

➤ Dynamic Deployment of Virtual Clusters

Table 3.5 summarizes four virtual cluster research projects. We briefly introduce them here just to identify their design objectives and reported results. The Cellular Disco at Stanford is a virtual cluster built in a shared-memory multiprocessor system. The INRIA virtual cluster was built to test parallel algorithm performance. The COD and VIOLIN clusters are studied in forthcoming examples.

Table 3.5 Experimental Results on Four Research Virtual Clusters

Project Name	Design Objectives	Reported Results and References
Cluster-on-Demand at Duke Univ. Cellular Disco at Stanford Univ.	Dynamic resource allocation with a virtual cluster management system To deploy a virtual cluster on a shared-memory multiprocessor	Sharing of VMs by multiple virtual clusters using Sun GridEngine [12] VMs deployed on multiple processors under a VMM called Cellular Disco [8]
VIOLIN at Purdue Univ.	Multiple VM clustering to prove the advantage of dynamic adaptation	Reduce execution time of applications running VIOLIN with adaptation [25,55]
GRAAL Project at INRIA in France	Performance of parallel algorithms in Xen-enabled virtual clusters	75% of max. performance achieved with 30% resource slacks over VM clusters

⇒ Virtualization for Data-Center Automation

- ◆ Data centers have grown rapidly in recent years, and all major IT companies are pouring their resources into building new data centers. In addition, Google, Yahoo!, Amazon, Microsoft, HP, Apple, and IBM are all in the game. All these companies have invested billions of dollars in datacenter construction and automation.
- ◆ Data-center automation means that huge volumes of hardware, software, and database resources in these data centers can be allocated dynamically to millions of Internet users simultaneously, with guaranteed QOS and cost-effectiveness.

➤ **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 and non interactive 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. Non interactive workloads do not require people's efforts to make progress after they are submitted.

Server consolidation is an approach to improve the low utility ratio of hardware resources by reducing the number of physical servers. Among several server consolidation techniques such as centralized and physical consolidation, virtualization-based server consolidation is the most powerful. Data centers need to optimize their resource management.

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.

➤ **Virtual Storage Management**

In system virtualization, virtual storage includes the storage managed by VMMs and guest OSes. Generally, the data stored in this environment can be classified into two categories: VM images and application data. *Parallax* is a distributed storage system customized for virtualization environments, 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.

➤ Cloud OS for Virtualized Data Centers

Data centers must be virtualized to serve as cloud providers. Nimbus, Eucalyptus, and OpenNebula are all open source software available to the general public. Only vSphere4 is a proprietary OS for cloud resource virtualization and management over data centers.

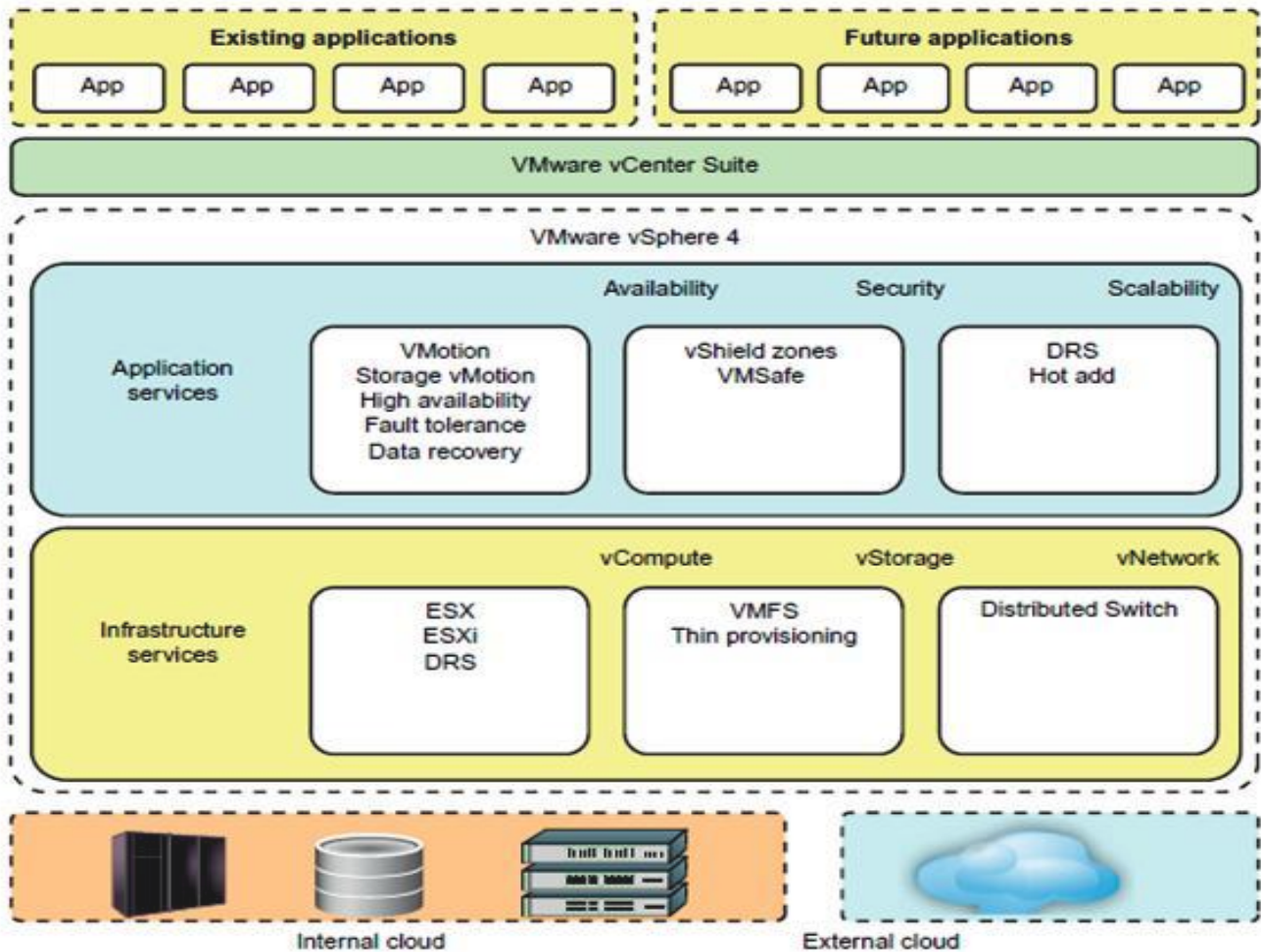


FIGURE 3.28

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

➤ Trust Management in Virtualized Data Centers

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. 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.

➤ VM-Based Intrusion Detection

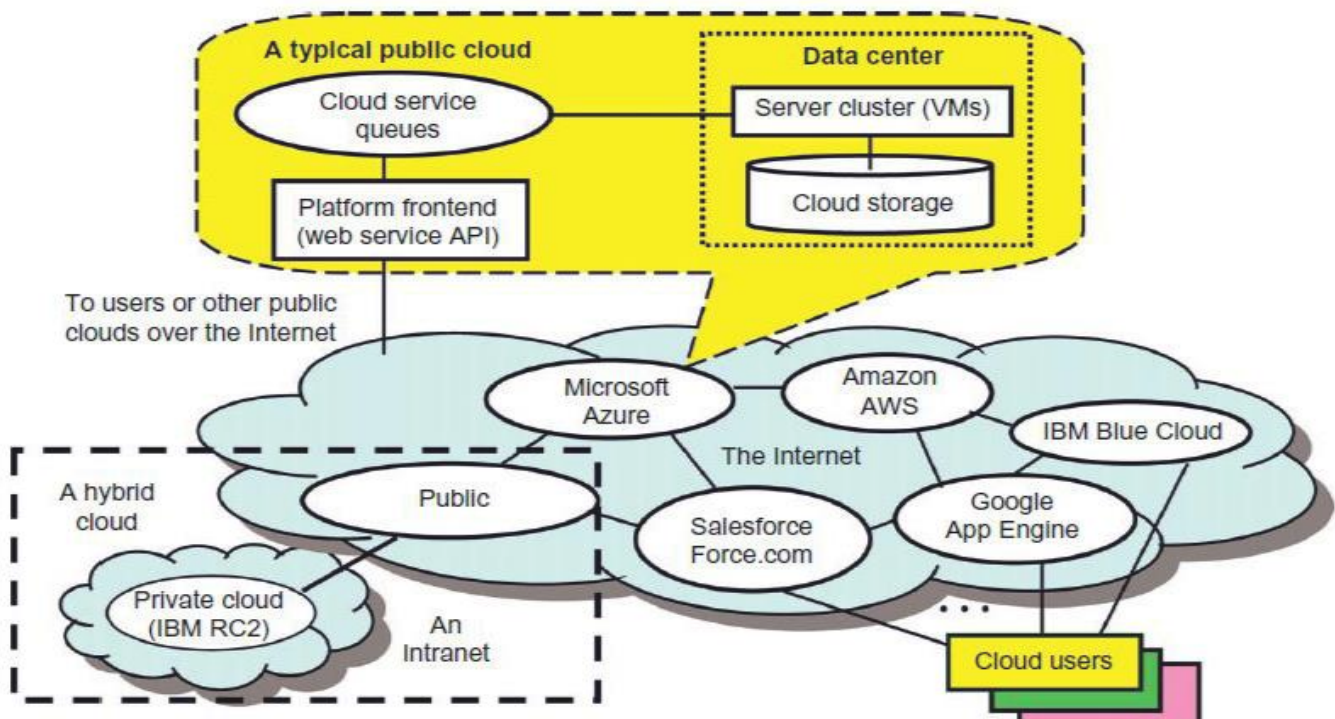
Virtualization-based intrusion detection can isolate guest VMs on the same hardware platform. An intrusion detection system (IDS) is built on operating systems, and is based on the characteristics of intrusion actions. A typical IDS can be classified as a host-based IDS (HIDS) or a network-based IDS (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.

UNIT- IIICLOUD INFRASTRUCTURETopics Covered:

- ② Cloud Computing and Service Models
- ② Data-Center Design and Interconnection Networks
- ② Architectural Design of Compute and Storage Clouds
- ② Layered Cloud Architecture Development
- ② Architectural Design Challenges
- ② Inter Cloud Resource Management
- ② Resource Provisioning and Platform Deployment
- ② Global Exchange of Cloud Resources

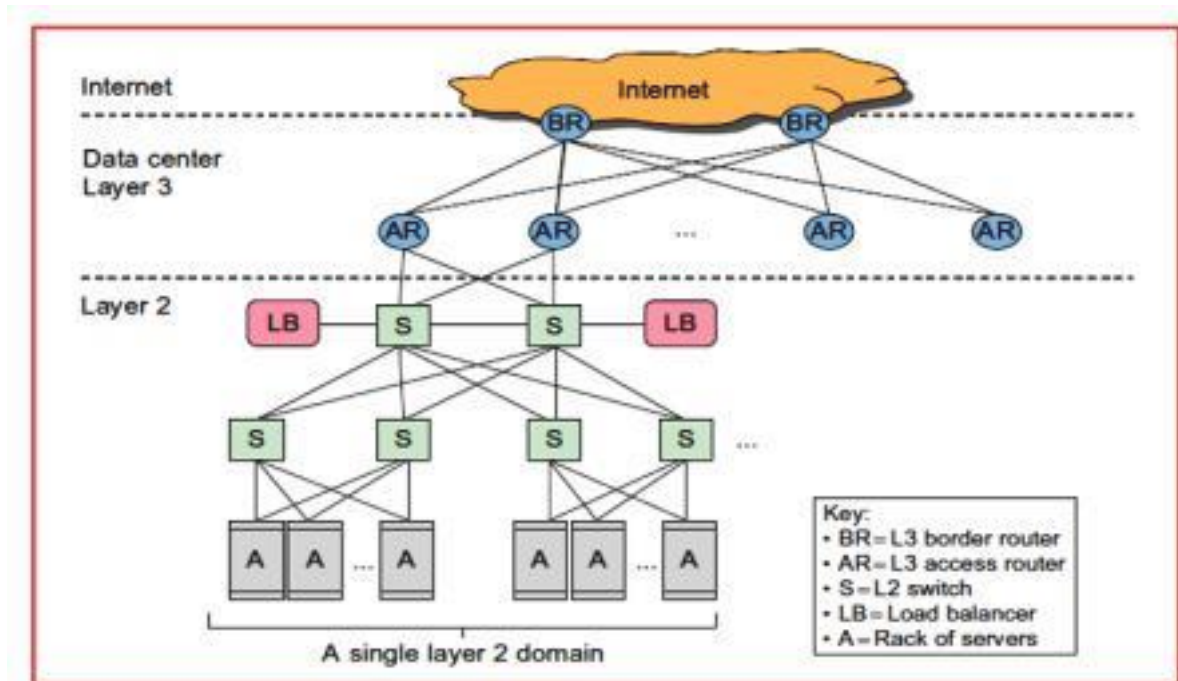
⇒ Cloud Computing and Service Models

- ✓ Clouds aim to power the next generation of data centers by architecting them as virtual resources over automated hardware, databases, user interfaces, and application environments.
 - ✓ 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.
 - ✓ A **Private cloud** is built within the domain of an **Intranet** owned by a **single organization**.
 - ✓ A **Hybrid cloud** is built with both public and private clouds.
- Public, Private, and Hybrid clouds illustrated by functional architecture and connectivity of representative clouds available.



➤ Data-Center Networking Structure

- Data-center server clusters are typically built with **large number of servers, ranging from thousands to millions of servers (nodes)**. For example, Microsoft has a data center in the Chicago area that has 100,000 eight-core servers, housed in 50 containers.
- Data-center networks are mostly **IP-based commodity networks**, such as the 10 Gbps Ethernet networks, which is optimized for Internet access. The server racks are at the bottom Layer 2, and they are connected through fast switches (S) as the hardware core. The data center is connected to the Internet at Layer 3 with many **access routers (ARs)** and **border routers (BRs)**. Figure shows a multilayer structure for accessing the Internet.



➤ Cloud Design Objectives: 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.

⇒ Data-Center Design and Interconnection Networks

- **“The cloud is built on massive datacenters”.** The data centers are built economics of scale— meaning **lower unit cost for larger data centers**. A small data center could have 1,000 servers. **The larger the data center, the lower the operational cost.** The approximate monthly cost to operate a huge 400-server data center is estimated by network cost \$13/Mbps; storage cost \$0.4/GB; and administration costs. These unit costs are greater than those of a 1,000-server data center. The network cost to operate a small data center is about seven times greater and the storage cost is 5.7 times greater. Microsoft has about 100 data centers, large or small, which are distributed around the globe.



Data-Center Construction Requirements

- Most data centers are built with commercially available components. An off-the-shelf server consists of **a number of processor sockets**, each with a **multi core CPU** and its internal cache hierarchy, local shared and coherent **DRAM**, and a number of directly attached **disk drives**. The DRAM and disk resources within the rack are accessible through first-level rack switches and all resources in all racks are accessible via a **cluster-level switch**. Consider a data center built with 2,000 servers, each with 8 GB of DRAM and four 1 TB disk drives. Each group of 40 servers is connected through a 1 Gbps link to a rack-level switch that has an additional eight 1 Gbps ports used for connecting the rack to the cluster-level switch.
- The data-center module is housed in a **truck-towable container**. The modular container design includes the **network, computer, storage, and cooling gear**. A modular data center built in a truck-towed ICE Cube container that can be cooled by chilled air circulation with cold-water heat exchanges.



Data-Center Management Issues: Here are basic requirements for managing the resources of a data center.

- **Making common users happy** The data center should be designed to provide quality service to the majority of users for at least 30 years.
- **Controlled information flow** Information flow should be streamlined. Sustained services and high availability (HA) are the primary goals.
- **Multiuser manageability** The system must be managed to support all functions of a data center, including traffic flow, database updating, and server maintenance.
- **Scalability to prepare for database growth** The system should allow growth as workload increases. The storage, processing, I/O, power, and cooling subsystems should be scalable.
- **Reliability in virtualized infrastructure** Failover, fault tolerance, and VM live migration should be integrated to enable recovery of critical applications from failures or disasters.

- **Low cost to both users and providers** The cost to users and providers of the cloud system built over the data centers should be reduced, including all operational costs.
- **Security enforcement and data protection** Data privacy and security defense mechanisms must be deployed to protect the data center against network attacks and system interrupts and to maintain data integrity from user abuses or network attacks.
- **Green information technology** Saving power consumption and upgrading energy efficiency are in high demand when designing and operating current and future data centers.

⇒ ARCHITECTURAL DESIGN OF COMPUTE AND STORAGE CLOUDS

➤ A Generic Cloud Architecture Design

An Internet cloud is envisioned as a **public cluster of servers** provisioned on demand to perform collective web services or distributed applications using **data-center resources**.

❖ Cloud Platform design objectives:

There four major design goals of a cloud computing platform:

- Scalability,
- Virtualization,
- Efficiency, and
- Reliability

❖ Enabling Technologies for Clouds

Table 4.3 Cloud-Enabling Technologies in Hardware, Software, and Networking

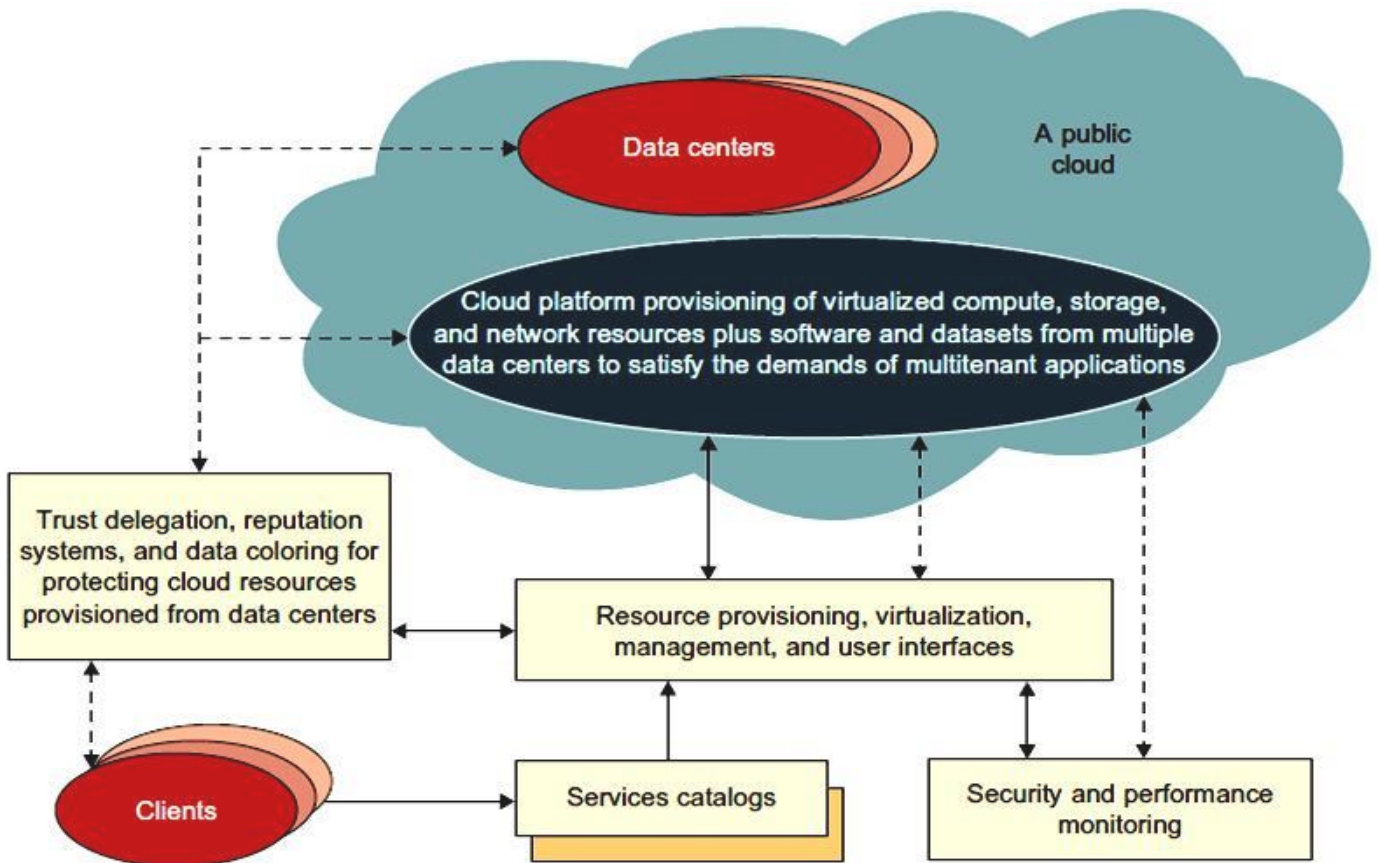
Technology	Requirements and Benefits
Fast platform deployment	Fast, efficient, and flexible deployment of cloud resources to provide dynamic computing environment to users
Virtual clusters on demand	Virtualized cluster of VMs provisioned to satisfy user demand and virtual cluster reconfigured as workload changes
Multitenant techniques	SaaS for distributing software to a large number of users for their simultaneous use and resource sharing if so desired
Massive data processing	Internet search and web services which often require massive data processing, especially to support personalized services
Web-scale communication	Support for e-commerce, distance education, telemedicine, social networking, digital government, and digital entertainment applications
Distributed storage	Large-scale storage of personal records and public archive information which demands distributed storage over the clouds
Licensing and billing services	License management and billing services which greatly benefit all types of cloud services in utility computing

➤ A Generic Cloud Architecture

Figure shows a security-aware cloud platform built with a **virtual cluster of VMs, storage, and networking resources over the data-center servers operated by providers**. The Internet cloud is envisioned as a massive cluster of servers. These servers are provisioned on demand to perform collective web services or distributed applications using data-center resources. The cloud platform is formed dynamically by provisioning or de provisioning servers, software, and database resources. Servers in the

cloud can be physical machines or VMs. User interfaces are applied to request services. The provisioning tool carves out the cloud system to deliver the requested service.

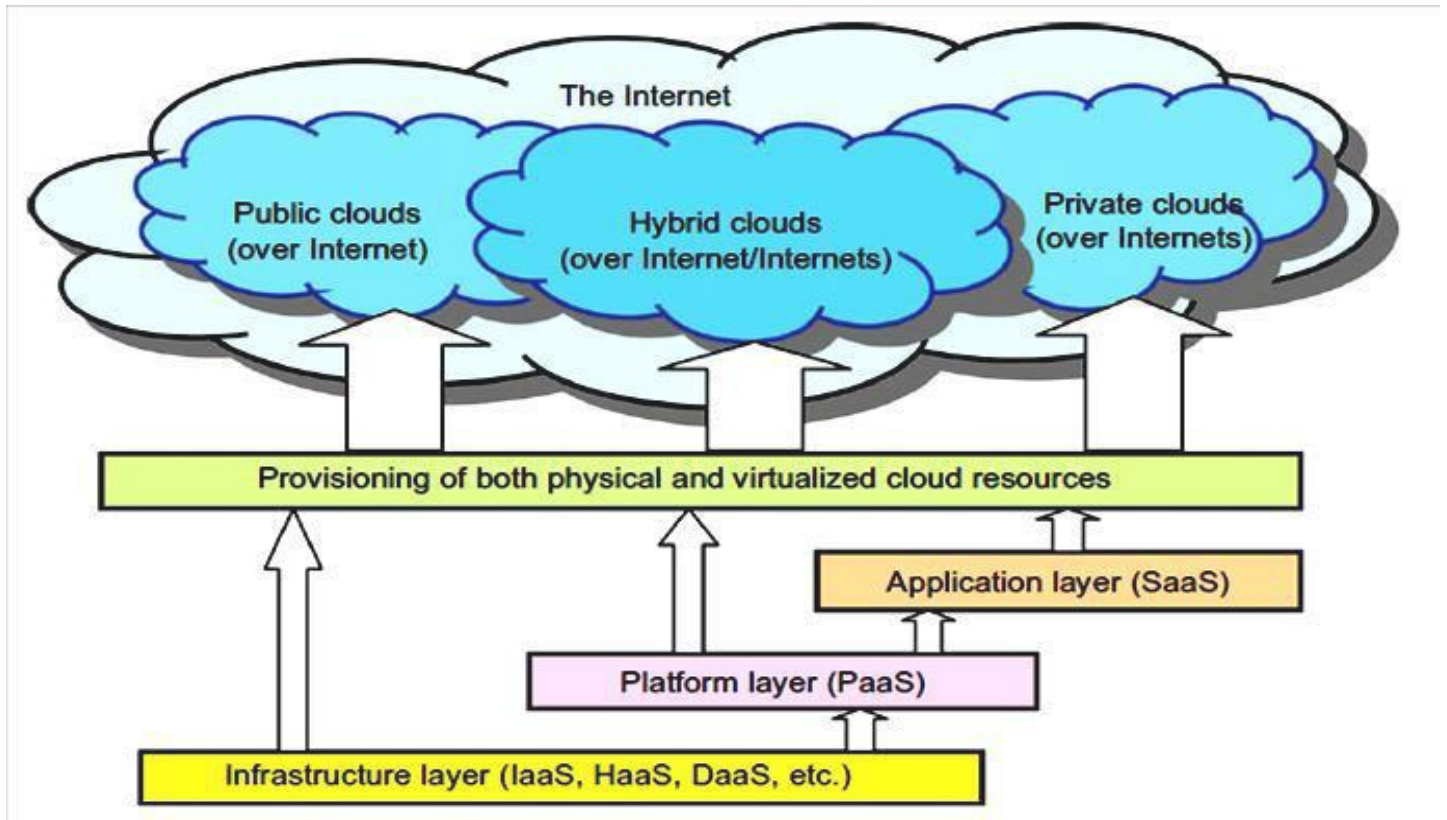
The cloud platform demands distributed storage and accompanying services. Other cloud resources are added into a cloud platform, including storage area networks (SANs), database systems, firewalls, and security devices.



⇒ Layered Cloud Architectural Development:

- The architecture of a cloud is developed at three layers: **infrastructure, platform, and application**, these three development layers are implemented with virtualization and standardization of hardware and software resources provisioned in the cloud. The services to public, private, and hybrid clouds are conveyed to users through networking support over the Internet and Intranets involved.
 - ▶ **The infrastructure layer** is deployed first to support **IaaS** services, is built with **virtualized compute, storage, and network resources**.
 - ▶ **The platform layer** is a foundation for implementing the application layer for **SaaS** applications. The platform layer is for general-purpose and repeated usage of the collection of software resources. This layer provides users with an **environment to develop their applications, to test operation flows, and to monitor execution results and performance**.
 - ▶ **The application layer** is formed with a collection of all needed software modules for SaaS applications. Service applications in this layer include **daily office management work, such as information retrieval, document processing, and calendar and authentication services**. The

application layer is also heavily used by enterprises in **business marketing and sales**, **consumer relationship management (CRM)**, **financial transactions**, and **supply chain management**.

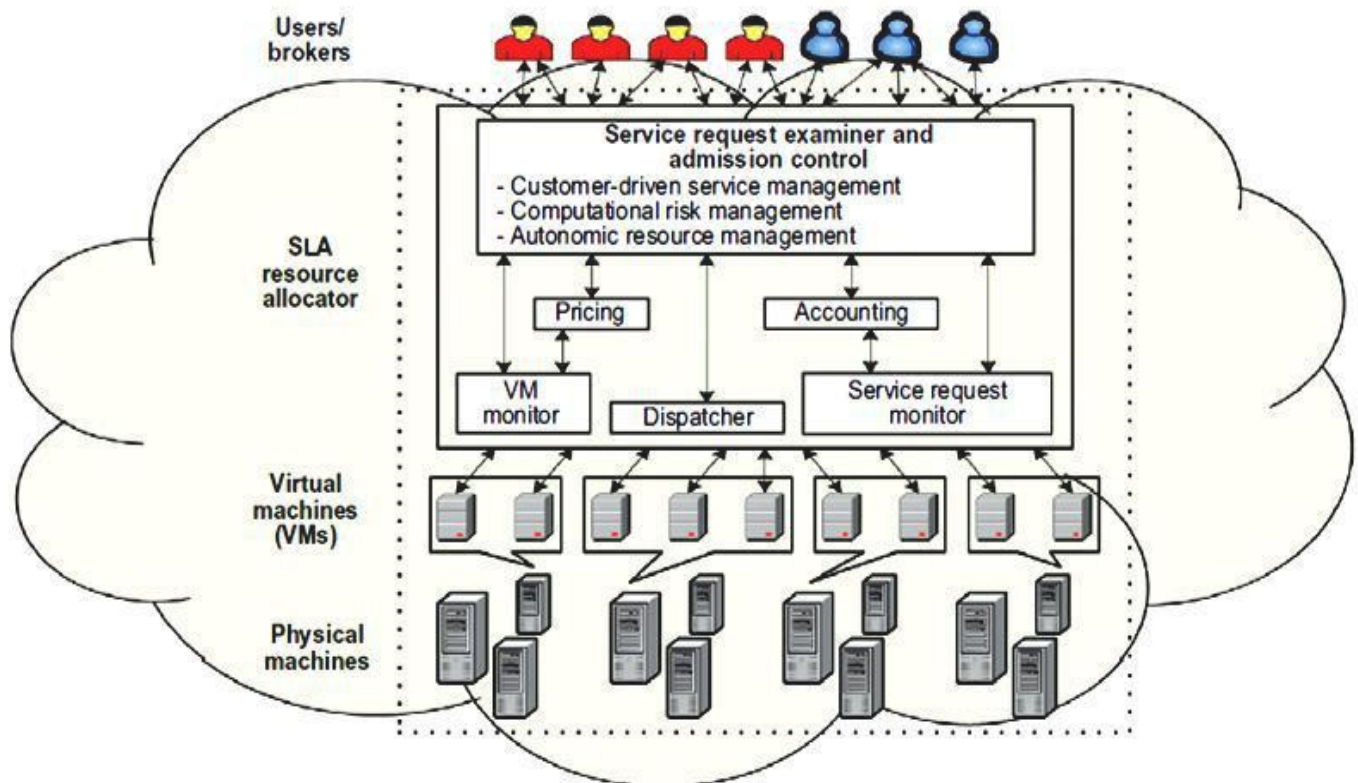


❖ Market-Oriented Cloud Architecture

- The below figure shows the high-level architecture for supporting **market-oriented resource allocation** in a cloud computing environment. This cloud is basically built with the following entities:
 - ▶ **Users or brokers** acting on user's behalf submit service requests from anywhere in the world to the data center and cloud to be processed.
 - ▶ **The SLA resource allocator** acts as the interface between the data center/cloud service provider and external users/brokers.
 - ▶ **The request examiner** ensures that there is no overloading of resources whereby many service requests cannot be fulfilled successfully due to limited resources. **The Pricing mechanism** decides how service requests are charged.
 - ▶ **The VM Monitor** mechanism keeps track of the availability of VMs and their resource entitlements.
 - ▶ **The Dispatcher** mechanism starts the execution of accepted service requests on allocated VMs.
 - ▶ **The Service Request Monitor** mechanism keeps track of the execution progress of service requests.

- ▶ **Multiple VMs** can be started and stopped on demand on a single physical machine to meet accepted service requests, hence providing maximum flexibility to configure various partitions of resources on the same physical machine to different specific requirements of service requests.
- ▶ **Quality of Service Factors:** there are critical QOS parameters to consider in a service request, such as time, cost, reliability, and trust/security.

Market-oriented cloud architecture to expand/shrink leasing of resources with variation in QOS/demand from users.



❖ Virtualization Support and Disaster Recovery

One very distinguishing feature of cloud computing infrastructure is the use of system **virtualization** and the modification to provisioning tools.

- ▶ Hardware Virtualization -Virtualized servers, storage, and network for cloud platform construction.
- ▶ Virtualization Support in Public Clouds
- ▶ Storage Virtualization for Green Data Centers
- ▶ Virtualization for IaaS
- ▶ VM Cloning for Disaster Recovery

⇒ **Architectural Design Challenges:** six open challenges in cloud architecture development

1) Challenge 1—Service Availability and Data Lock-in Problem

The source of single points of failure, to achieve high availability one can consider using multiple cloud providers. Using multiple cloud providers may provide more protection from failures. Another availability obstacle is distributed denial of service (DDoS) attacks. Data lock-in concerns, standardization of APIs enables a new usage model in which the same software infrastructure can be used in both public and private clouds.

2) Challenge 2—Data Privacy and Security Concerns

Traditional network attacks include buffer overflows, DoS attacks, spyware, malware, rootkits, Trojan horses, and worms. In a cloud environment, newer attacks may result from hypervisor malware, guest hopping and hijacking, or VM rootkits. Passive attacks steal sensitive data or passwords. Active attacks may manipulate kernel data structures which will cause major damage to cloud servers.

3) Challenge 3—Unpredictable Performance and Bottlenecks

Multiple VMs can share CPUs and main memory in cloud computing, but I/O sharing is problematic. One solution is to **improve I/O architectures and operating systems** to efficiently virtualize interrupts and I/O channels. Therefore, data transfer bottlenecks must be removed, bottleneck links must be widened, and weak servers should be removed.

4) Challenge 4—Distributed Storage and Widespread Software Bugs

The opportunity is to create a storage system that will not only meet this growth, but also combine it with the cloud advantage of scaling arbitrarily up and down on demand. Data centers must meet programmers' expectations in terms of scalability, data durability, and HA. Large-scale distributed bugs cannot be reproduced, so the debugging must occur at a scale in the production data centers.

5) Challenge 5—Cloud Scalability, Interoperability, and Standardization

The pay-as-you-go model applies to storage and network bandwidth; both are counted in terms of the number of bytes used. Computation is different depending on virtualization level. Google App Engine (GAE) automatically scales in response to load increases and decreases; users are charged by the cycles used. Open Virtualization Format (OVF) describes an open, secure, portable, efficient, and extensible format for the packaging and distribution of VMs. In terms of cloud standardization, we suggest the ability for virtual appliances to run on any virtual platform. We also need to enable VMs to run on heterogeneous hardware platform hypervisors.

6) Challenge 6—Software Licensing and Reputation Sharing

Many cloud computing providers originally relied on open source software because the licensing model for commercial software is not ideal for utility computing. One can consider using both **pay-for-use** and **bulk-use licensing schemes** to widen the business coverage. One customer's bad behavior can affect the reputation of the entire cloud. An opportunity would be to create reputation-guarding services similar to the "trusted e-mail" services currently offered (for a fee) to services hosted on smaller ISPs.

⇒ Inter Cloud Resource Management

Here we discuss the various cloud service models and their extensions. The cloud service trends are outlined. Cloud resource management and inter cloud resource exchange schemes are reviewed.

➤ **Extended Cloud Computing Services:**

Figure shows **six layers of cloud services**, ranging from hardware, network, and collocation to infrastructure, platform, and software applications. We already introduced the top three service layers as **SaaS, PaaS, and IaaS**, respectively. The bottom three layers are more related to physical requirements.

The bottommost layer provides **Hardware as a Service (HaaS)**. The next layer is for interconnecting all the hardware components, and is simply called **Network as a Service (NaaS)**. Virtual LANs fall within the scope of NaaS. The next layer up offers **Location as a Service (Laas)**, which provides a collocation service to house, power, and secure all the physical hardware and network resources. Some authors say this layer provides Security as a Service (“SaaS”). The cloud infrastructure layer can be further subdivided as **Data as a Service (DaaS)** and **Communication as a Service (CaaS)** in addition to compute and storage in IaaS.

Cloud application (SaaS)			Concur, RightNOW, Teleo, Kenexa, Webex, Blackbaud, salesforce.com, Netsuite, Kenexa, etc.
Cloud software environment (PaaS)			Force.com, App Engine, Facebook, MS Azure, NetSuite, IBM BlueCloud, SGI Cyclone, eBay
Cloud software infrastructure			Amazon AWS, OpSource Cloud, IBM Ensembles, Rackspace cloud, Windows Azure, HP, Banknorth
Computational resources (IaaS)	Storage (DaaS)	Communications (CaaS)	
Collocation cloud services (Laas)			Savvis, Internap, NTTCommunications, Digital Realty Trust, 365 Main
Network cloud services (NaaS)			Owest, AT&T, AboveNet
Hardware/Virtualization cloud services (HaaS)			VMware, Intel, IBM, XenEnterprise

The cloud players are divided into three classes: **(1) cloud service providers and IT administrators**, **(2) software developers or vendors**, and **(3) end users or business users**. These cloud players vary in their roles under the IaaS, PaaS, and SaaS models. The table entries distinguish the three cloud models as viewed by different players.

Table 4.7 Cloud Differences in Perspectives of Providers, Vendors, and Users

Cloud Players	IaaS	PaaS	SaaS
IT administrators/cloud providers	Monitor SLAs	Monitor SLAs and enable service platforms	Monitor SLAs and deploy software
Software developers (vendors)	To deploy and store data	Enabling platforms via configurators and APIs	Develop and deploy software
End users or business users	To deploy and store data	To develop and test web software	Use business software



Cloud Service Tasks and Trends

Cloud services are introduced in five layers.

The top layer is for SaaS applications, as further subdivided into the five application areas in above Figure, mostly for business applications. For example, CRM is heavily practiced in business promotion, direct sales, and marketing services. CRM offered the first SaaS on the cloud successfully.

PaaS is provided by Google, Salesforce.com, and Facebook, among others. IaaS is provided by Amazon, Windows Azure, and RackRack, among others.

Network cloud services provide communications such as those by AT&T, Qwest, and AboveNet.



Software Stack for Cloud Computing

Despite the various types of nodes in the cloud computing cluster, the overall software stacks are built from scratch to meet rigorous goals (see above Table). By using VMs, the platform can be flexible, that is, the running services are not bound to specific hardware platforms. Other layers running on top of the file system are the layers for executing cloud computing applications. They include the **database storage system, programming for large-scale clusters, and data query language support**. The next layers are the components in the software stack.



Runtime Support Services

Cluster monitoring is used to collect the runtime status of the entire cluster. One of the most important facilities is the cluster job management system. The scheduler queues the tasks submitted to the whole cluster and assigns the tasks to the processing nodes according to node availability. Runtime support is software needed in browser-initiated applications applied by thousands of cloud customers. The **SaaS** model provides the **software applications as a service**, rather than letting users purchase the software.

⇒ Resource Provisioning and Platform Deployment

In this section, we will discuss techniques to **provision computer resources** or **VMs**.



Provisioning of Compute Resources (VMs)

Providers supply cloud services by signing **SLAs** with end users. The SLAs must commit sufficient resources such as **CPU, memory, and bandwidth** that the user can use for a preset period. Under provisioning of resources will lead to broken SLAs and penalties. Over provisioning of resources will lead to resource underutilization, and consequently, a decrease in revenue for the provider.

Efficient VM provisioning depends on the **cloud architecture and management of cloud infrastructures**. Resource provisioning schemes also demand fast discovery of services and data in cloud computing infrastructures. Public or Private clouds promise to streamline the on-demand provisioning of software, hardware, and data as a service, achieving economies of scale in IT deployment and operation.



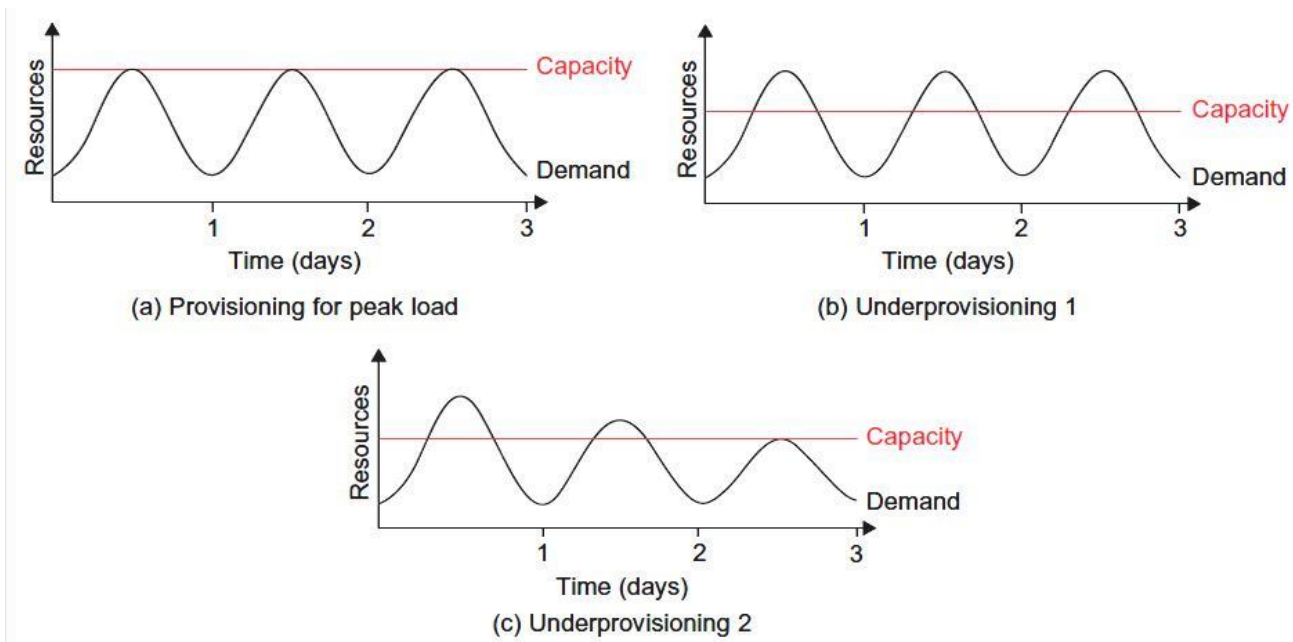
Resource-Provisioning Methods

Figure shows **three cases of static cloud resource provisioning policies**.

In case (a), **over provisioning** with the peak load causes heavy resource waste (shaded area).

In case (b), **under provisioning** (along the capacity line) of resources results in losses by both user and provider in that paid demand by the users (the shaded area above the capacity) is not served and wasted resources still exist for those demanded areas below the provisioned capacity.

In case (c), **under- and then over provisioning**, the constant provisioning of resources with fixed capacity to a declining user demand could result in even worse resource waste. The user may give up the service by canceling the demand, resulting in reduced revenue for the provider. Both the user and provider may be losers in resource provisioning without elasticity.



Three resource-provisioning methods are presented in the following sections.

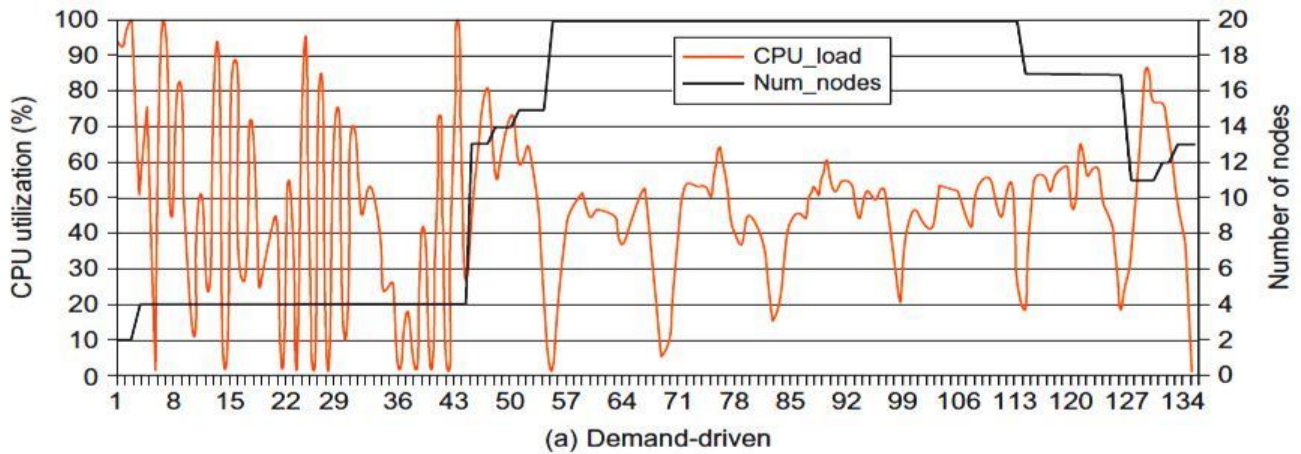
- The **demand-driven method** provides static resources and has been used in grid computing for many years.
- The **event driven method** is based on predicted workload by time.
- The **popularity-driven method** is based on Internet traffic monitored.

1) Demand-Driven Resource Provisioning

This method adds or removes computing instances **based on the current utilization level of the allocated resources**. The demand-driven method automatically allocates two Xeon processors for the user application, when the user was using one Xeon processor more than 60 percent of the time for an extended period. In general, when a resource has surpassed a threshold for a certain amount of time, the scheme increases that resource based on demand.

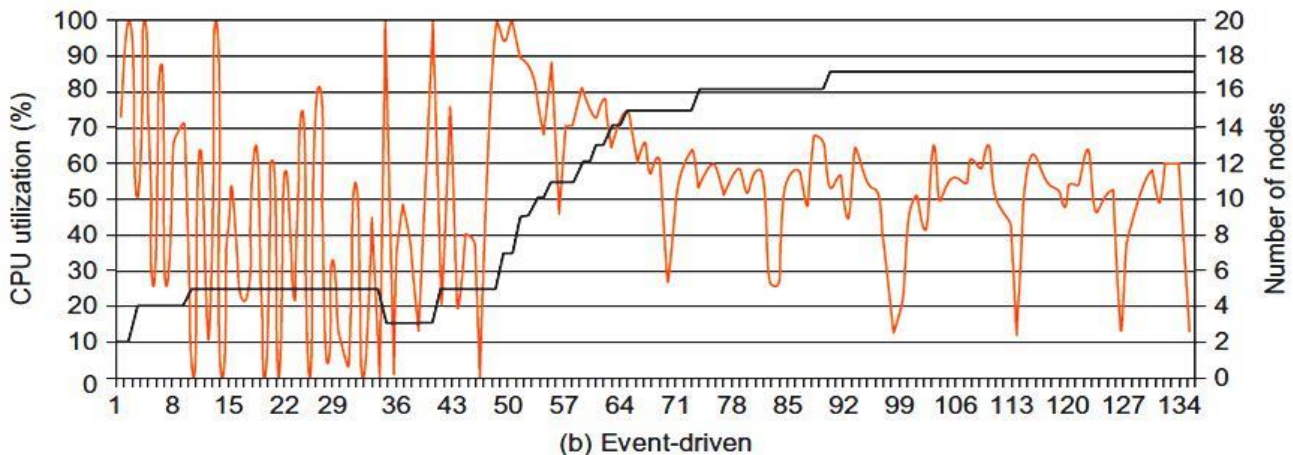
The x-axis in Figure is the time scale in milliseconds. In the beginning, **heavy fluctuations of CPU load** are encountered. All three methods have demanded a few VM instances initially.

Gradually, the utilization rate becomes more stabilized with a maximum of 20 VMs (100 percent utilization) provided for demand-driven provisioning in Figure (a). However, the event-driven method reaches a stable peak of 17 VMs toward the end of the event and drops quickly in Figure (b). The popularity provisioning shown in Figure (c) leads to a similar fluctuation with peak VM utilization in the middle of the plot.



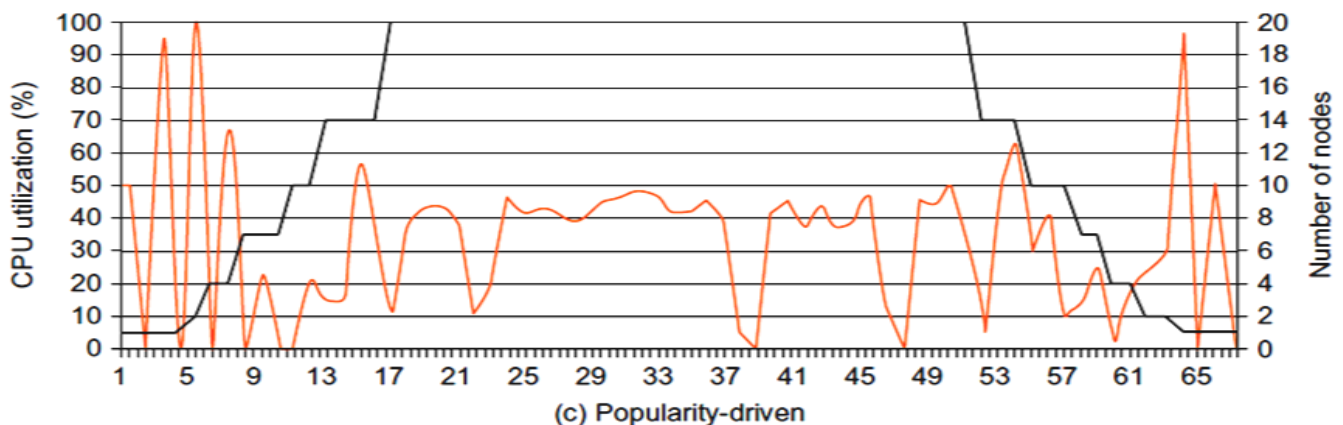
2) Event-Driven Resource Provisioning

This scheme **adds or removes machine instances based on a specific time event**. This scheme anticipates peak traffic before it happens. The method results in a minimal loss of QoS, if the event is predicted correctly. Otherwise, wasted resources are even greater due to events that do not follow a fixed pattern.



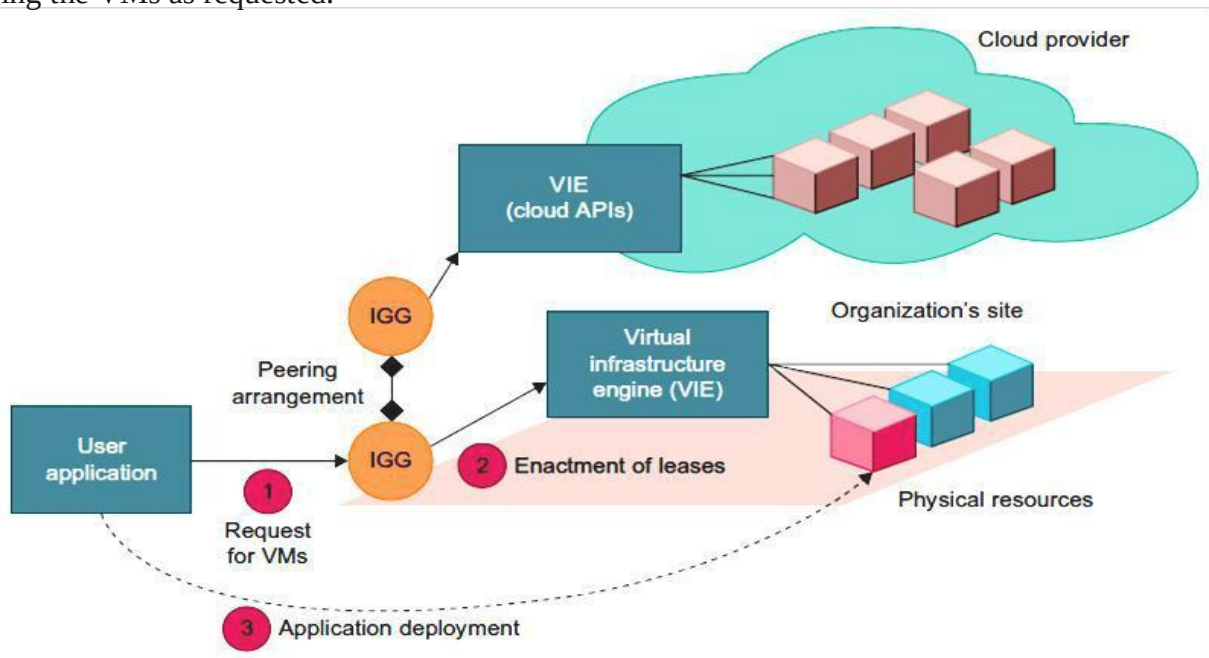
3) Popularity-Driven Resource Provisioning

In this method, the **Internet searches for popularity of certain applications and creates the instances by popularity demand**. The scheme anticipates increased traffic with popularity. Again, the scheme has a minimal loss of QoS, if the predicted popularity is correct. Resources may be wasted if traffic does not occur as expected.



➤ Dynamic Resource Deployment

The cloud uses VMs as building blocks to create an execution environment across multiple resource sites. The **Inter-Grid** is a **Java-implemented software system that lets users create execution cloud environments on top of all participating grid resources**. Peering arrangements established between gateways enable the allocation of resources from multiple grids to establish the execution environment. In [Figure](#), a scenario is illustrated by which an Inter Grid Gateway (IGG) allocates resources from a local cluster to deploy applications in three steps: (1) requesting the VMs, (2) enacting the leases, and (3) deploying the VMs as requested.



➤ Provisioning of Storage Resources

The data storage layer is built on top of the physical or virtual servers. As the cloud computing applications often provide service to users, it is unavoidable that the data is stored in the clusters of the cloud provider. One example is **e-mail systems**.

A distributed file system is very important for **storing large-scale data**. Many cloud computing companies have developed large-scale data storage systems to keep huge amount of data collected every day. For example, see the below table:

Table 4.8 Storage Services in Three Cloud Computing Systems

Storage System	Features
GFS: Google File System	Very large sustainable reading and writing bandwidth, mostly continuous accessing instead of random accessing. The programming interface is similar to that of the POSIX file system accessing interface.
HDFS: Hadoop Distributed File System	The open source clone of GFS. Written in Java. The programming interfaces are similar to POSIX but not identical.
Amazon S3 and EBS	S3 is used for retrieving and storing data from/to remote servers. EBS is built on top of S3 for using virtual disks in running EC2 instances.

⇒ Global Exchange of Cloud Resources

- ❖ In order to support a large number of application service consumers from around the world, cloud infrastructure providers (i.e., IaaS providers) have established data centers in multiple geographical locations to provide redundancy and ensure reliability in case of site failures.
- ❖ For example, **Amazon** has data centers in the United States (e.g., one on the East Coast and another on the West Coast) and Europe. However, currently Amazon expects its cloud customers (i.e., SaaS providers) to express a preference regarding where they want their application services to be hosted. Amazon does not provide seamless/automatic mechanisms for scaling its hosted services across multiple geographically distributed data centers.
- ❖ This approach has many shortcomings. First, it is difficult for cloud customers to determine in advance the best location for hosting their services as they may not know the origin of consumers of their services. Second, SaaS providers may not be able to meet the QoS expectations of their service consumers originating from multiple geographical locations.
- ❖ This necessitates building mechanisms for seamless federation of data centers of a cloud provider or providers supporting dynamic scaling of applications across multiple domains in order to meet QoS targets of cloud customers.
- ❖ Cloud infrastructure provider would like to make use of services of multiple cloud infrastructure service providers who can provide better support for their specific consumer needs. This kind of requirement often arises in enterprises with global operations and applications such as Internet services, media hosting, and Web 2.0 applications.

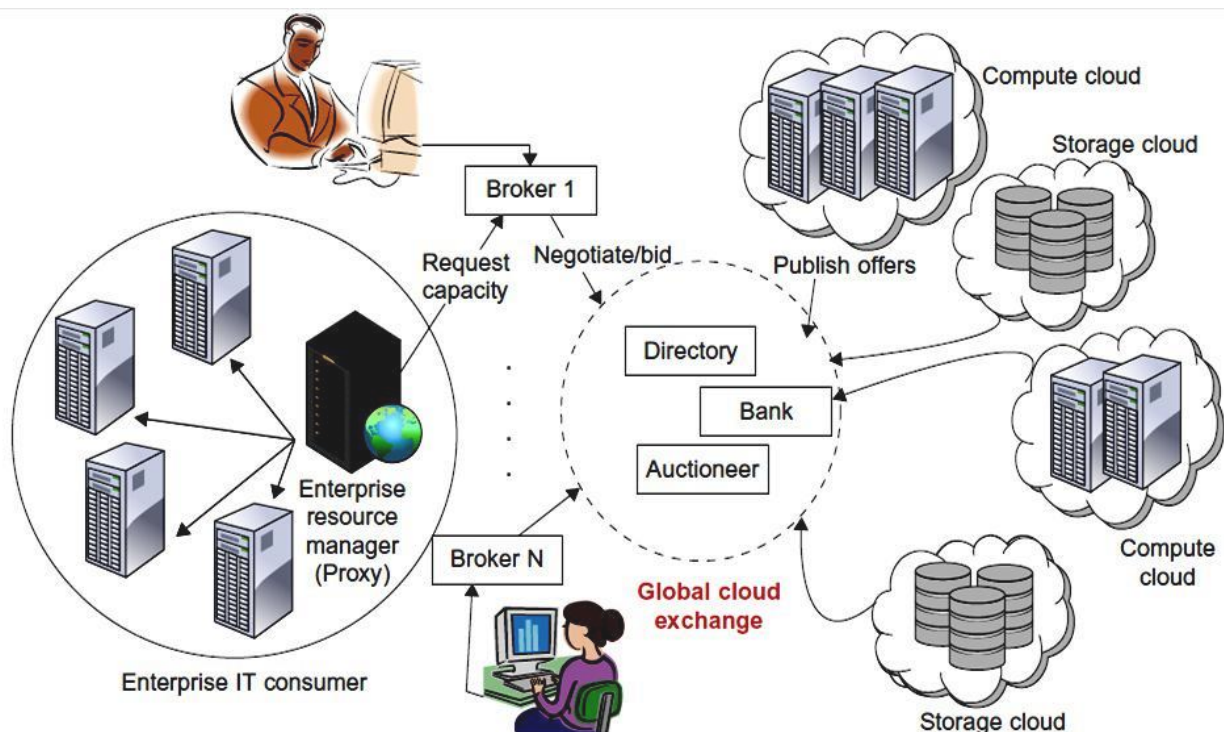


Fig: Inter-cloud exchange of cloud resources through brokering

- ❖ Cloud providers will be able to dynamically expand or resize their provisioning capability based on sudden spikes in workload demands by leasing available computational and storage capabilities from other cloud service providers; operate as part of a market-driven resource leasing federation, where application service providers such as [Salesforce.com](https://www.salesforce.com) host their services based on negotiated SLA contracts driven by competitive market prices; and deliver on-demand, reliable, cost-effective, and QoS-aware services based on virtualization technologies while ensuring high QoS standards and minimizing service costs. They need to be able to utilize market-based utility models as the basis for provisioning of virtualized software services and federated hardware infrastructure among users with heterogeneous applications.
- ❖ They consist of client brokering and coordinator services that support utility-driven federation of clouds: application scheduling, resource allocation, and migration of workloads. The architecture cohesively couples the administratively and topologically distributed storage and compute capabilities of clouds as part of a single resource leasing abstraction. The system will ease the cross domain capability integration for on-demand, flexible, energy-efficient, and reliable access to the infrastructure based on virtualization technology.
- ❖ The Cloud Exchange (CEX) acts as a market maker for bringing together service producers and consumers. It aggregates the infrastructure demands from application brokers and evaluates them against the available supply currently published by the cloud coordinators.
- ❖ An SLA specifies the details of the service to be provided in terms of metrics agreed upon by all parties, and incentives and penalties for meeting and violating the expectations, respectively.

UNIT- IVCLOUD PROGRAMMING MODELTopics Covered:

- ② Features of Cloud and Grid Platforms
- ② Parallel and Distributed Programming Paradigms
- ② Mapping Applications to Parallel and Distributed Systems
- ② Programming Support of Google App Engine
- ② Programming on Amazon AWS
- ② Emerging Cloud Software Environments
 - ▶ Open Source Eucalyptus OpenNebula Sector/Sphere
 - ▶ OpenStack Manjrasoft Aneka Cloud CloudSim

⇒ Features of Cloud and Grid Platforms

We summarize important features in real cloud and grid platforms. In four tables, we cover the capabilities, traditional features, data features, and features for programmers and runtime systems to use.

Table 6.1 Important Cloud Platform Capabilities

Capability	Description
Physical or virtual computing platform	The cloud environment consists of some physical or virtual platforms. Virtual platforms have unique capabilities to provide isolated environments for different applications and users.
Massive data storage service, distributed file system	With large data sets, cloud data storage services provide large disk capacity and the service interfaces that allow users to put and get data. The distributed file system offers massive data storage service. It can provide similar interfaces as local file systems.
Massive database storage service	Some distributed file systems are sufficient to provide the underlying storage service application developers need to save data in a more semantic way. Just like DBMS in the traditional software stack, massive database storage services are needed in the cloud.
Massive data processing method and programming model	Cloud infrastructure provides thousands of computing nodes for even a very simple application. Programmers need to be able to harness the power of these machines without considering tedious infrastructure management issues such as handling network failure or scaling the running code to use all the computing facilities provided by the platforms.
Workflow and data query language support	The programming model offers abstraction of the cloud infrastructure. Similar to the SQL language used for database systems, in cloud computing, providers have built some workflow language as well as data query language to support better application logic.
Programming interface and service deployment	Web interfaces or special APIs are required for cloud applications: J2EE, PHP, ASP, or Rails. Cloud applications can use Ajax technologies to improve the user experience while using web browsers to access the functions provided. Each cloud provider opens its programming interface for accessing the data stored in massive storage.
Runtime support	Runtime support is transparent to users and their applications. Support includes distributed monitoring services, a distributed task scheduler, as well as distributed locking and other services. They are critical in running cloud applications.
Support services	Important support services include data and computing services. For example, clouds offer rich data services and interesting data parallel execution models like MapReduce.

Table 6.3 Traditional Features in Cluster, Grid, and Parallel Computing Environments

Cluster management: ROCKS and packages offering a range of tools to make it easy to bring up clusters

Data management: Included metadata support such as RDF triple stores (Semantic web success and can be built on MapReduce as in SHARD); SQL and NOSQL included in

Grid programming environment: Varies from link-together services as in Open Grid Services Architecture (OGSA) to GridRPC (Ninf, GridSolve) and SAGA

OpenMP/threading: Can include parallel compilers such as Cilk; roughly shared memory technologies. Even transactional memory and fine-grained data flow come here

Portals: Can be called (science) gateways and see an interesting change in technology from portlets to HUBzero and now in the cloud: Azure Web Roles and GAE

Scalable parallel computing environments: MPI and associated higher level concepts including ill-fated HP FORTRAN, PGAS (not successful but not disgraced), HPCS languages (X-10, Fortress, Chapel), patterns (including Berkeley dwarves), and functional languages such as F# for distributed memory

Virtual organizations: From specialized grid solutions to popular Web 2.0 capabilities such as Facebook

Workflow: Supports workflows that link job components either within or between grids and clouds; relate to LIMS Laboratory Information Management Systems.

➤ Data Features and Databases

- **Program Library:** Many efforts have been made to design a VM image library to manage images used in academic and commercial clouds.
- **Blobs and Drives:** This concept is similar to shared file systems such as Lustre used in TeraGrid. The cloud storage is intrinsically fault-tolerant while that on TeraGrid needs backup storage.
- **DPFS:** In general, data transport will be needed to link these two data views. It seems important to consider this carefully, as DPFS file systems are precisely designed for efficient execution of data-intensive applications.
- **SQL and Relational Databases:** Both Amazon and Azure clouds offer relational databases and it is straightforward for academic systems to offer a similar capability unless there are issues of huge scale where, in fact, approaches based on tables and/or MapReduce might be more appropriate.
- **Table and NOSQL Non Relational Databases:** A substantial number of important developments have occurred regarding simplified database structures—termed “NOSQL”—typically emphasizing distribution and scalability. These are present in the three major clouds: BigTable in Google, SimpleDB in Amazon, and Azure Table for Azure.
- **Queuing Services:** Both Amazon and Azure offer similar scalable, robust queuing services that are used to communicate between the components of an application. The messages are short (less than 8 KB) and have a Representational State Transfer (REST) service interface with “deliver at least once” semantics.

➤ Programming and Runtime Support

- Programming and runtime support are desired to facilitate parallel programming and provide runtime support of important functions in today’s grids and clouds.
- Here, **MapReduce** is used to group the processes in clouds to reduce individual process mapping. Example: Google MapReduce, Apache Hadoop uses MapReduce

- The major open source/commercial MapReduce implementations are Hadoop and Dryad with execution possible with or without VMs.
- We desire a SaaS environment that provides many useful tools to develop cloud applications over large data sets. In addition to the technical features, such as MapReduce, BigTable, EC2, S3, Hadoop, AWS, GAE, and WebSphere2, we need protection features that may help us to achieve scalability, security, privacy, and availability.

⇒ Parallel and Distributed Programming Paradigms

- Consider a distributed computing system consisting of a set of networked nodes or workers. The system issues for running a typical parallel program in either a parallel or a distributed manner would include the following,
 - **Partitioning:** This is applicable to both computation and data as follows:
 - **Computation partitioning** This splits a given job or a program into smaller tasks. Partitioning greatly depends on correctly identifying portions of the job or program that can be performed concurrently. In other words, upon identifying parallelism in the structure of the program, it can be divided into parts to be run on different workers. Different parts may process different data or a copy of the same data.
 - **Data partitioning** This splits the input or intermediate data into smaller pieces. Similarly, upon identification of parallelism in the input data, it can also be divided into pieces to be processed on different workers. Data pieces may be processed by different parts of a program or a copy of the same program.
 - **Mapping:** This assigns the either smaller parts of a program or the smaller pieces of data to underlying resources. This process aims to appropriately assign such parts or pieces to be run simultaneously on different workers and is usually handled by resource allocators in the system.
 - **Synchronization:** Because different workers may perform different tasks, synchronization and coordination among workers is necessary so that race conditions are prevented and data dependency among different workers is properly managed.
 - **Communication:** Because data dependency is one of the main reasons for communication among workers, communication is always triggered when the intermediate data is sent to workers.
 - **Scheduling:** For a job or program, when the number of computation parts (tasks) or data pieces is more than the number of available workers, a scheduler selects a sequence of tasks or data pieces to be assigned to the workers. It is worth noting that the resource allocator performs the actual mapping of the computation or data pieces to workers, while the scheduler only picks the next part from the queue of unassigned tasks based on a set of rules called the scheduling policy. For multiple jobs or programs, a scheduler selects a sequence of jobs or programs to be run on the distributed computing system.

Because handling the whole data flow of parallel and distributed programming is very time consuming and requires specialized knowledge of programming, dealing with these issues may affect the productivity of the programmer and may even result in affecting the program's time to market. Therefore, parallel and distributed programming paradigms or models are offered to abstract many parts of the data flow from

users to provide users with an abstraction layer to hide implementation details of the data flow which users formerly ought to write codes for.

Other motivations behind parallel and distributed programming models are

- (1) to improve productivity of programmers,
- (2) to decrease programs' time to market,
- (3) to leverage underlying resources more efficiently,
- (4) to increase system throughput, and
- (5) to support higher levels of abstraction.

MapReduce, Hadoop, and Dryad are three of the most recently proposed parallel and distributed programming models. They were developed for information retrieval applications but have been shown to be applicable for a variety of important applications

➤ MapReduce, Twister, and Iterative MapReduce

MapReduce is a software framework which supports parallel and distributed computing on large data sets. This software framework abstracts the data flow of running a parallel program on a distributed computing system by providing users with two interfaces in the form of two functions: Map and Reduce. Users can override these two functions to interact with and manipulate the data flow of running their programs.

Given below FIGURE illustrates the logical data flow from the Map to the Reduce function in MapReduce frameworks. In this framework, the “value” part of the data, (key, value), is the actual data, and the “key” part is only used by the MapReduce controller to control the data flow.

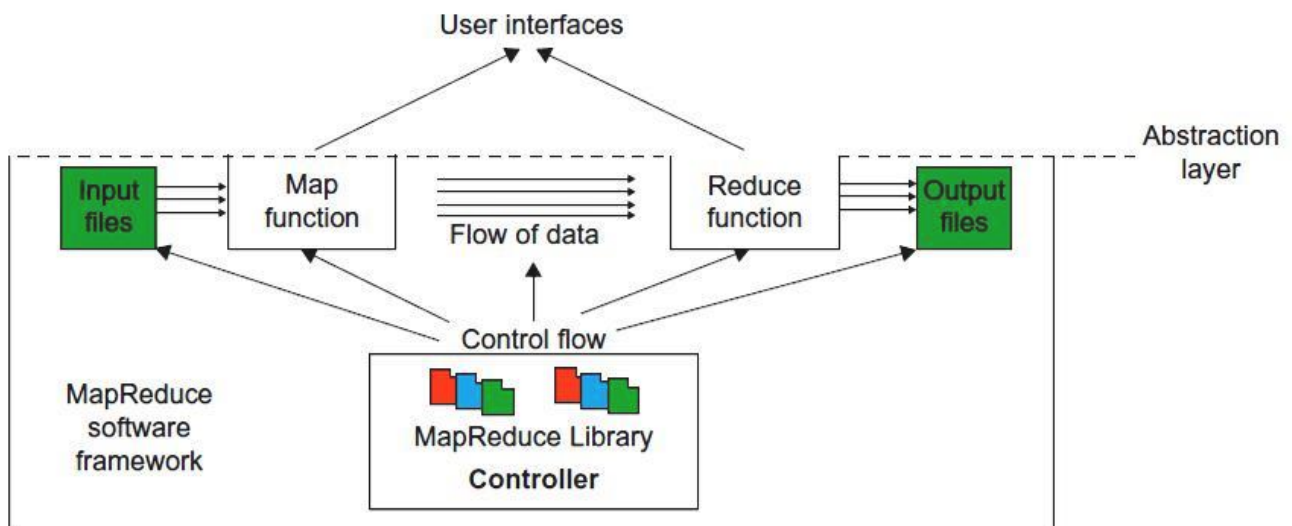


Figure: MapReduce framework: Input data flows through the Map and Reduce functions to generate the output result under the control flow using MapReduce software library. Special user interfaces are used to access the Map and Reduce resources.

Formal Definition of MapReduce

The MapReduce software framework provides an abstraction layer with the data flow and flow of control to users, and hides the implementation of all data flow steps such as data partitioning, mapping, synchronization, communication, and scheduling. The abstraction layer provides two well-defined interfaces in the form of two functions: Map and Reduce. The user overrides the Map and Reduce functions first and then invokes the provided MapReduce (Spec, & Results) function from the library to start the flow of data. The MapReduce function, MapReduce (Spec, & Results), takes an important parameter which is a

specification object, the Spec. This object is first initialized inside the user's program, and then the user writes code to fill it with the names of input and output files, as well as other optional tuning parameters. This object is also filled with the name of the Map and reduce functions to identify these userdefined functions to the MapReduce library.

The overall structure of a user's program containing the Map, Reduce, and the Main functions is given below. The Map and Reduce are two major subroutines. They will be called to implement the desired function performed in the main program.

```

Map Function (... )
{
... ..
}
Reduce Function (... )
{
... ..
}
Main Function (... )
{
Initialize Spec object
... ..
MapReduce (Spec, & Results)
}

```

MapReduce Logical Data Flow

The input data to the Map function is in the form of a (key, value) pair. For example, the key is the line offset within the input file and the value is the content of the line. The output data from the Map function is structured as (key, value) pairs called intermediate (key, value) pairs. In other words, the user-defined Map function processes each input (key, value) pair and produces a number of (zero, one, or more) intermediate (key, value) pairs. Here, the goal is to process all input (key, value) pairs to the Map function in parallel. The Reduce function receives the intermediate (key, value) pairs in the form of a group of intermediate values associated with one intermediate key, (key, [set of values]). The MapReduce framework forms these groups by first sorting the intermediate (key, value) pairs and then grouping values with the same key. It should be noted that the data is sorted to simplify the grouping process. The Reduce function processes each (key, [set of values]) group and produces a set of (key, value) pairs as output.

Intermediate (key, val) pairs

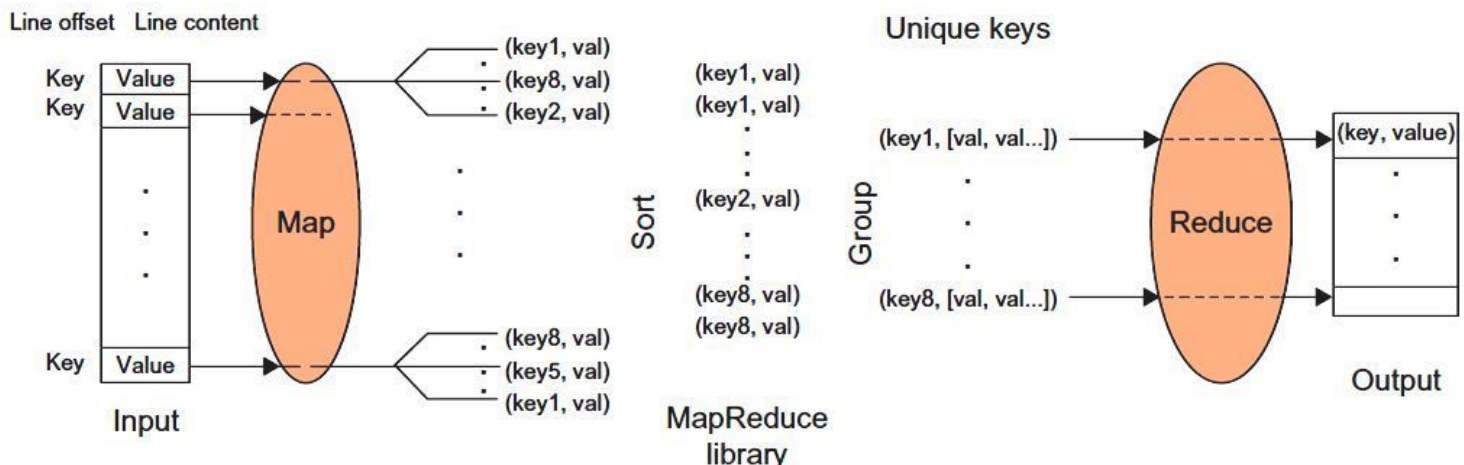


Figure: MapReduce logical data flow in 5 processing stages over successive (key, value) pairs.

Example: sample MapReduce application word count with two lines (1) “most people ignore most poetry” and (2) “most poetry ignores most people.” The Map function simultaneously produces a number of intermediate (key, value) pairs for each line of content so that each word is the intermediate key with 1 as its intermediate value; for example, (ignore, 1). Then the MapReduce library collects all the generated intermediate (key, value) pairs and sorts them to group the 1’s for identical words; for example, (people, [1,1]). Groups are then sent to the Reduce function in parallel so that it can sum up the 1 values for each word and generate the actual number of occurrence for each word in the file; for example, (people, 2).

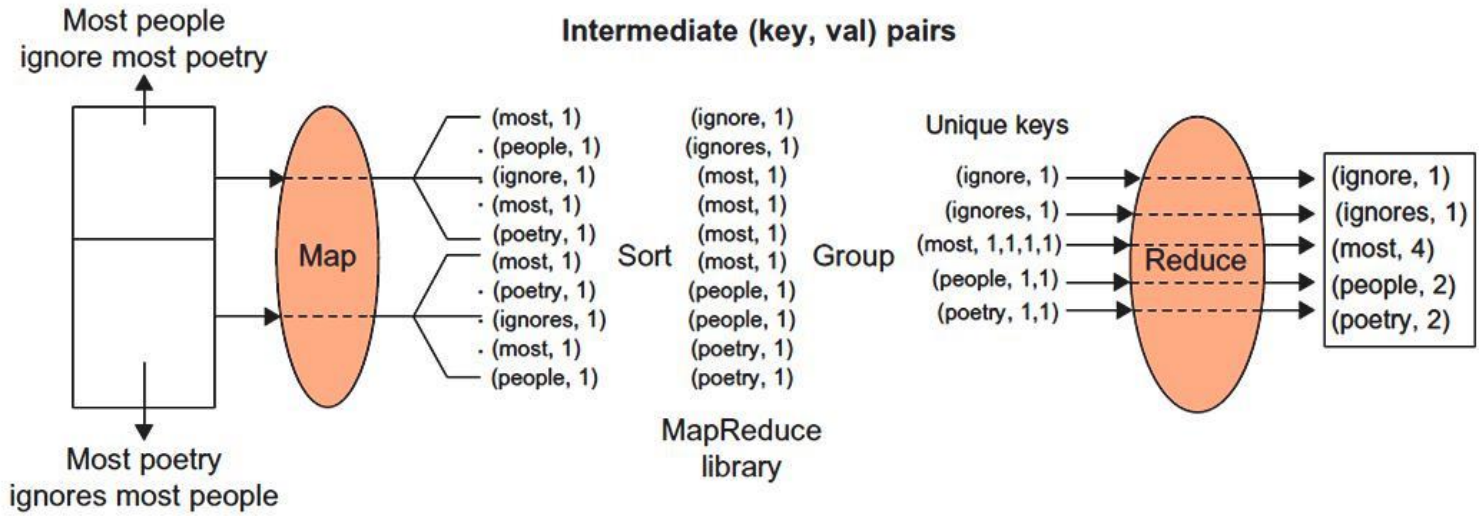


Figure: The data flow of a word-count problem using the MapReduce functions (Map, Sort, Group and Reduce) in a cascade operations.

Formal Notation of MapReduce Data Flow

The Map function is applied in parallel to every input (key, value) pair, and produces new set of intermediate (key, value) pairs

$$(key_1, val_1) \xrightarrow{\text{Map Function}} \text{List}(key_2, val_2)$$

Then the MapReduce library collects all the produced intermediate (key, value) pairs from all input (key, value) pairs, and sorts them based on the “key” part. Finally, the Reduce function is applied in parallel to each group producing the collection of values as output as illustrated here:

$$(key_2, \text{List}(val_2)) \xrightarrow{\text{Reduce Function}} \text{List}(val_2)$$

MapReduce Actual Data and Control Flow: The main responsibility of the MapReduce framework is to efficiently run a user’s program on a distributed computing system, which handles all partitioning, mapping, synchronization, communication, and scheduling details of such data flows

- 1) Data partitioning:** The MapReduce library splits the input data (files), already stored in GFS, into M pieces that also correspond to the number of map tasks.
- 2) Computation partitioning** This is implicitly handled (in the MapReduce framework) by obliging users to write their programs in the form of the Map and Reduce functions. The MapReduce library only

generates copies of a user program (e.g., by a fork system call) containing the Map and the Reduce functions, distributes them, and starts them up on a number of available computation engines.

- 3) Determining the master and workers:** The MapReduce architecture is based on a master worker model and one of the copies of the user program becomes the master and the rest become workers. The master picks idle workers, and assigns the map and reduce tasks to them. A map/reduce worker is typically a computation engine such as a cluster node to run map/ reduce tasks by executing Map/Reduce functions.
- 4) Reading the input data (data distribution):** Each map worker reads its corresponding portion of the input data, namely the input data split, and sends it to its Map function.
- 5) Map function:** Each Map function receives the input data split as a set of (key, value) pairs to process and produce the intermediated (key, value) pairs.
- 6) Combiner function:** This is an optional local function within the map worker which applies to intermediate (key, value) pairs. The user can invoke the Combiner function inside the user program. The Combiner function runs the same code written by users for the Reduce function as its functionality is identical to it. The Combiner function merges the local data of each map worker before sending it over the network to effectively reduce its communication costs.
- 7) Partitioning function:** The intermediate (key, value) pairs produced by each map worker are partitioned into R regions, equal to the number of reduce tasks, by the Partitioning function to guarantee that all (key, value) pairs with identical keys are stored in the same region. As a result, since reduce worker i reads the data of region i of all map workers, all (key, value) pairs with the same key will be gathered by reduce worker i accordingly. a Partitioning function could simply be a hash function (e.g., $\text{Hash}(\text{key}) \bmod R$) that forwards the data into particular regions.
- 8) Synchronization:** MapReduce applies a simple synchronization policy to coordinate map workers with reduce workers, in which the communication between them starts when all map tasks finish.
- 9) Communication:** Reduce worker i, already notified of the location of region i of all map workers, uses a remote procedure call to read the data from the respective region of all map workers. Since all reduce workers read the data from all map workers, all-to-all communication among all map and reduce workers, which incurs network congestion, occurs in the network.
- 10) Sorting and Grouping:** When the process of reading the input data is finalized by a reduce worker, the data is initially buffered in the local disk of the reduce worker. Then the reduce worker groups intermediate (key, value) pairs by sorting the data based on their keys, followed by grouping all occurrences of identical keys.
- 11) Reduce function:** The reduce worker iterates over the grouped (key, value) pairs, and for each unique key, it sends the key and corresponding values to the Reduce function. Then this function processes its input data and stores the output results in predetermined files in the user's program.

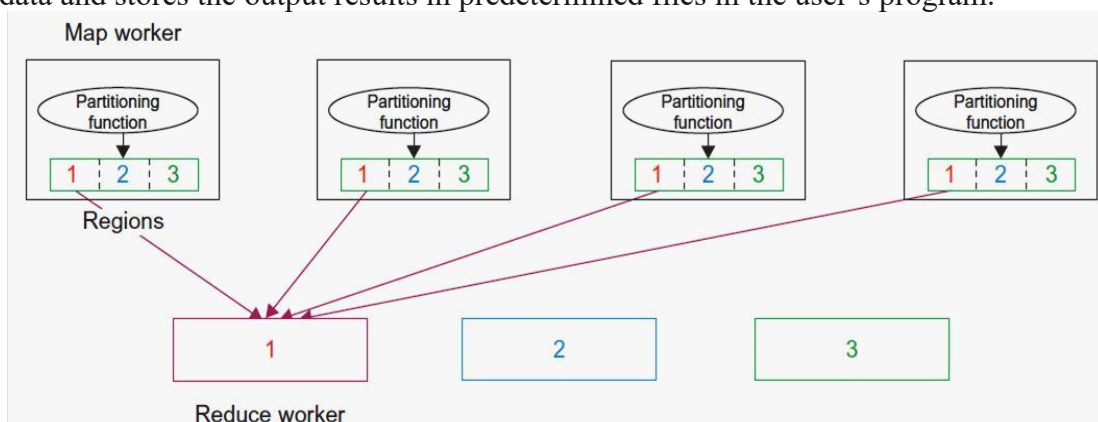


Figure: Use of MapReduce partitioning function to link the Map and Reduce workers

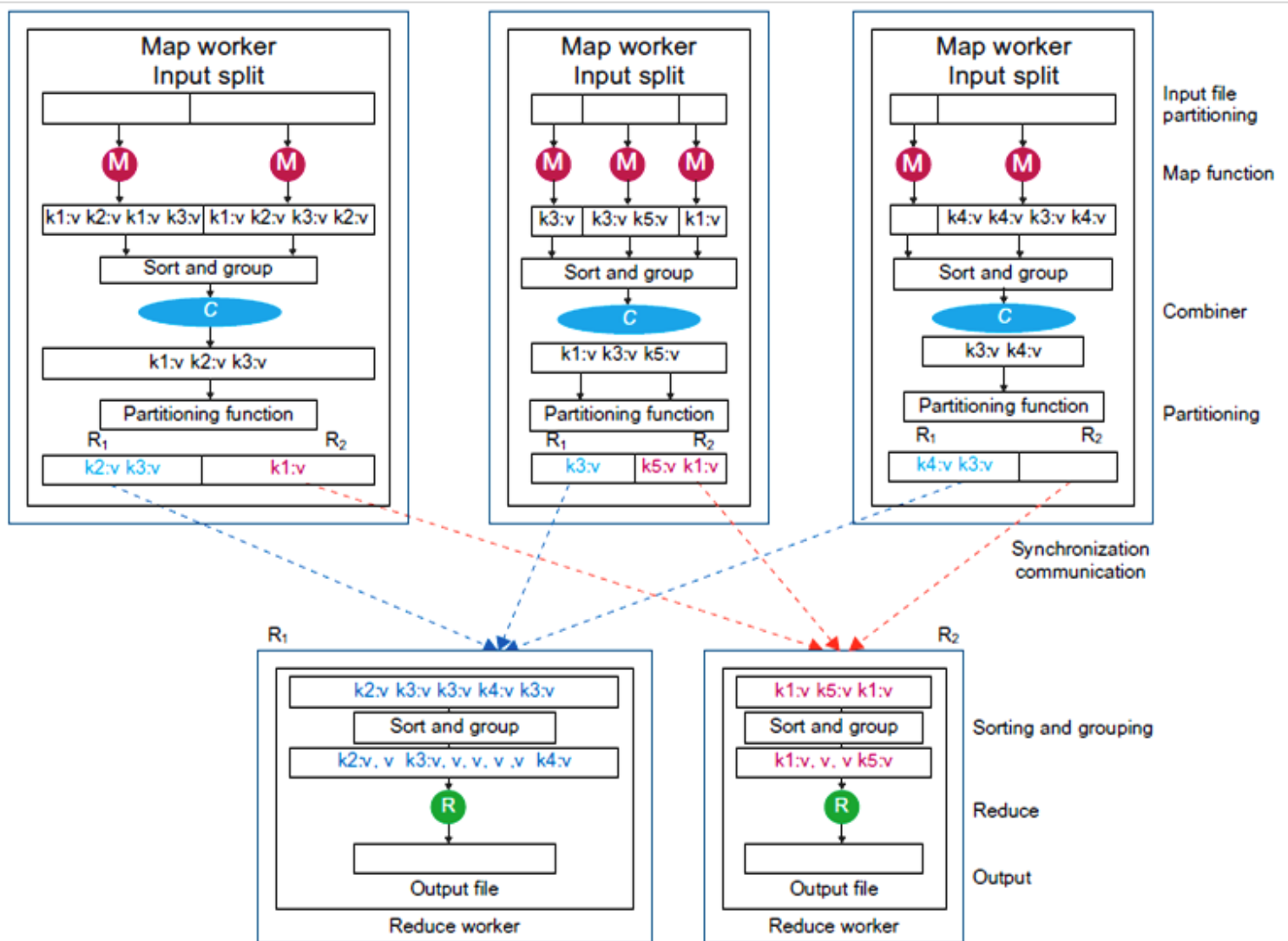
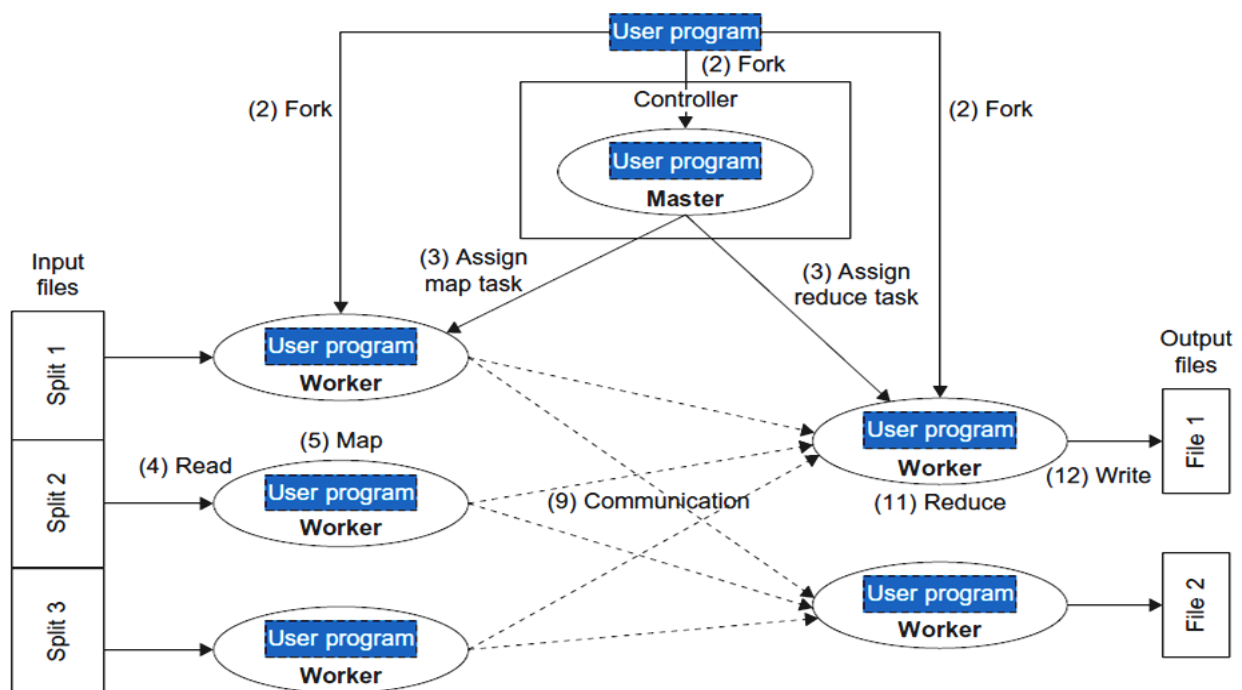


Figure: Data flow implementation of many functions in the Map workers and in the Reduce workers through multiple sequences of partitioning, combining, synchronization and communication, sorting and grouping, and reduce operations.



Twister and Iterative MapReduce

It is important to understand the performance of different runtimes and, in particular, to compare Message Passing Interface (MPI) and MapReduce. The two major sources of parallel overhead are load imbalance and communication. The communication overhead in MapReduce can be quite high, for two reasons:

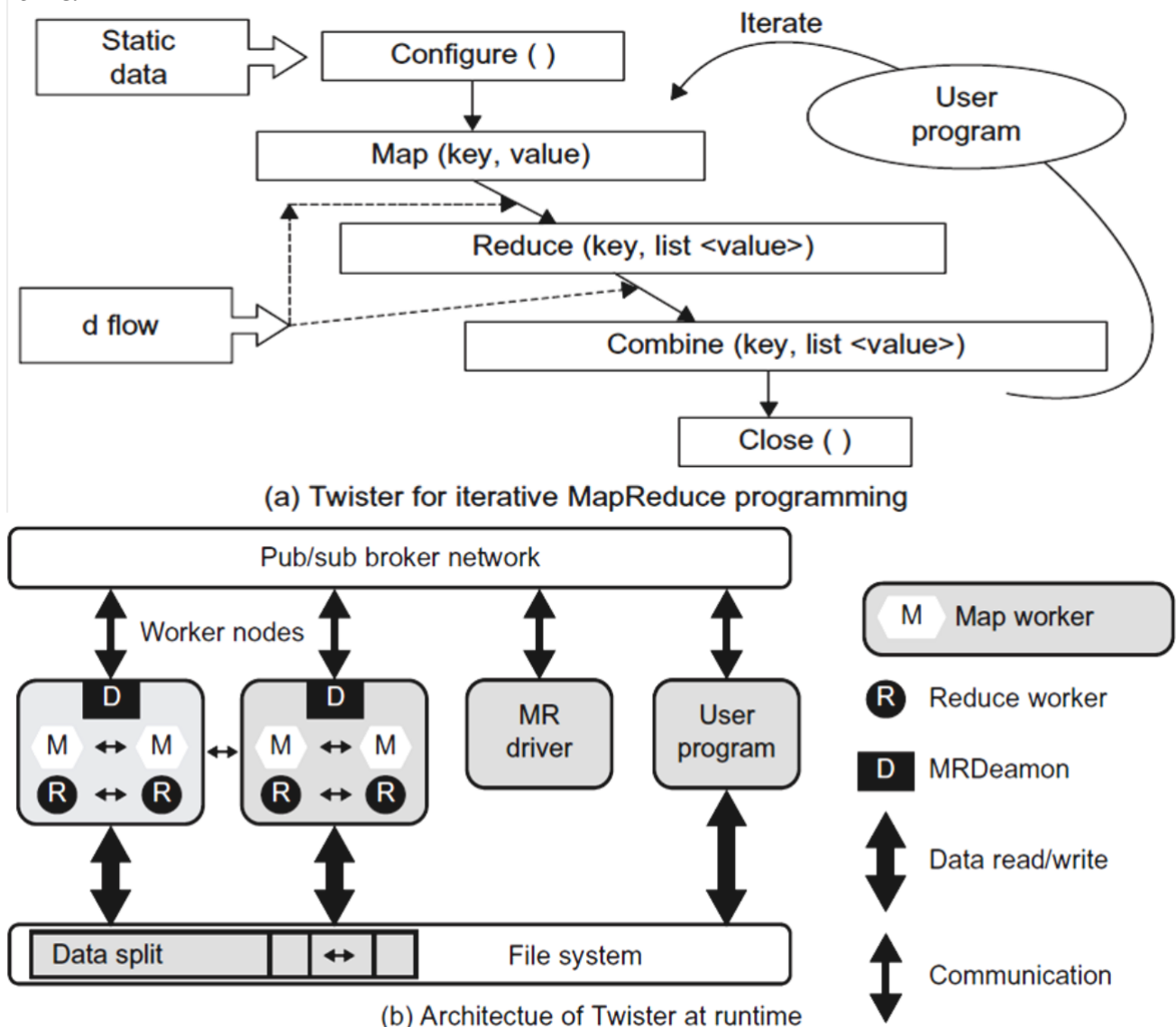
- MapReduce reads and writes via files, whereas MPI transfers information directly between nodes over the network.
- MPI does not transfer all data from node to node, but just the amount needed to update information. We can call the MPI flow δ flow and the MapReduce flow full data flow.

The same phenomenon is seen in all “classic parallel” loosely synchronous applications which typically exhibit an iteration structure over compute phases followed by communication phases. We can address the performance issues with two important changes:

- Stream information between steps without writing intermediate steps to disk.
- Use long-running threads or processors to communicate the δ (between iterations) flow.

The Twister programming paradigm and its implementation architecture at run time are illustrated in below Figure. Twister distinguishes the static data which is never reloaded from the dynamic δ flow that is communicated.

Figure: Twister - An iterative MapReduce programming paradigm for repeated MapReduce executions. (a) Twister for iterative MapReduce programming and (b) Architectue of Twister at runtime.



The Map-Reduce pair is iteratively executed in long-running threads. We compare in Figure the different thread and process structures of 4 parallel programming paradigms: namely Hadoop, Dryad, Twister (also called MapReduce++), and MPI.

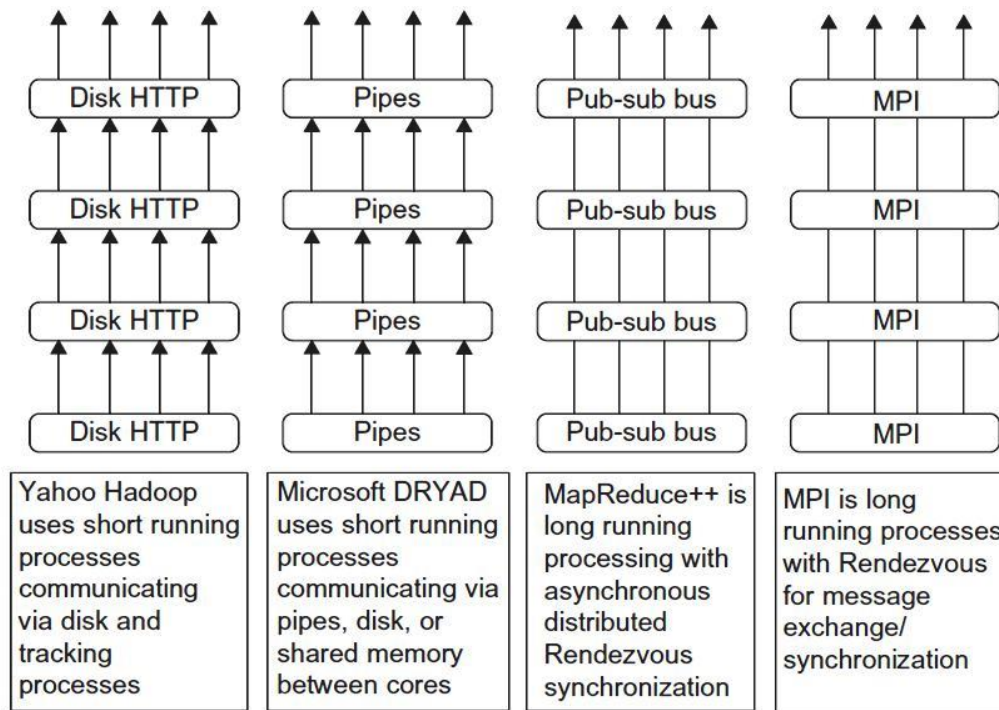


Figure: compare the different thread and process structures of 4 parallel programming paradigms: namely Hadoop, Dryad, Twister (also called MapReduce++), and MPI.

► Hadoop Library from Apache

- Hadoop is an open source implementation of MapReduce coded and released in Java.
 - The Hadoop implementation of MapReduce uses the Hadoop Distributed File System (HDFS) as its underlying layer.
 - The Hadoop core is divided into two fundamental layers: the MapReduce engine and HDFS. The MapReduce engine is the computation engine running on top of HDFS as its data storage manager. The following two sections cover the details of these two fundamental layers.
 - HDFS: HDFS is a distributed file system inspired by GFS that organizes files and stores their data on a distributed computing system.
- **HDFS Architecture:** HDFS has a master/slave architecture containing a single NameNode as the master and a number of DataNodes as workers (slaves). To store a file in this architecture, HDFS splits the file into fixed-size blocks (e.g., 64 MB) and stores them on workers (DataNodes). The mapping of blocks to DataNodes is determined by the NameNode. The NameNode (master) also manages the file system's metadata and namespace. In such systems, the namespace is the area maintaining the metadata, and metadata refers to all the information stored by a file system that is needed for overall management of all files.

Each DataNode, usually one per node in a cluster, manages the storage attached to the node. Each Data Node is responsible for storing and retrieving its file blocks.

► HDFS Features:

Distributed file systems have special requirements, such as performance, scalability, concurrency control, fault tolerance, and security requirements, to operate efficiently. Two important characteristics of HDFS are,



HDFS Fault Tolerance

- **Block Replication:** HDFS stores a file as a set of blocks and each block is replicated and distributed across the whole cluster. The replication factor is set by the user and is three by default.
- **Replica placement:** HDFS stores one replica in the same node the original data is stored, one replica on a different node but in the same rack, and one replica on a different node in a different rack to provide three copies of the data.
- **Heartbeat and Blockreport messages:** Heartbeats and Blockreports are periodic messages sent to the NameNode by each DataNode in a cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly, while each Blockreport contains a list of all blocks on a DataNode.



HDFS High-Throughput Access to Large Data Sets (Files), this is for providing batch processing rather than interactive processing, data access throughput in HDFS is more important than latency. Applications run on HDFS typically have large data sets, individual files are broken into large blocks (e.g., 64MB) to allow HDFS to decrease the amount of metadata storage required per file.

- **HDFS Operation:** The control flow of HDFS operations such as write and read can properly highlight roles of the NameNode and DataNodes in the managing operations. The control flow of the main operations of HDFS on files is further described to manifest the interaction between the user, the NameNode, and the DataNodes in such systems.

- 1) **Reading a file:** To read a file in HDFS, a user sends an “open” request to the NameNode to get the location of file blocks. For each file block, the NameNode returns the address of a set of DataNodes containing replica information for the requested file. The user calls the read function to connect to the closest DataNode containing the first block of the file. After the first block is streamed from the respective DataNode to the user, the established connection is terminated and the same process is repeated for all blocks of the requested file until the whole file is streamed to the user.
- 2) **Writing to a file:** To write a file in HDFS, a user sends a “create” request to the NameNode to create a new file in the file system namespace. If the file does not exist, the NameNode notifies the user and allows him to start writing data to the file by calling the write function. The first block of the file is written to an internal queue termed the data queue while a data streamer monitors its writing into a DataNode.

► Architecture of MapReduce in Hadoop

The topmost layer of Hadoop is the MapReduce engine that manages the data flow and control flow of MapReduce jobs over distributed computing systems. The MapReduce engine also has a master/slave architecture consisting of a single JobTracker as the master and a number of TaskTrackers as the slaves (workers). The JobTracker manages the MapReduce job over a cluster and is responsible for monitoring jobs and assigning tasks to TaskTrackers.

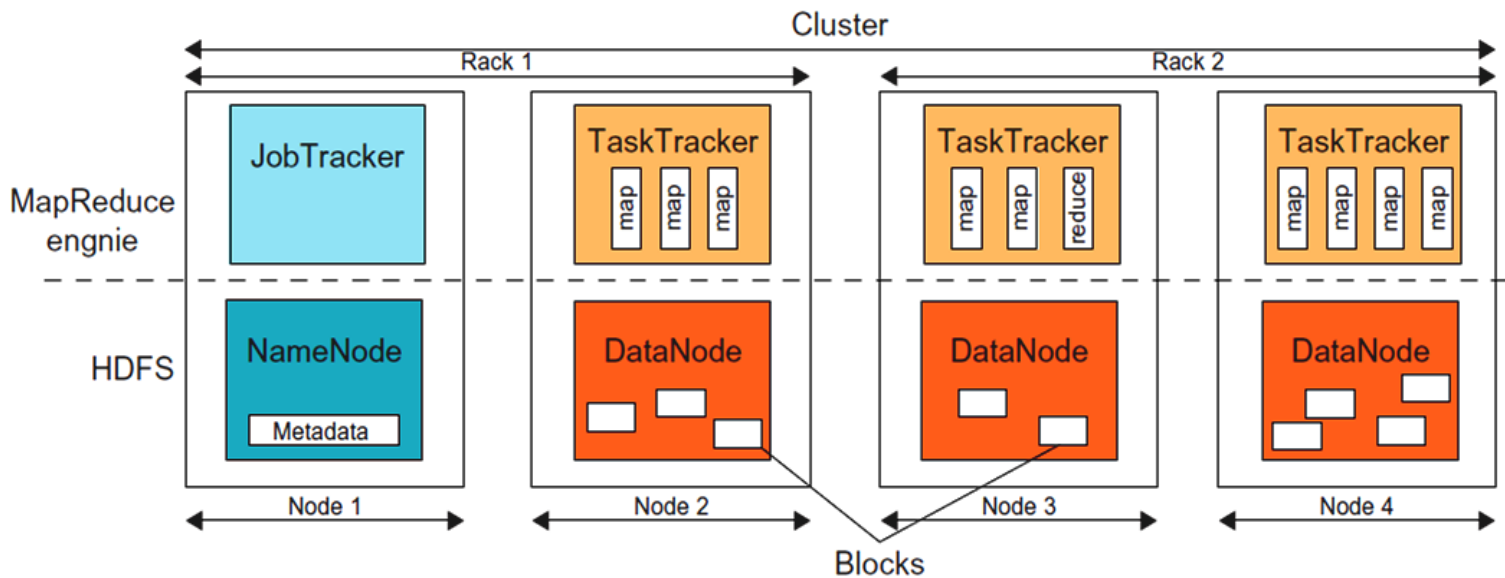


Figure: HDFS and MapReduce architecture in Hadoop where boxes with different shadings refer to different functional nodes applied to different blocks of data.

▶ Running a Job in Hadoop

Three components contribute in running a job in this system: a user node, a JobTracker, and several TaskTrackers. The data flow starts by calling the `runJob(conf)` function inside a user program running on the user node, in which `conf` is an object containing some tuning parameters for the MapReduce framework and HDFS. The `runJob(conf)` function and `conf` are comparable to the `MapReduce(Spec, &Results)` function and `Spec` in the first implementation of MapReduce by Google.

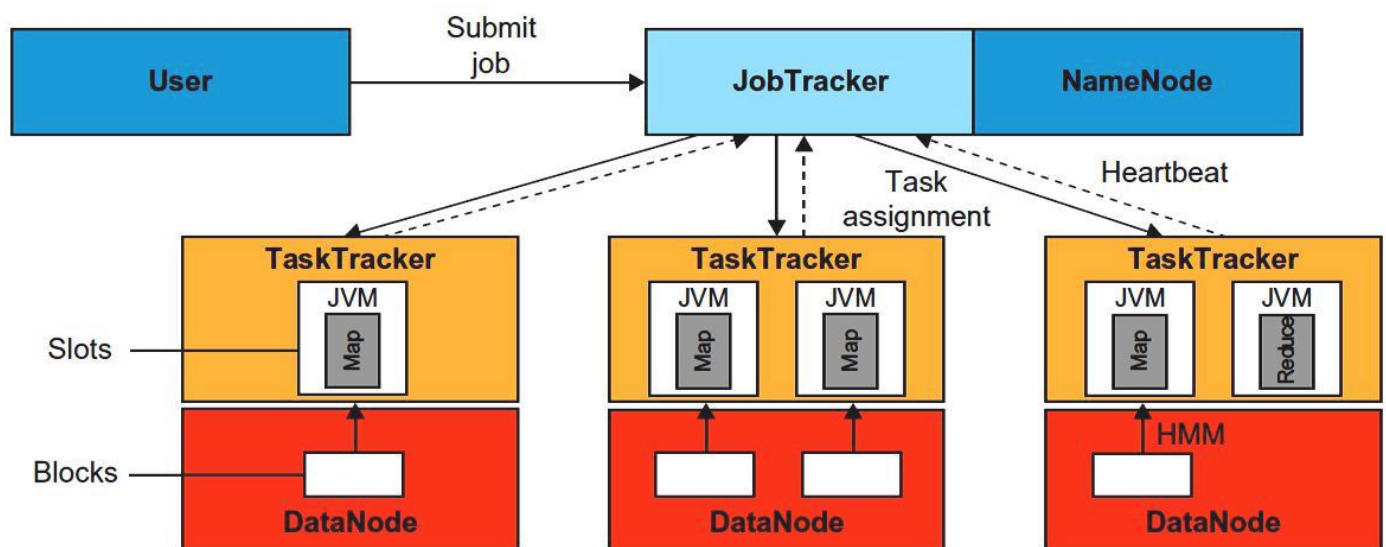


Figure: Data flow in running a MapReduce job at various task trackers using the Hadoop library.

Job Submission Each job is submitted from a user node to the JobTracker node that might be situated in a different node within the cluster through the following procedure:

- A user node asks for a new job ID from the JobTracker and computes input file splits.
- The user node copies some resources, such as the job's JAR file, configuration file, and computed input splits, to the JobTracker's file system.
- The user node submits the job to the JobTracker by calling the `submitJob()` function.

Task Assignment The JobTracker creates one map task for each computed input split by the user node and assigns the map tasks to the execution slots of the TaskTrackers. The JobTracker considers the localization of the data when assigning the map tasks to the TaskTrackers. The JobTracker also creates reduce tasks and assigns them to the TaskTrackers. The number of reduce tasks is predetermined by the user, and there is no locality consideration in assigning them.

Task Execution The control flow to execute a task (either map or reduce) starts inside the TaskTracker by copying the job JAR file to its file system. Instructions inside the job JAR file are executed after launching a Java Virtual Machine (JVM) to run its map or reduce task.

Task running check A task running check is performed by receiving periodic heartbeat messages to the JobTracker from the TaskTrackers. Each heartbeat notifies the JobTracker that the sending TaskTracker is alive, and whether the sending TaskTracker is ready to run a new task.

⇒ Mapping Applications to Parallel and Distributed Systems:

Mapping applications to different hardware and software in terms of five application architectures, these initial five categories are listed in Table.

Category	Class	Description	Machine Architecture
1	Synchronous	The problem class can be implemented with instruction-level lockstep operation as in SIMD architectures.	SIMD
2	Loosely synchronous (BSP or bulk synchronous processing)	These problems exhibit iterative compute-communication stages with independent compute (map) operations for each CPU that are synchronized with a communication step. This problem class covers many successful MPI applications including partial differential equation solutions and particle dynamics applications.	MIMD on MPP (massively parallel processor)
3	Asynchronous	Illustrated by Compute Chess and Integer Programming; combinatorial search is often supported by dynamic threads. This is rarely important in scientific computing, but it is at the heart of operating systems and concurrency in consumer applications such as Microsoft Word.	Shared memory
4	Pleasingly parallel	Each component is independent. In 1988, Fox estimated this at 20 percent of the total number of applications, but that percentage has grown with the use of grids and data analysis applications including, for example, the Large Hadron Collider analysis for particle physics.	Grids moving to clouds
5	Metaproblems	These are coarse-grained (asynchronous or data flow) combinations of categories 1-4 and 6. This area has also grown in importance and is well supported by grids and described by workflow in Section 3.5 .	Grids of clusters
6	MapReduce++ (Twister)	This describes file (database) to file (database) operations which have three subcategories (see also Table 6.11): 6a) Pleasingly Parallel Map Only (similar to category 4) 6b) Map followed by reductions 6c) Iterative "Map followed by reductions" (extension of current technologies that supports linear algebra and data mining)	Data-intensive clouds a) Master-worker or MapReduce b) MapReduce c) Twister

Category 1 was popular 20 years ago, but is no longer significant. It describes applications that can be parallelized with lock-step operations controlled by hardware, run on SIMD (single-instruction and multiple-data) machines.

Category 2 is now much more important and corresponds to a SPMD (single-program and multiple data) model running on MIMD (multiple instruction multiple data) machines. Here, each decomposed and includes dynamic irregular cases with complex geometries for solving partial differential equations or particle dynamics. Also note that category 2 consists of compute–communicate phases and the computations are synchronized by communication. No additional synchronization is needed.

Category 3 consists of asynchronously interacting objects and is often considered the people’s view of a typical parallel problem. It probably does describe the concurrent threads in a modern operating system, as well as some important applications, such as event-driven simulations and areas such as search in computer games and graph algorithms. Shared memory is natural due to the low latency often needed to perform dynamic synchronization. It wasn’t clear in the past, but now it appears that this category is not very common in large-scale parallel problems of importance.

Category 4 is the simplest algorithmically, with disconnected parallel components. Both grids and clouds are very natural for this class, which does not need high-performance communication between different nodes.

Category 5 refers to the coarse-grained linkage of different “atomic” problems, Grids or clouds are suitable for metaproblems as coarse grained decomposition does not usually require stringent performance.

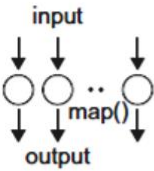
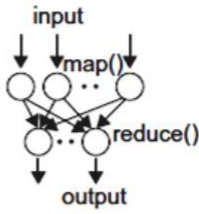
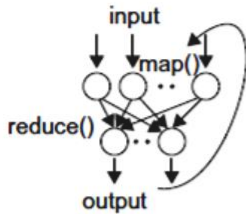
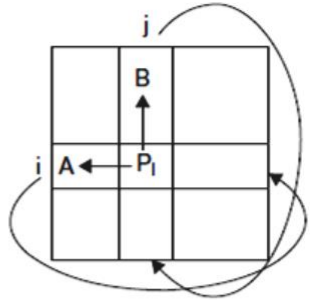
we added a sixth category to cover data-intensive applications motivated by the clear importance of MapReduce as a new programming model. We call this category MapReduce++, and it has three subcategories: “map only” applications similar to pleasingly parallel category 4.

⇒ Programming Support of Google App Engine

➤ Programming the Google App Engine

- Below Figure summarizes some key features of GAE programming model for two supported languages: Java and Python.
- There are several powerful constructs for storing and accessing data. The data store is a NOSQL data management system for entities that can be, at most, 1 MB in size and are labeled by a set of schema-less properties. Queries can retrieve entities of a given kind filtered and sorted by the values of the properties. Java offers Java Data Object (JDO) and Java Persistence API (JPA) interfaces implemented by the open source Data Nucleus Access platform, while Python has a SQL-like query language called GQL. The data store is strongly consistent and uses optimistic concurrency control.
- The data store implements transactions across its distributed network using “entity groups.” A transaction manipulates entities within a single group. Entities of the same group are stored together for efficient execution of transactions. Your GAE application can assign entities to groups when the entities are created. The performance of the data store can be enhanced by in-memory caching using the memcache, which can also be used independently of the data store.

Table 6.11 Comparison of MapReduce++ Subcategories along with the Loosely Synchronous Category Used in MPI

Map-Only	Classic MapReduce	Iterative MapReduce	Loosely Synchronous
			
• Document conversion (e.g., PDF->HTML) • Brute force searches in cryptography • Parametric sweeps • Gene assembly • PolarGrid Matlab data analysis (www.polargrid.org)	• High-energy physics (HEP) histograms • Distributed search • Distributed sort • Information retrieval • Calculation of pairwise distances for sequences (BLAST)	• Expectation maximization algorithms • Linear algebra • Data mining including • Clustering • K-means • Deterministic annealing clustering • Multidimensional scaling (MDS)	• Many MPI scientific applications utilizing a wide variety of communication constructs including local interactions • Solving differential equations and particle dynamics with short-range forces
← Domain of MapReduce and Iterative Extensions →			MPI

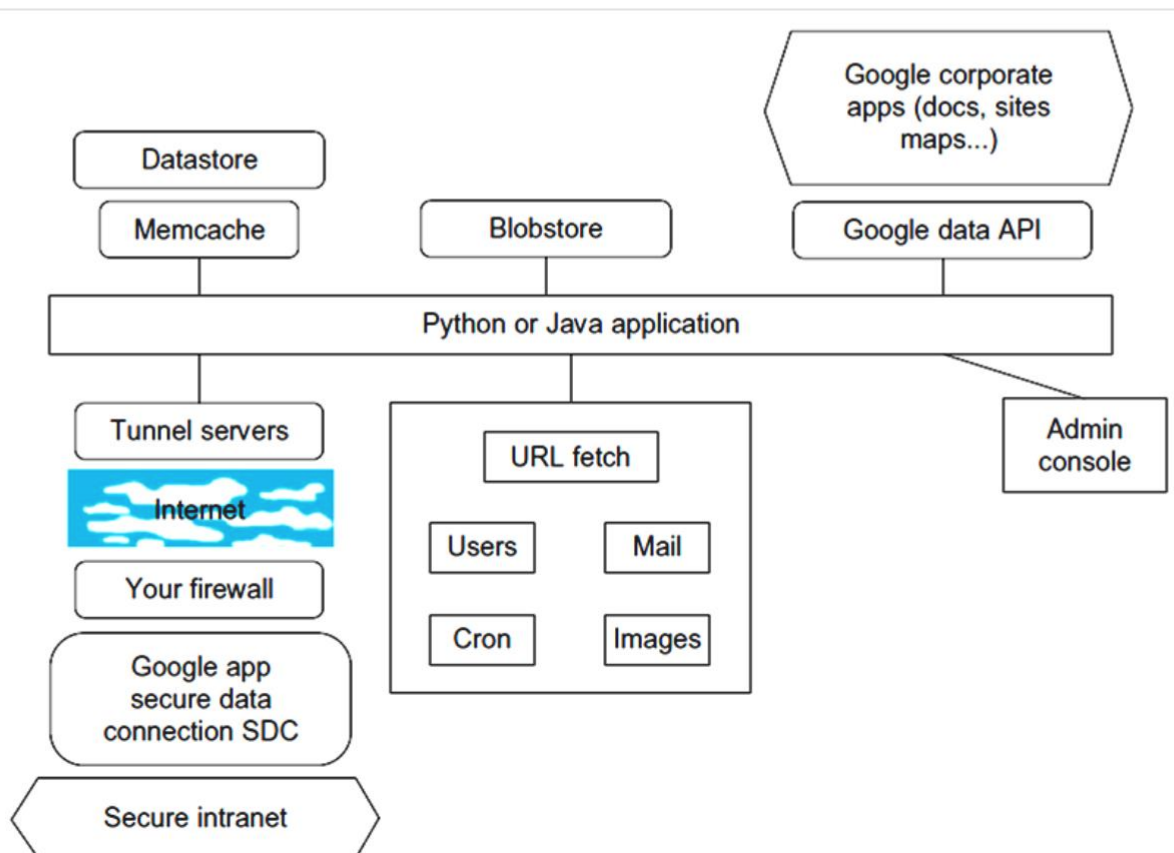


Figure: Programming environment for Google App Engine.

- Google added the blobstore which is suitable for large files as its size limit is 2 GB. There are several mechanisms for incorporating external resources. The Google SDC Secure Data Connection can tunnel through the Internet and link your intranet to an external GAE application. The URL Fetch operation provides the ability for applications to fetch resources and communicate with other hosts over the Internet using HTTP and HTTPS requests. There is a specialized mail mechanism to send e-mail from your GAE application.
- Applications can access resources on the Internet, such as web services or other data, using GAE's URL fetch service. The URL fetch service retrieves web resources using the same highspeed Google infrastructure that retrieves web pages for many other Google products. There are dozens of Google "corporate" facilities including maps, sites, groups, calendar, docs, and YouTube, among others. These support the Google Data API which can be used inside GAE.
- Applications can access resources on the Internet, such as web services or other data, using GAE's URL fetch service. The URL fetch service retrieves web resources using the same highspeed Google infrastructure that retrieves web pages for many other Google products. There are dozens of Google "corporate" facilities including maps, sites, groups, calendar, docs, and YouTube, among others. These support the Google Data API which can be used inside GAE.

➤ Google File System (GFS)

- GFS typically will hold a large number of huge files, each 100MB or larger, with files that are multiple GB in size quite common. Thus, Google has chosen its file data block size to be 64MB instead of the 4 KB in typical traditional file systems.
- Other nodes act as the chunk servers for storing data, while the single master stores the metadata. The file system namespace and locking facilities are managed by the master. The master periodically communicates with the chunk servers to collect management information as well as give instructions to the chunk servers to do work such as load balancing or fail recovery.

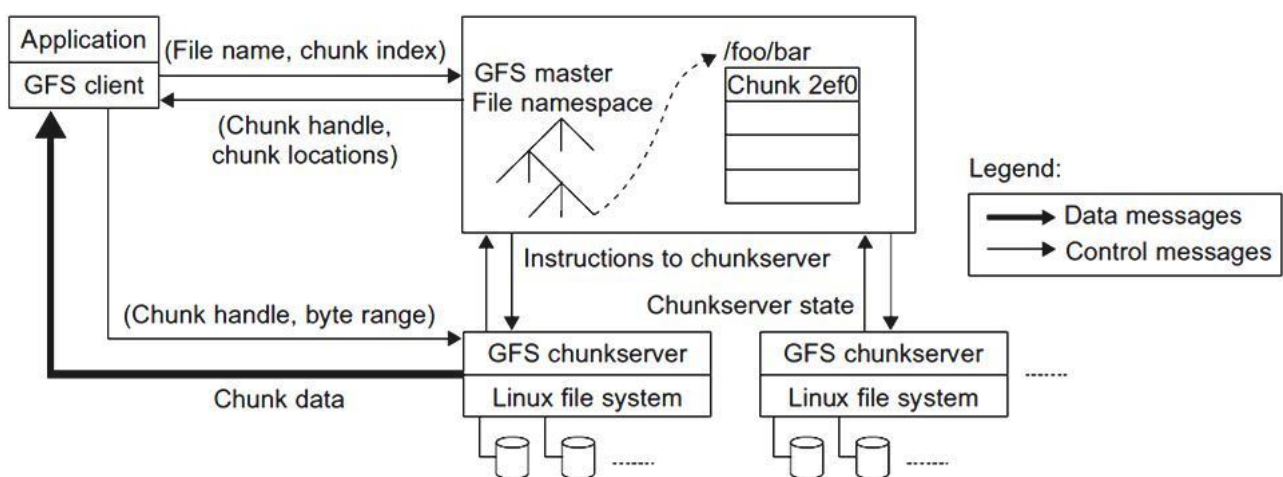
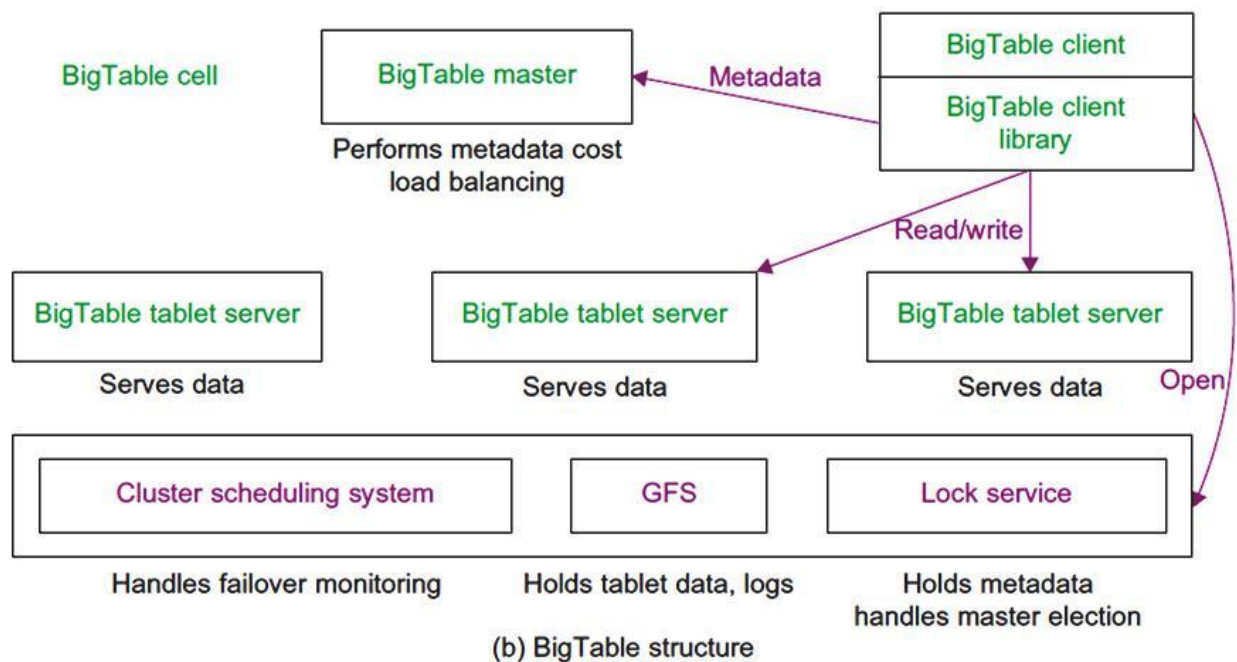


Figure: Architecture of Google File System (GFS).

➤ BigTable, Google's NOSQL System

- Big Table was designed to provide a service for storing and retrieving structured and semi structured data.

- Big Table applications include storage of web pages, per-user data, and geographic locations. Here we use web pages to represent URLs and their associated data, such as contents, crawled metadata, links, anchors, and page rank values.
- BigTable can be viewed as a distributed multilevel map. It provides a fault-tolerant and persistent database as in a storage service. The BigTable system is scalable, which means the system has thousands of servers, terabytes of in-memory data, petabytes of disk-based data, millions of reads/writes per second, and efficient scans.
- BigTable is used in many projects, including Google Search, Orkut, and Google Maps/Google Earth, among others. One of the largest BigTable cell manages ~200 TB of data spread over several thousand machines.



Large tables are broken into tablets at row boundaries. A tablet holds a contiguous range of rows. Clients can often choose row keys to achieve locality. The system aims for about 100MB to 200MB of data per tablet.

➤ Chubby, Google's Distributed Lock Service

Chubby is intended to provide a coarse-grained locking service. It can store small files inside Chubby storage which provides a simple namespace as a file system tree. The files stored in Chubby are quite small compared to the huge files in GFS. Based on the Paxos agreement protocol, the Chubby system can be quite reliable despite the failure of any member node.

⇒ Programming on Amazon AWS :

we will consider the programming support in the AWS platform. The Elastic MapReduce capability is equivalent to Hadoop running on the basic EC2 offering. Amazon has NOSQL support. Amazon offers the Simple Queue Service (SQS) and Simple Notification Service (SNS).

➤ Programming on Amazon EC2

Amazon was the first company to introduce VMs in application hosting. Customers can rent VMs instead of physical machines to run their own applications. By using VMs, customers can load any software of

their choice. The elastic feature of such a service is that a customer can create, launch, and terminate server instances as needed, paying by the hour for active servers. Amazon provides several types of preinstalled VMs. Instances are often called Amazon Machine Images (AMIs) which are preconfigured with operating systems based on Linux or Windows, and additional software.

AMIs are the templates for instances, which are running VMs. The workflow to create a VM is Create an AMI→Create Key Pair→Configure Firewall→Launch.

Three Types of AMI

Private AMI Images created by you, which are private by default. You can grant access to other users to launch your private images.

Public AMI Images created by users and released to the AWS community, so anyone can launch instances based on them and use them any way they like.

Paid QAMI You can create images providing specific functions that can be launched by anyone willing to pay you per each hour of usage on top of Amazon's charges.

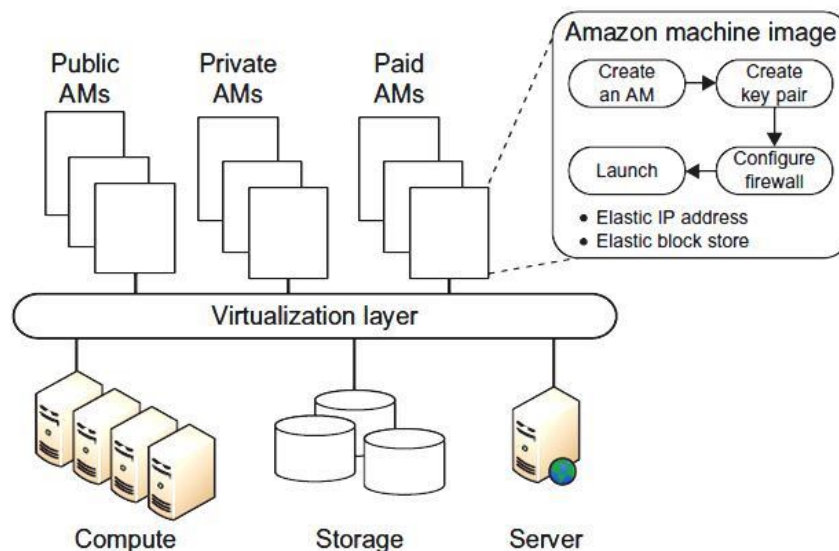


Figure: Amazon EC2 execution environment.

Both auto-scaling and elastic load balancing are enabled by CloudWatch which monitors running instances. CloudWatch is a web service that provides monitoring for AWS cloud resources, starting with Amazon EC2. It provides customers with visibility into resource utilization, operational performance, and overall demand patterns—including metrics such as CPU utilization, disk reads and writes, and network traffic.

➤ Amazon Simple Storage Service (S3)

Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. S3 provides the object-oriented storage service for users. Users can access their objects through Simple Object Access Protocol (SOAP) with either browsers or other client programs which support SOAP. SQS is responsible for ensuring a reliable message service between two processes, even if the receiver processes are not running.

The fundamental operation unit of S3 is called an object. Each object is stored in a bucket and retrieved via a unique, developer-assigned key. In other words, the bucket is the container of the object. Besides unique key attributes, the object has other attributes such as values, metadata, and access control information.

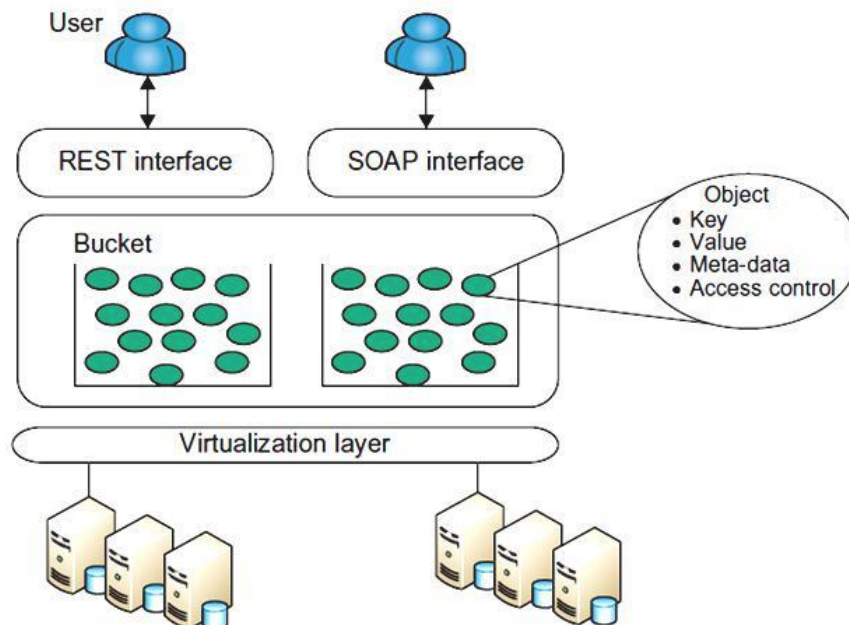


Figure: Amazon S3 execution environment

Here are some key features of S3:

- Redundant through geographic dispersion
- Designed to provide 99.999999999 percent durability and 99.99 percent availability of objects over a given year with cheaper reduced redundancy storage (RRS).
- Authentication mechanisms to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.
- Per-object URLs and ACLs (access control lists).
- Default download protocol of HTTP. A BitTorrent protocol interface is provided to lower costs for high-scale distribution.
- First 1 GB per month input or output free and then \$.08 to \$.15 per GB for transfers outside an S3 region.
- There is no data transfer charge for data transferred between Amazon EC2 and Amazon S3 within the same region.

➤ **Amazon Elastic Block Store (EBS) and SimpleDB**

The Elastic Block Store (EBS) provides the volume block interface for saving and restoring the virtual images of EC2 instances. Traditional EC2 instances will be destroyed after use. The status of EC2 can now be saved in the EBS system after the machine is shut down. Users can use EBS to save persistent data and mount to the running instances of EC2. Note that S3 is “Storage as a Service” with a messaging interface. EBS allows you to create storage volumes from 1 GB to 1 TB that can be mounted as EC2 instances.

➤ **Amazon SimpleDB Service**

SimpleDB provides a simplified data model based on the relational database data model. Structured data from users must be organized into domains. Each domain can be considered a table. The items are the rows in the table.

⇒ Emerging Cloud Software Environments

We will assess popular cloud operating systems and emerging software environments. We cover the open source Eucalyptus, OpenNebula, Open Stack , and ClodSim.

➤ Open Source Eucalyptus

- Eucalyptus was initially aimed at bringing the cloud computing paradigm to academic supercomputers and clusters. Eucalyptus provides an AWS-compliant EC2-based web service interface for interacting with the cloud service. Additionally, Eucalyptus provides services, such as the AWS-compliant Walrus, and a user interface for managing users and images.
- Eucalyptus Architecture: The Eucalyptus system is an open software environment. Figure shows the architecture based on the need to manage VM images. The system supports cloud programmers in VM image management as follows. Essentially, the system has been extended to support the development of both the computer cloud and storage cloud.
- VM Image Management: Eucalyptus stores images in Walrus, the block storage system that is analogous to the Amazon S3 service. As such, any user can bundle her own root file system, and upload and then register this image and link it with a particular kernel and ramdisk image.

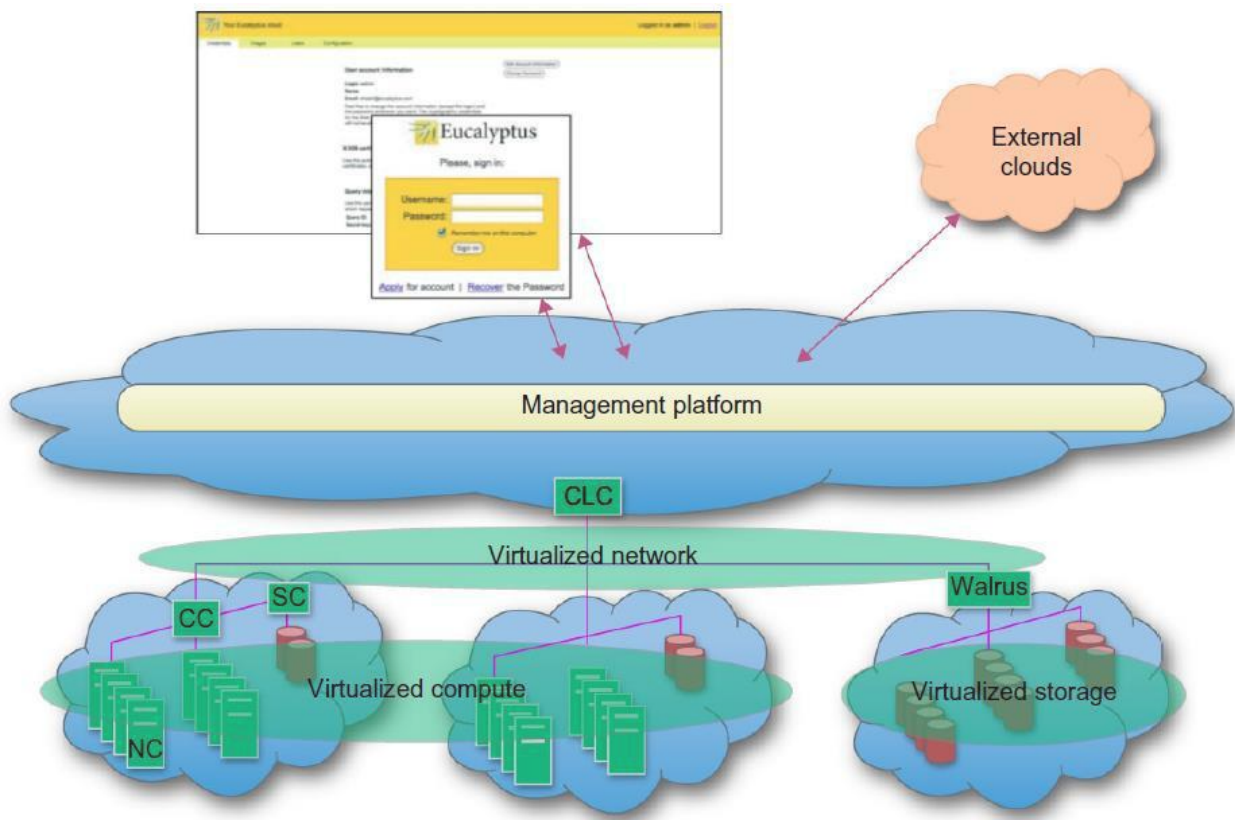
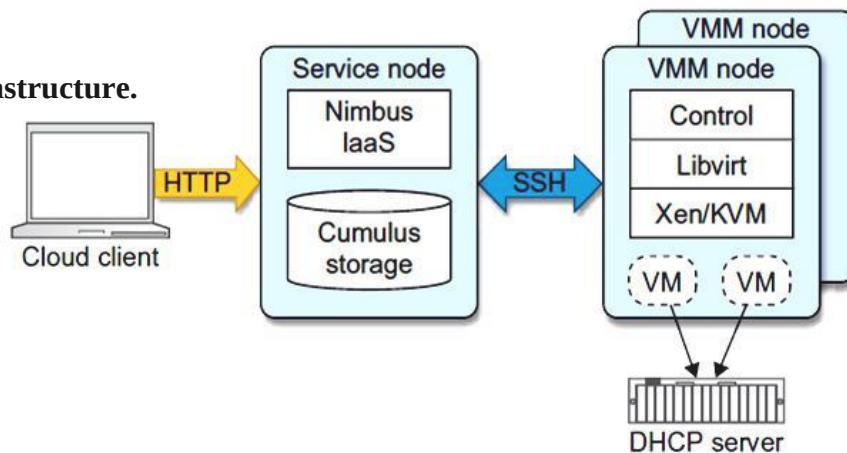


Figure: The Eucalyptus architecture for VM image management.

- **Nimbus:** Nimbus is a set of open source tools that together provide an IaaS cloud computing solution. Nimbus is a set of open source tools that together provide an IaaS cloud computing solution. A storage cloud implementation called Cumulus has been tightly integrated with the other central services, although it can also be used stand-alone.

Figure: Nimbus cloud infrastructure.



➤ OpenNebula

- OpenNebula is an open source toolkit which allows users to transform existing infrastructure into an IaaS cloud with cloud-like interfaces. Figure shows the OpenNebula architecture and its main components.

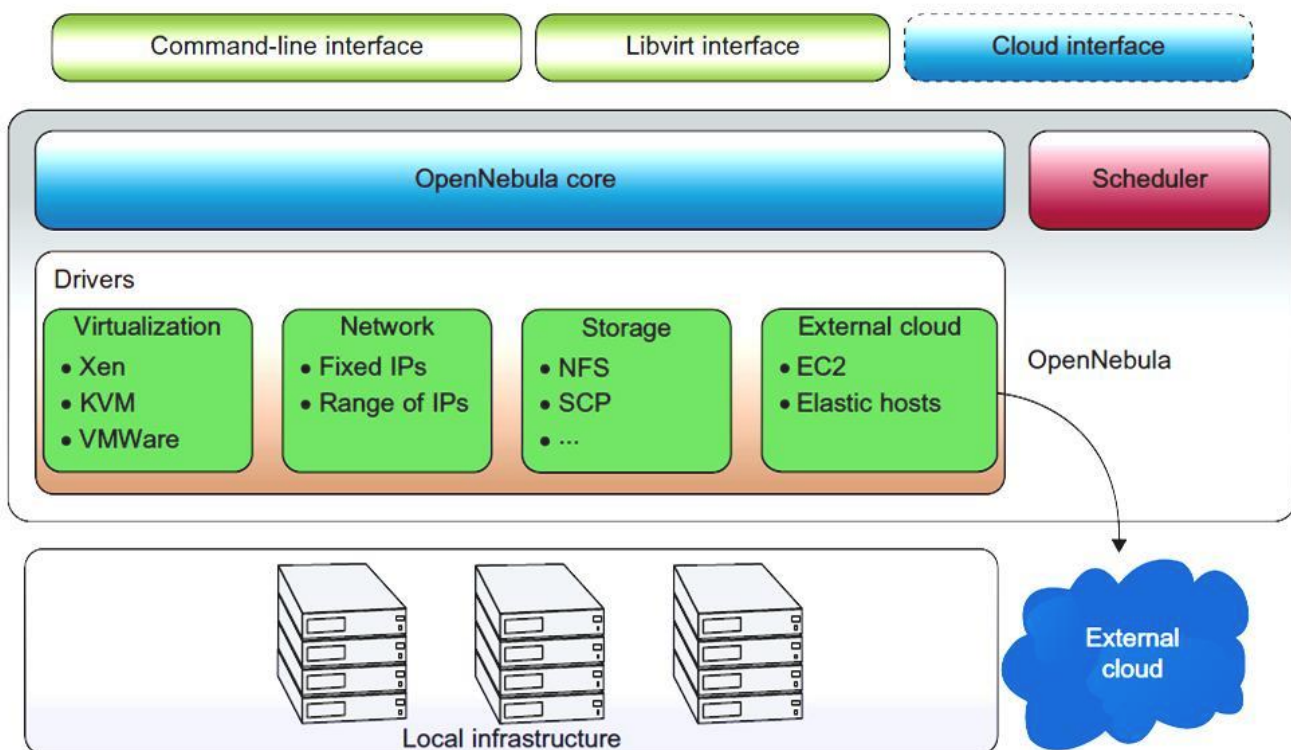


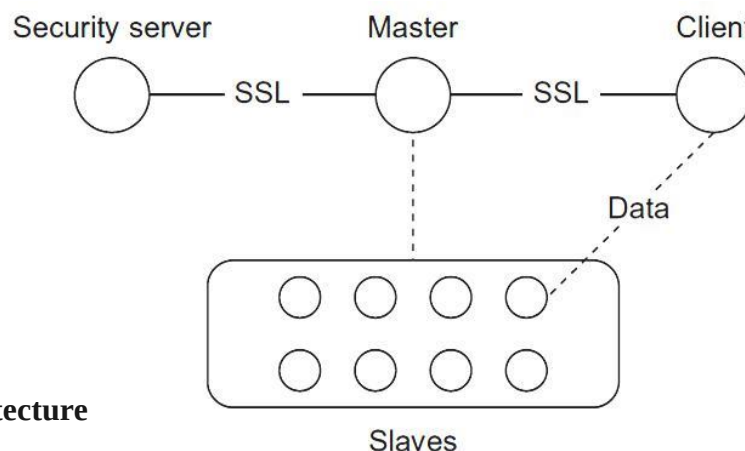
Figure: OpenNebula architecture and its main components

- Here, the core is a centralized component that manages the VM full life cycle, including setting up networks dynamically for groups of VMs and managing their storage requirements, such as VM disk image deployment or on-the-fly software environment creation.
- The default capacity scheduler is a requirement/rank matchmaker. However, it is also possible to develop more complex scheduling policies, through a lease model and advance reservations.

- The access drivers , provide an abstraction of the underlying infrastructure to expose the basic functionality of the monitoring, storage, and virtualization services available in the cluster.
- OpenNebula can support a hybrid cloudmodel by using cloud drivers to interface with external clouds. Regarding storage, an Image Repository allows users to easily specify disk images from a catalog without worrying about low-level disk configuration attributes or block device mapping.

➤ Sector/Sphere

- Sector/Sphere is a software platform that supports very large distributed data storage and simplified distributed data processing over large clusters of commodity computers, either within a data center or across multiple data centers. The system consists of the Sector distributed file system and the Sphere parallel data processing framework. Sector is a distributed file system (DFS) that can be deployed over a wide area and allows users to manage large data sets from any location with a high speed network connection
- Since Sector is aware of the network topology when it places replicas, it also provides better reliability, availability, and access throughout. The communication is performed using User Datagram Protocol (UDP) for message passing and user-defined type (UDT) for data transfer.



The Sector/Sphere system architecture

- Sphere is a parallel data processing engine designed to work with data managed by Sector. This coupling allows the system to make accurate decisions about job scheduling and data location. Sphere provides a programming framework that developers can use to process data stored in Sector.
- The first component is the security server, which is responsible for authenticating master servers, slave nodes, and users. The last component is the client component.

➤ OpenStack

- OpenStack focuses on the development of two aspects of cloud computing to address compute and storage aspects with the OpenStack Compute and OpenStack Storage solutions. “OpenStack Compute is the internal fabric of the cloud creating and managing large groups of virtual private servers” and “OpenStack Object Storage is software for creating redundant, scalable object storage using clusters of commodity servers to store terabytes or even petabytes of data.”

OpenStack Compute

- As part of its computing support efforts, OpenStack is developing a cloud computing fabric controller, a component of an IaaS system, known as Nova. The architecture for Nova is built on the concepts of shared-nothing and messaging-based information exchange.
- In this architecture, the API Server receives HTTP requests from boto, converts the commands to and from the API format, and forwards the requests to the cloud controller. The cloud controller maintains the global state of the system, ensures authorization while interacting with the User Manager via Lightweight Directory Access Protocol (LDAP), interacts with the S3 service, and manages nodes, as well as storage workers through a queue. Nova integrates networking components to manage private networks, public IP addressing, virtual private network (VPN) connectivity, and firewall rules. It includes the following types:
 - NetworkController manages address and virtual LAN (VLAN) allocations
 - RoutingNode governs the NAT (network address translation) conversion of public IPs to private IPs, and enforces firewall rules
 - AddressingNode runs Dynamic Host Configuration Protocol (DHCP) services for private Networks
 - TunnelingNode provides VPN connectivity The network state (managed in the distributed object store) consists of the following:
 - VLAN assignment to a project
 - Private subnet assignment to a security group in a VLAN
 - Private IP assignments to running instances
 - Public IP allocations to a project
 - Public IP associations to a private IP/running instance

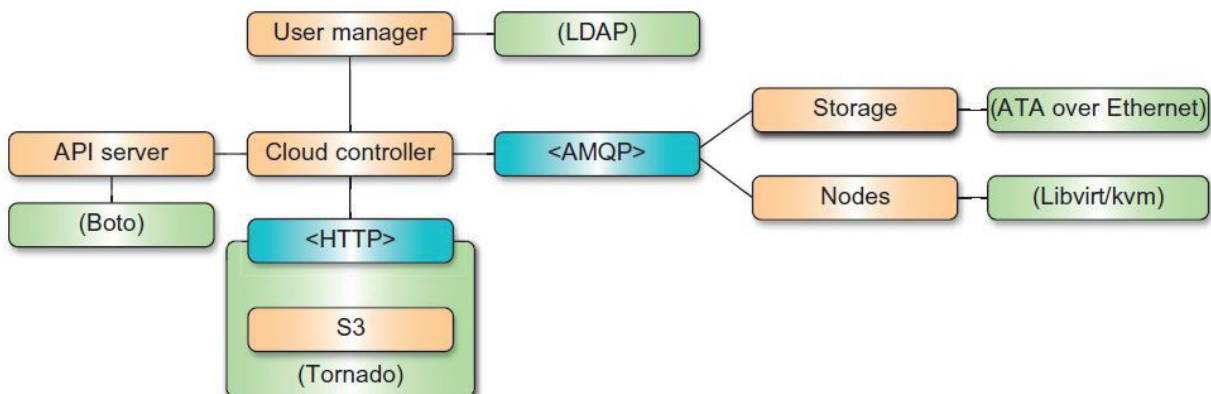


Figure: OpenStack Nova system architecture. The AMQP (Advanced Messaging Queuing Protocol)

Open Stack Storage

The OpenStack storage solution is built around a number of interacting components and concepts, including a proxy server, a ring, an object server, a container server, an account server, replication, updaters, and auditors. The role of the proxy server is to enable lookups to the accounts, containers, or objects in OpenStack storage rings and route the requests. A ring represents a mapping between the names of entities stored on disk and their physical locations. Objects are stored as binary files with metadata stored in the file's extended attributes.

➤ Manjrasoft Aneka Cloud

- Aneka is a cloud application platform developed by Manjrasoft, based in Melbourne, Australia. It is designed to support rapid development and deployment of parallel and distributed applications on private or public clouds. It provides a rich set of APIs for transparently exploiting distributed resources and expressing the business logic of applications by using preferred programming abstractions. System administrators can leverage a collection of tools to monitor and control the deployed infrastructure.
- Aneka acts as a workload distribution and management platform for accelerating applications in both Linux and Microsoft .NET framework environments. Some of the key advantages of Aneka over other workload distribution solutions include:
 - 1) Support of multiple programming and application environments
 - 2) Simultaneous support of multiple runtime environments
 - 3) Rapid deployment tools and framework
 - 4) Ability to harness multiple virtual and/or physical machines for accelerating application provisioning based on users' Quality of Service/service-level agreement (QoS/SLA) requirements
 - 5) Built on top of the Microsoft .NET framework, with support for Linux environments
- Aneka offers three types of capabilities which are essential for building, accelerating, and managing clouds and their applications:
 - 1) **Build** Aneka includes a new SDK which combines APIs and tools to enable users to rapidly develop applications. Aneka also allows users to build different runtime environments.
 - 2) **Accelerate** Aneka supports rapid development and deployment of applications in multiple runtime environments running different operating systems such as Windows or Linux/UNIX. Aneka uses physical machines as much as possible to achieve maximum utilization in local environments.
 - 3) **Manage** Management tools and capabilities supported by Aneka include a GUI and APIs to set up, monitor, manage, and maintain remote and global Aneka compute clouds. Aneka also has an accounting mechanism and manages priorities and scalability based on SLA/QoS which enables dynamic provisioning.
- Here are three important programming models supported by Aneka for both cloud and traditional parallel applications:
 - 1) Thread programming model, best solution to adopt for leveraging the computing capabilities of multicore nodes in a cloud of computers
 - 2) Task programming model, which allows for quickly prototyping and implementing an independent bag of task applications
 - 3) MapReduce programming model, as discussed

Aneka Architecture

Aneka as a cloud application platform features a homogeneous distributed runtime environment for applications. This environment is built by aggregating together physical and virtual nodes hosting the Aneka container. The container is a lightweight layer that interfaces with the hosting environment and manages the services deployed on a node. The interaction with the hosting platform is mediated through the Platform Abstraction Layer (PAL), which hides in its implementation all the heterogeneity of the different operating systems.

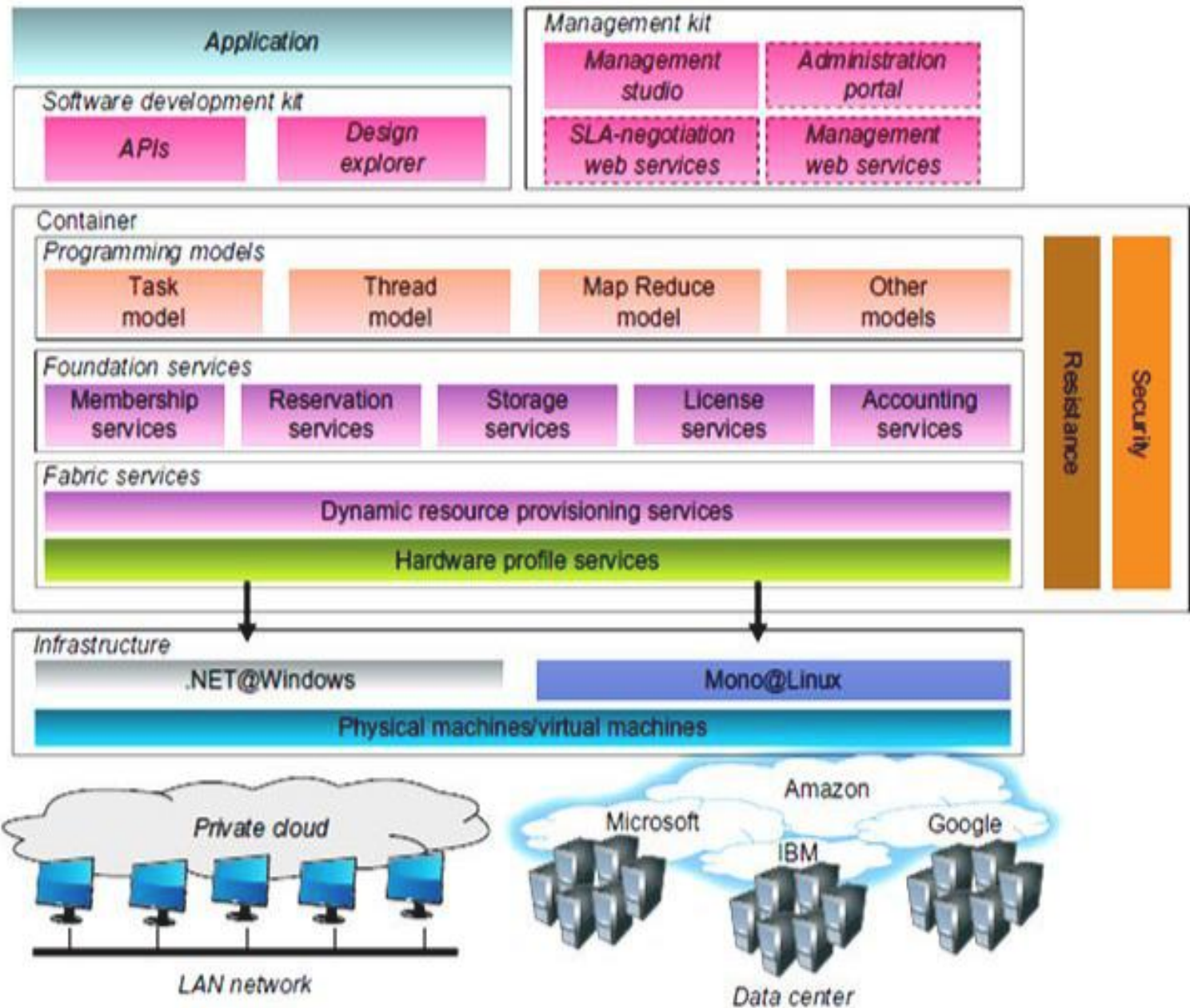


Figure: Architecture and components of Aneka.

The available services can be aggregated into three major categories:

- ❖ **Fabric Services:** Fabric services implement the fundamental operations of the infrastructure of the cloud. These services include HA and failover for improved reliability, node membership and directory, resource provisioning, performance monitoring, and hardware profiling.
- ❖ **Foundation Services:** Foundation services constitute the core functionalities of the Aneka middleware. They provide a basic set of capabilities that enhance application execution in the cloud. These services provide added value to the infrastructure and are of use to system administrators and developers. Within this category we can list storage management, resource reservation, reporting, accounting, billing, services monitoring, and licensing. Services in this level operate across the range of supported application models.
- ❖ **Application Services:** Application services deal directly with the execution of applications and are in charge of providing the appropriate runtime environment for each application model. They leverage

foundation and fabric services for several tasks of application execution such as elastic scalability, data transfer, and performance monitoring, accounting, and billing. At this level, Aneka expresses its true potential in supporting different application models and distributed programming patterns.

CloudSim

- **CloudSim** is a library for the simulation of cloud scenarios. It provides essential classes for describing data centres, computational resources, virtual machines, applications, users, and policies for the management of various parts of the system such as scheduling and provisioning.
- CloudSim provides a generalised and extensible simulation framework that enables seamless modelling and simulation of app performance. By using CloudSim, developers can focus on specific systems design issues that they want to investigate, without getting concerned about details related to cloud-based infrastructures and services.

An introduction to CloudSim

- ✓ CloudSim is a simulation tool that allows cloud developers to test the performance of their provisioning policies in a repeatable and controllable environment, free of cost.
- ✓ Using these components, it is easy to evaluate new strategies governing the use of clouds, while considering policies, scheduling algorithms, load balancing policies, etc.
- ✓ It can also be used to assess the competence of strategies from various perspectives such as cost, application execution time, etc. It also supports the evaluation of Green IT policies. It can be used as a building block for a simulated cloud environment and can add new policies for scheduling, load balancing and new scenarios.
- ✓ It is flexible enough to be used as a library that allows you to add a desired scenario by writing Java program.
- ✓ By using CloudSim, organisations, R&D centres and industry-based developers can test the performance of a newly developed application in a controlled and easy to set-up environment. The prominent features offered by CloudSim are given in Figure 1.



Architecture of CloudSim

The CloudSim layer provides support for modelling and simulation of cloud environments including dedicated management interfaces for memory, storage, bandwidth and VMs. It also provisions hosts to VMs, application execution management and dynamic system state monitoring. A cloud service provider can implement customised strategies at this layer to study the efficiency of different policies in VM provisioning.

The user code layer exposes basic entities such as the number of machines, their specifications, etc, as well as applications, VMs, number of users, application types and scheduling policies.

The main components of the CloudSim framework,

Regions: It models geographical regions in which cloud service providers allocate resources to their customers. In cloud analysis, there are six regions that correspond to six continents in the world.

Data centres: It models the infrastructure services provided by various cloud service providers. It encapsulates a set of computing hosts or servers that are either heterogeneous or homogeneous in nature, based on their hardware configurations.

Data centre characteristics: It models information regarding data centre resource configurations.

Hosts: It models physical resources (compute or storage).

The user base: It models a group of users considered as a single unit in the simulation, and its main responsibility is to generate traffic for the simulation.

Cloudlet: It specifies the set of user requests. It contains the application ID, name of the user base that is the originator to which the responses have to be routed back, as well as the size of the request execution commands, and input and output files. It models the cloud-based application services. CloudSim categorises the complexity of an application in terms of its computational requirements. Each application service has a pre-assigned instruction length and data transfer overhead that it needs to carry out during its life cycle.

Service broker: The service broker decides which data centre should be selected to provide the services to the requests from the user base.

VMM allocation policy: It models provisioning policies on how to allocate VMs to hosts.

VM scheduler: It models the time or space shared, scheduling a policy to allocate processor cores to VMs.

UNIT- VSECURITY IN THE CLOUDTopics Covered:

- ② Security Overview
- ② Cloud Security Challenges and Risks
- ② Software-as-a-Service Security
- ② Security Governance
- ② Risk Management
- ② Security Monitoring
- ② Security Architecture Design
- ② Data Security
- ② Application Security
- ② Virtual Machine Security
- ② Identity Management and Access Control
- ② Autonomic Security

⇒ Cloud Security Overview

Security is a principal concern when **entrusting** an organization's critical information to geographically dispersed cloud platforms not under the direct control of that organization.

- **Information Life Cycle Management** — “Understand cloud provider policies and processes for *data retention* and *destruction* and how they compare with internal organizational policy. Be aware that data retention assurance may be easier for the cloud provider to demonstrate, but data destruction may be very difficult. Perform regular backup and recovery tests to assure that logical segregation and controls are effective.”
- **Application Security** — “IaaS, PaaS and SaaS create differing trust boundaries for the software development lifecycle, which must be accounted for during the development, testing and production deployment of applications.”
- **Storage** — “Understand cloud provider storage retirement processes. Data destruction is extremely difficult in a multi-tenant environment and the cloud provider should be utilizing strong storage encryption that renders data unreadable when storage is recycled, disposed of, or accessed by any means outside of authorized applications.”



With cloud computing providing **SaaS**, secure software is a critical issue. From the cloud consumer's point of view, using SaaS in the cloud reduces the need for secure software development by the customer. The requirement for secure software development is transferred to the cloud provider.

➤ Cloud Information Security Objectives

The Data and Analysis Center for Software (DACS) requires that software must exhibit the following three properties to be considered secure:

- ❖ **Dependability** — Software that executes predictably and operates correctly under a variety of conditions, including when under attack or running on a malicious host.
- ❖ **Trustworthiness** — Software that contains a minimum number of vulnerabilities or no vulnerabilities or weaknesses that could sabotage the software's dependability. It must also be resistant to malicious logic.
- ❖ **Survivability (Resilience)** — Software that is resistant to or tolerant of attacks and has the ability to recover as quickly as possible with as little harm as possible.

Seven complementary principles that support information assurance are **confidentiality, integrity, availability, authentication, authorization, auditing, and accountability**.

Confidentiality, Integrity, and Availability

Confidentiality, integrity, and availability are sometimes known as the *CIA triad* of information system security, and are important pillars of cloud software assurance.

❖ *Confidentiality*

Confidentiality refers to the prevention of intentional or unintentional unauthorized **disclosure of information**. Confidentiality in cloud systems is related to the areas of intellectual property rights, covert channels, traffic analysis, encryption, and inference:

- **Intellectual property rights** — Intellectual property (IP) includes inventions, designs, and artistic, musical, and literary works. Rights to intellectual property are covered by copyright laws, which protect creations of the mind, and patents, which are granted for new inventions.
- **Covert channels** — A *covert channel* is an unauthorized and unintended communication path that enables the exchange of information. Covert channels can be accomplished through timing of messages or inappropriate use of storage mechanisms.
- **Traffic analysis** — *Traffic analysis* is a form of confidentiality breach that can be accomplished by analyzing the volume, rate, source, and destination of message traffic, even if it is encrypted. Increased message activity and high bursts of traffic can indicate a major event is occurring. Countermeasures to traffic analysis include maintaining a near-constant rate of message traffic and disguising the source and destination locations of the traffic.
- **Encryption** — *Encryption* involves scrambling messages so that they cannot be read by an unauthorized entity, even if they are intercepted. The amount of effort (*work factor*) required to decrypt the message is a function of the strength of the encryption key and the robustness and quality of the encryption algorithm.
- **Inference** — *Inference* is usually associated with database security. Inference is the ability of an entity to use and correlate information protected at one level of security to uncover information that is protected at a higher security level.

❖ *Integrity*

The concept of cloud information *integrity* requires that the following three principles are met:

- Modifications are not made to data by unauthorized personnel or processes.
- **Unauthorized modifications** are not made to **data** by authorized personnel or processes.
- The data is internally and externally consistent — in other words, the internal information is consistent both among all sub-entities and with the real-world, external situation.

❖ *Availability*

Availability ensures the **reliable and timely access to cloud data** or cloud computing resources by the appropriate personnel. Availability guarantees that the systems are functioning properly when needed. In

addition, this concept guarantees that the security services of the cloud system are in working order. A denial-of-service attack is an example of a threat against availability.

The reverse of confidentiality, integrity, and availability is disclosure, alteration, and destruction (DAD).

Cloud Security Services

Additional factors that directly affect cloud software assurance include authentication, authorization, auditing, and accountability.

❖ **Authentication**

Authentication is the testing or reconciliation of evidence of a **user's identity**. It establishes the user's identity and ensures that users are who they claim to be. For example, a user presents an identity (user ID) to a computer login screen and then has to provide a password. The computer system authenticates the user by verifying that the password corresponds to the individual presenting the ID.

❖ **Authorization**

Authorization refers to **rights** and **privileges** granted to an individual or process that enable access to computer resources and information assets. Once a user's identity and authentication are established, authorization levels determine the extent of system rights a user can hold.

❖ **Auditing**

To maintain operational assurance, organizations use two basic methods: system audits and monitoring. These methods can be employed by the cloud customer, the cloud provider, or both, depending on asset architecture and deployment.

- A **system audit** is a one-time or periodic event to evaluate security.
- **Monitoring** refers to an ongoing activity that examines either the system or the users, such as intrusion detection.

IT auditors typically audit the following functions:

- System and transaction controls
- Systems development standards
- Backup controls
- Data library procedures
- Data center security
- Contingency plans

Audit logs should record the following:

- The transaction's date and time
- Who processed the transaction
- At which terminal the transaction was processed
- Various security events relating to the transaction

In addition, an auditor should examine the audit logs for the following:

- Amendments to production jobs
- Production job reruns
- Computer operator practices
- All commands directly initiated by the user
- All identification and authentication attempts
- Files and resources accessed

❖ **Accountability**

Accountability is the ability to determine the actions and behaviors of a single individual within a cloud system and to identify that particular individual. Accountability is related to the concept of *non repudiation*, where an individual cannot successfully deny the performance of an action.

➤ Relevant Cloud Security Design Principles

11 security design principles:

Least privilege

The principle of *least privilege* maintains that an individual, process, or other type of entity should be given the minimum privileges and resources for the minimum period of time required to complete a task.

Separation of duties

Separation of duties requires that completion of a specified sensitive activity or access to sensitive objects is dependent on the satisfaction of a plurality of conditions.

Defense in depth

Defense in depth is the application of multiple layers of protection wherein a subsequent layer will provide protection if a previous layer is breached.

Fail safe

Fail safe means that if a cloud system fails it should fail to a state in which the security of the system and its data are not compromised.

Economy of mechanism

Economy of mechanism promotes simple and comprehensible design and implementation of protection mechanisms, so that unintended access paths do not exist or can be readily identified and eliminated.

Complete mediation

In complete meditation, every request by a subject to access an object in a computer system must undergo a valid and effective authorization procedure.

Open design

There has always been an ongoing discussion about the merits and strengths of security designs that are kept secret versus designs that are open to scrutiny and evaluation by the community at large.

Least common mechanism

This principle states that a minimum number of protection mechanisms should be common to multiple users, as shared access paths can be sources of unauthorized information exchange.

Psychological acceptability

Psychological acceptability refers to the ease of use and intuitiveness of the user interface that controls and interacts with the cloud access control mechanisms.

Weakest link

As in the old saying “A chain is only as strong as its weakest link,” the security of a cloud system is only as good as its weakest component.

Leveraging existing components

To increase cloud system security by leveraging existing components is to partition the system into defended subunits.

Then, if a security mechanism is penetrated for one sub-unit, it will not affect the other sub-units, and damage to the computing resources will be minimized.

➤ Secure Cloud Software Requirements

- ➔ *Software requirements engineering* is the process of determining customer software expectations and needs, and it is conducted before the software design phase.
- ➔ Department of Defense Data and Analysis Center for Software (DACS) state that all software shares the following three security needs:
 - ✓ It must be **dependable** under anticipated operating conditions, and remain dependable under hostile operating conditions.
 - ✓ It must be **trustworthy** in its own behavior, and in its inability to be compromised by an attacker through exploitation of vulnerabilities or insertion of malicious code.
 - ✓ It must be **resilient** enough to recover quickly to full operational capability with a minimum of damage to itself, the resources and data it handles, and the external components with which it interacts.

➤ NIST Security Principles National Institute of Standards and Technology's

Principle 1 — Establish a sound security policy as the “foundation” for design.

Principle 2 — Treat security as an integral part of the overall system design.

Principle 3 — clearly delineates the physical and logical security boundaries governed by associated security policies.

Principle 6 — Assume that external systems are insecure.

Principle 7 — Identify potential trade-offs between reducing risk and increased costs and decreases in other aspects of operational effectiveness.

Principle 8 — Implement layered security; ensure there is no single point of vulnerability.

Principle 9 — Isolate public access systems from mission-critical resources (e.g., data, processes, etc.).

Principle 10 — Use boundary mechanisms to separate computing systems and network infrastructures.

Principle 11 — Minimize the system elements to be trusted.

Principle 12 — Implement least privilege.

Principle 13 — Authenticate users and processes to ensure appropriate access control decisions both within and across domains.

Principle 14 — Use unique identities to ensure accountability.

⇒ Cloud Security Challenges and Risks

Cloud computing presents an organization with its own set of security issues.

❖ Confidentiality

Confidentiality is the prevention of the intentional or unintentional unauthorized disclosure of contents. Loss of confidentiality can occur in many ways. For example, loss of confidentiality can occur through the intentional release of private company information or through a misapplication of network rights.

Some of the elements of telecommunications used to ensure confidentiality are as follows:

- Network security protocols
- Network authentication services
- Data encryption services

❖ Integrity

Integrity is the guarantee that the message sent is the message received and that the message is not intentionally or unintentionally altered. Loss of integrity can occur through an intentional attack to change information (for example, a website defacement) or, more commonly, unintentionally (data is accidentally

altered by an operator). Integrity also contains the concept of non repudiation of a message source, which we will describe later.

Some of the elements used to ensure integrity include the following:

- Firewall services
- Communications security management
- Intrusion detection services

❖ **Availability**

This concept refers to the elements that create reliability and stability in networks and systems. It ensures that connectivity is accessible when needed, allowing authorized users to access the network or systems.

Some of the elements that are used to ensure availability are as follows:

- Fault tolerance for data availability, such as backups and redundant disk systems
- Acceptable logins and operating process performance
- Reliable and interoperable security processes and network security mechanisms

There are also several other important concepts and terms that apply equally well to traditional IT computing and cloud computing:

Identification — The means by which users claim their identities to a system. Most commonly used for access control, identification is necessary for authentication and authorization.

Authentication — The testing or reconciliation of evidence of a user's identity. It establishes the user's identity and ensures that the users are who they say they are.

Accountability — A system's capability to determine the actions and behaviors of a single individual within a system and to identify that particular individual. Audit trails and logs support accountability.

Authorization — The rights and permissions granted to an individual or process that enable access to a computer resource. Once a user's identity and authentication are established, authorization levels determine the extent of a user's system rights.

Privacy — The level of confidentiality and privacy protection given to a user in a system. This is often an important component of security controls. Privacy not only guarantees the fundamental tenet of confidentiality of company data, but also guarantees the data's level of privacy, which is being used by the operator. We examine privacy risks in more detail in the following section.

Privacy and Compliance Risks

Privacy laws attempt to provide protection to an individual from unauthorized disclosure of the individual's personally identifiable information (PII).

- Names
- Postal address information, other than town or city, state, and zip code
- Telephone numbers
- Fax numbers
- Electronic mail addresses
- Social security numbers
- Medical record numbers
- Health plan beneficiary numbers
- Account numbers
- Certificate/license numbers

- Vehicle identifiers and serial numbers, including license plate numbers
- Device identifiers and serial numbers
- Web Universal Resource Locators (URLs)
- Internet Protocol (IP) address numbers
- Biometric identifiers, including fingerprints and voiceprints
- Full face photographic images and any comparable images

Threats to Infrastructure, Data, and Access Control

This includes the following:

- Communications and network security as it relates to voice, data, multimedia, and facsimile transmissions in terms of local area, wide area, and remote access networks
- Internet/intranet/extranet in terms of firewalls, routers, gateways, and various protocols
- A threat is simply any event that, if realized, can cause damage to a system and create a loss of confidentiality, availability, or integrity.
- A *vulnerability* is a weakness in a system that can be exploited by a threat. Reducing the vulnerable aspects of a system can reduce the risk and impact of threats on the system.

Common threats to both cloud and traditional infrastructure include the following:

- **Eavesdropping** — Data scavenging, traffic or trend analysis, social engineering, economic or political espionage, sniffing, dumpster diving, keystroke monitoring, and shoulder surfing are all types of eavesdropping to gain information or to create a foundation for a later attack. Eavesdropping is a primary cause of the failure of confidentiality.
- **Fraud** — Examples of fraud include collusion, falsified transactions, data manipulation, and other altering of data integrity for gain.
- **Theft** — Examples of theft include the theft of information or trade secrets for profit or unauthorized disclosure, and physical theft of hardware or software.
- **Sabotage** — Sabotage includes denial-of-service (DoS) attacks, production delays, and data integrity sabotage.
- **External attack** — Examples of external attacks include malicious cracking, scanning, and probing to gain infrastructure information, demon dialing to locate an unsecured modem line, and the insertion of a malicious code or virus.

The most common types of attacks are,

- ▶ ***Inappropriate System Use***
- ▶ ***Eavesdropping***
- ▶ ***Network Intrusion***
- ▶ ***Denial-of-Service (DoS) Attacks***
- ▶ ***Session Hijacking Attacks***
- ▶ ***Fragmentation Attacks***

It is important for the information security professional to understand and identify other types of attacks

- Back-Door
- Spoofing
- Man-in-the-Middle
- TCP Hijacking
- Trojan Horses and Malware

Cloud Computing Security Challenges

The introduction of cloud services presents many challenges to an organization. When an organization migrates to consuming cloud services, and especially public cloud services, much of the computing system infrastructure will now be under the control of a third-party Cloud Services Provider (CSP).

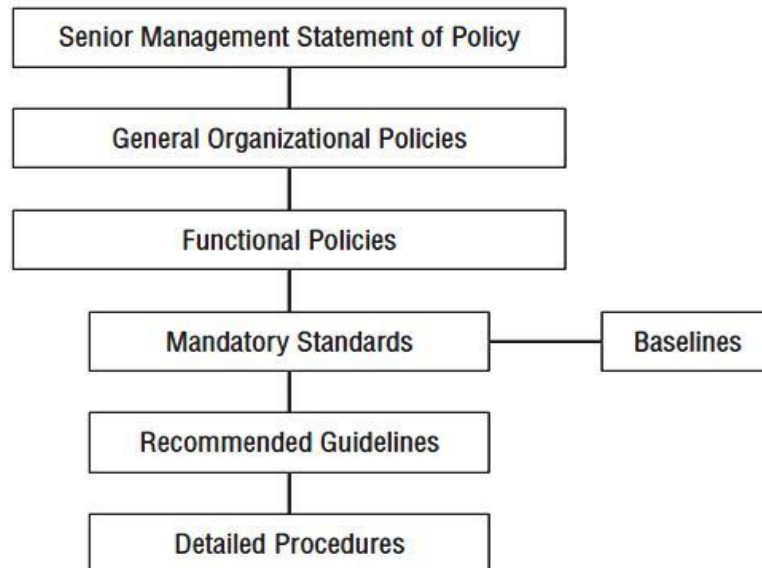


Figure 5-1: Security policy hierarchy

The main tasks of a CSIRT (computer security incident response team) are as follows:

- Analysis of an event notification
- Response to an incident if the analysis warrants it
- Escalation path procedures
- Resolution, post-incident follow-up, and reporting to the appropriate parties

Five immutable laws of virtualization security must be understood and used to drive security decisions:

Law 1: All existing OS-level attacks work in the exact same way.

Law 2: The hypervisor attack surface is additive to a system's risk profile.

Law 3: Separating functionality and/or content into VMs will reduce risk.

Law 4: Aggregating functions and resources onto a physical platform will increase risk.

Law 5: A system containing a "trusted" VM on an "un trusted" host has a higher risk level than a system containing a "trusted" host with an "un trusted" VM.

VM Security Recommendations

The following security implementation techniques are required for most computer systems, and are still best practices for virtualized systems. These areas include physical security, patching, and remote management techniques.

Hardening the Host Operating System

Some of these techniques include the following:

- Use strong passwords, such as lengthy, hard to guess passwords with letters, numbers, and symbol combinations, and change them often.
- Disable unneeded services or programs, especially networked services.

- Require full authentication for access control.
- The host should be individually firewalled.
- Patch and update the host regularly, after testing on a nonproduction unit.

Limiting Physical Access to the Ho

When attackers can access a host they can do the following:

- Use OS-Specific keystrokes to kill processes, monitor resource usage, or shut down the machine, commonly without needing a valid login account and password
- Reboot the machine, booting to external media with a known root password
- Steal files using external media (floppy, CD/DVD-RW, USB/flash drives, etc.)
- Capture traffic coming into or out of the network interfaces
- Remove one or more disks, mounting them in a machine with a known administrator or root password, potentially providing access to the entire contents of the host and guest VMs
- Simply remove the entire machine

Using Encrypted Communications

Encryption technologies, such as Secure HTTP (HTTPS), encrypted Virtual Private Networks (VPNs), Transport Layer Security (TLS), Secure Shell (SSH), and so on should be used to provide secure communications links between the host domain and the guest domain, or from hosts to management systems.

- Disabling Background Tasks
- Updating and Patching
- Enabling Perimeter Defense on the VM
- Implementing File Integrity Checks
- Maintaining Backups

VM-Specific Security Techniques

A fundamental requirement for a successful virtualization security process is recognizing the dynamic nature of virtual machines.

Hardening the Virtual Machine

This hardening can include many steps, such as the following:

- Putting limits on virtual machine resource consumption
- Configuring the virtual network interface and storage appropriately
- Disabling or removing unnecessary devices and services
- Ensuring that components that might be shared across virtual network devices are adequately isolated and secured
- Keeping granular and detailed audit logging trails for the virtualized infrastructure.

Harden the Hypervisor

Root Secure the Monitor

Implement Only One Primary Function per VM

Firewall Any Additional VM Ports

Harden the Host Domain

These include but are not limited to:

- Remove unnecessary accounts and groups.
- Disable unnecessary services.
- Remove unnecessary binaries, libraries, and files.
- Firewall network access to the host.
- Install monitoring or Host Intrusion Detection Systems.
- Ensure that the Host Domain is not accessible from the Guest Domains.
- Ensure that monitoring or remote console interfaces for the Host Domain are not accessible via the Guest Domains.
- Ensure that the Guest Domains cannot directly affect any network storage or other resources that the Host Domain relies on for boot, configuration, or authentication.

Use Unique NICs (network interfaces) for Sensitive VMs Disconnect Unused Devices

Securing VM Remote Access

Standard practices for remote administration include the following:

- Strong authentication practices should be employed:
 - Two-factor authentication
 - Strong passwords
 - One-time passwords
- Private/public PKI key pairs
- Use encrypted communications only, such as a SSH or VPNs.
- MAC address or IP address filtering should be employed.
- Telnet access to the unit should be denied, as it does not encrypt the communications channel.

⇒ **Cloud Computing Security Architecture Design**



The security posture of a cloud system is based on its security architecture. While there is no standard definition for security architecture, the Open Security Alliance (OSA) defines *security architecture* as “the design artifacts that describe how the security controls are positioned, and how they relate to the overall IT Architecture. These controls serve the purpose to maintain the system’s quality attributes, among them confidentiality, integrity, availability, accountability and assurance”.



A *security architecture* as “a cohesive security design, which addresses the requirements (e.g., authentication, authorization, etc.) and in particular the risks of a particular environment/scenario, and specifies what security controls are to be applied where.



Architectural Considerations

General Issues: A variety of topics influence and directly affect the cloud security architecture. They include such factors as compliance, security management, administrative issues, controls, and security awareness.

- 1) **Compliance** - The cloud vendor should provide transparency to the client by supplying information about storage used, processing characteristics, and other relevant account information. The accessibility of a client’s data by the provider’s system engineers and certain other employees.

- 2) **Security Management** - Security architecture involves effective security management to realize the benefits of cloud computation. Proper cloud security management and administration should identify management issues in critical areas such as access control, vulnerability analysis, change control, incident response, fault tolerance, and disaster recovery and business continuity planning.
- 3) **Controls** - The objective of cloud security controls is to reduce vulnerabilities to a tolerable level and minimize the effects of an attack. To achieve this, an organization must determine what impact an attack might have, and the likelihood of loss.
 - **Deterrent controls** —reduce the likelihood of a deliberate attack.
 - **Preventative controls** — Protect vulnerabilities and make an attack unsuccessful or reduce its impact. Preventative controls inhibit attempts to violate security policy.
 - **Corrective controls** — reduce the effect of an attack.
 - **Detective controls** — Discover attacks and trigger preventative or corrective controls. Detective controls warn of violations or attempted violations of security policy and include such controls as intrusion detection systems, organizational policies, video cameras, and motion detectors.
- 4) **Complementary Actions** -Additional activities involved in cloud security management include the following:
 - Management and monitoring of service levels and service-level agreements.
 - Acquisition of adequate data to identify and analyze problem situations through instrumentation and dashboards.
 - Reduction of the loss of critical information caused by lack of controls.
 - Proper management of data on an organization’s distributed computing resources. Data centralized on the cloud reduces the potential for data loss in organizations with large numbers of laptop computers and other personal computing devices.
 - Monitoring of centrally stored cloud information, as opposed to having to examine data distributed throughout an organization on a variety of computing and storage devices.
 - Provisioning for rapid recovery from problem situations.

DATA SECURITY- Information Classification

The following classification terms are typical of those used in the private sector and are applicable to cloud data:

Public data — Information that is similar to unclassified information; all of a company’s information that does not fit into any of the next categories can be considered public. While it’s unauthorized disclosure may be against policy, it is not expected to impact seriously or adversely the organization, its employees, and/or its customers.

Sensitive data — Information that requires a higher level of classification than normal data. This information is protected from a loss of confidentiality as well as from a loss of integrity due to an unauthorized alteration. This classification applies to information that requires special precautions to ensure its integrity by protecting it from unauthorized modification or deletion. It is information that requires a higher-than-normal assurance of accuracy and completeness.

Private data — This classification applies to personal information that is intended for use within the organization. Its unauthorized disclosure could seriously and adversely impact the organization and/or its employees. For example, salary levels and medical information are considered private.

Confidential data — This classification applies to the most sensitive business information that is intended strictly for use within the organization. Its unauthorized disclosure could seriously and adversely impact the organization, its stockholders, its business partners, and/or its customers.



Classification Criteria

Several criteria may be used to determine the classification of an information object:

Value — Value is the number one commonly used criteria for classifying data in the private sector. If the information is valuable to an organization or its competitors, then it needs to be classified.

Age — The classification of information might be lowered if the information's value decreases over time. In the U.S. Department of Defense, some classified documents are automatically declassified after a predetermined time period has passed.

Useful life — If the information has been made obsolete due to new information, substantial changes in the company, or other reasons, the information can often be declassified.

Personal association — If information is personally associated with specific individuals or is addressed by a privacy law, it might need to be classified. For example, investigative information that reveals informant names might need to remain classified.



Information Classification Procedures

There are several steps in establishing a classification system. These are the steps in priority order:

- ✓ Identify the appropriate administrator and data custodian. The data custodian is responsible for protecting the information, running backups, and performing data restoration.
- ✓ Specify the criteria for classifying and labeling the information.
- ✓ Classify the data by its owner, who is subject to review by a supervisor.
- ✓ Specify and document any exceptions to the classification policy.
- ✓ Specify the controls that will be applied to each classification level.
- ✓ Specify the termination procedures for declassifying the information or for transferring custody of the information to another entity.
- ✓ Create an enterprise awareness program about the classification controls.



Employee Termination

It is important to understand the impact of employee terminations on the integrity of information stored in a cloud environment.

- ✗ The removal of access privileges, computer accounts, authentication tokens.
- ✗ The briefing on the continuing responsibilities of the terminated employee for confidentiality and privacy.
- ✗ The return of company computing property, such as laptops.
- ✗ Continued availability of data. In both the manual and the electronic worlds, this may involve documenting procedures or filing schemes, such as how documents are stored on the hard disk and how they are backed up.
- ✗ Employees should be instructed whether or not to “clean up” their PC before leaving.
- ✗ If cryptography is used to protect data, the availability of cryptographic keys to management personnel must be ensured.



Security Awareness, Training, and Education

Security awareness is often overlooked as element affecting cloud security architecture because most of a security practitioner's time is spent on controls, intrusion detection, risk assessment, and proactively or reactively administering security.

- Improving awareness of the need to protect system resources
- Developing skills and knowledge so computer users can perform their jobs more securely
- Building in-depth knowledge, as needed, to design, implement, or operate security programs for organizations and systems.

In general, a computer security awareness and training program should encompass the following seven steps:

1. Identify program scope, goals, and objectives.
2. Identify training staff.
3. Identify target audiences.
4. Motivate management and employees.
5. Administer the program.
6. Maintain the program.
7. Evaluate the program.

Training and Education

Training is different from awareness in that it provides security information in a more formalized manner, such as classes, workshops, or individualized instruction. The following types of training are related to cloud security:

- Security-related job training for operators and specific users
- Awareness training for specific departments or personnel groups with security-sensitive positions
- Technical security training for IT support personnel and system administrators
- Advanced training for security practitioners and information systems auditors
- Security training for senior managers, functional managers, and business unit managers

Trusted Cloud Computing

Trusted cloud computing can be viewed as a computer security architecture that is designed to protect cloud systems from malicious intrusions and attacks, and ensure that computing resources will act in a specific, predictable manner as intended. A trusted cloud computing system will protect data in use by hypervisors and applications, protect against unauthorized access to information, provide for strong authentication, apply encryption to protect sensitive data that resides on stolen or lost devices, and support compliance through hardware and software mechanisms.

Secure Execution Environments and Communications

In a cloud environment, applications are run on different servers in a distributed mode. These applications interact with the outside world and other applications and may contain sensitive information whose inappropriate access would be harmful to a client.

- **Secure Execution Environment**

Configuring computing platforms for secure execution is a complex task; and in many instances it is not performed properly because of the large number of parameters that are involved. Another major concern in secure execution of code is the widespread use of “unsafe” programming languages such as C and C++ instead of more secure languages such as object-oriented Java and structured, object-oriented C#.

- **Secure Communications** : As opposed to having managed, secure communications among the computing resources internal to an organization, movement of applications to the cloud requires a reevaluation of communications security. These communications apply to both data in motion and data at rest.

■ **Confidentiality** — Ensures that only those who are supposed to access data can retrieve it. Loss of confidentiality can occur through the intentional release of private company information or through a misapplication of network rights. Some of the elements of telecommunications used to ensure confidentiality are as follows:

- Network security protocols
- Network authentication services
- Data encryption services

■ **Integrity** — Ensures that data has not been changed due to an accident or malice. Integrity is the guarantee that the message sent is the message received and that the message is not intentionally or unintentionally altered. Integrity also contains the concept of non repudiation of a message source. Some of the constituents of integrity are as follows:

- Firewall services
- Communications Security Management
- Intrusion detection services

■ **Availability** — Ensures that data is accessible when and where it is needed, and that connectivity is accessible when needed, allowing authorized users to access the network or systems. Also included in that assurance is the guarantee that security services for the security practitioner are usable when they are needed. Some of the elements that are used to ensure availability are as follows:

- Fault tolerance for data availability, such as backups and redundant disk systems
- Acceptable logins and operating process performances
- Reliable and interoperable security processes and network security mechanisms



APIs(Application Programming Interfaces)

Common vulnerabilities such as weak antivirus software, unattended computing platforms, poor passwords, weak authentication mechanisms, and inadequate intrusion detection that can impact communications must be more stringently analyzed, and proper APIs must be used.

For example, in using IaaS, a cloud client typically communicates with cloud server instances through Representational State Transfer (REST) client/server model or Simple Object Access Protocol (SOAP) APIs.



Virtual Private Networks

Another important method to secure cloud communications is through a virtual private network (VPN). A VPN is created by building a secure communications link between two nodes by emulating the properties of a point-to-point private link. A VPN can be used to facilitate secure remote access into the cloud, securely connect two networks together, or create a secure data tunnel within a network.

Remote Access VPNs

A VPN can be configured to provide remote access to corporate resources over the public Internet to maintain confidentiality and integrity.

Network-to-Network VPNs

A VPN is commonly used to connect two networks, perhaps the main corporate LAN and a remote branch office LAN, through the Internet. This connection can use either dedicated lines to the Internet or dial-up connections to the Internet.

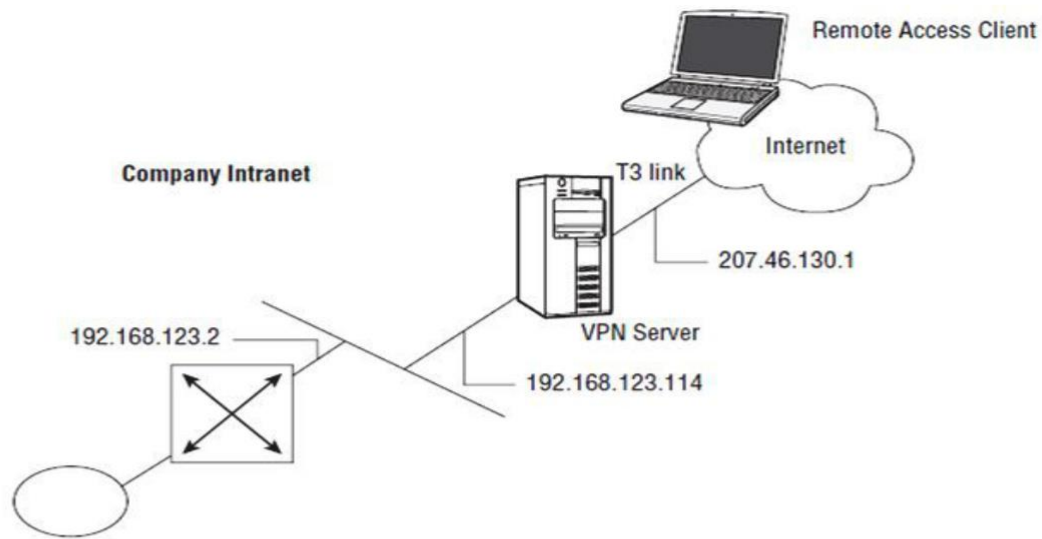


Figure 6-2: Remote access VPN configuration

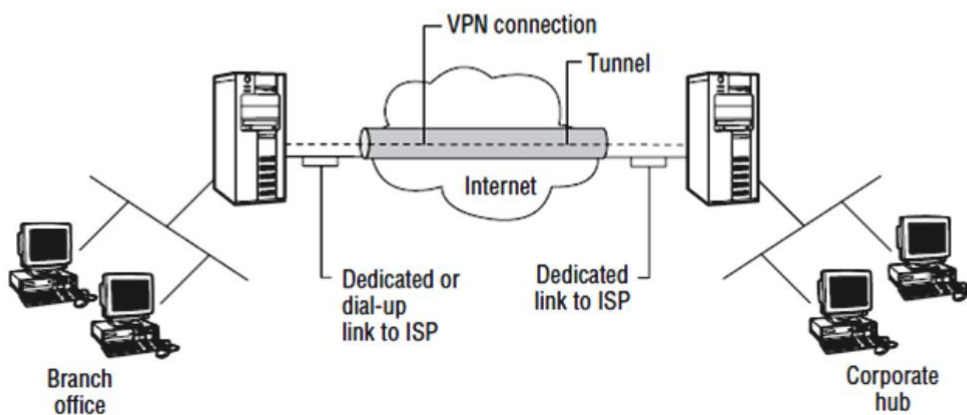


Figure 6-3: A network-to-network VPN configuration

Public Key Infrastructure and Encryption Key Management

To secure communications, data that is being exchanged with a cloud should be encrypted, calls to remote servers should be examined for imbedded malware, and digital certificates should be employed and managed.

- Digital certificates
- Certificate authority (CA)
- Registration authorities
- Policies and procedures
- Certificate revocation
- Non repudiation support
- Time stamping
- Lightweight Directory Access Protocol (LDAP)
- Security-enabled applications

Identity Management and Access Control

Identity management and access control are fundamental functions required for secure cloud computing. The simplest form of identity management is logging on to a computer system with a user ID and password.

True identity management, such as is required for cloud computing, requires more robust authentication, authorization, and access control. It should determine what resources are authorized to be accessed by a user or process by using technology such as biometrics or smart cards, and determine when a resource has been accessed by unauthorized entities.

Identification and authentication are the keystones of most access control systems. Identification is the act of a user professing an identity to a system, usually in the form of a username or user logon ID to the system.

Authentication is verification that the user's claimed identity is valid, and it is usually implemented through a user password at logon. Authentication is based on the following three factor types:

- **Type 1** — Something you know, such as a personal identification number (PIN) or password
- **Type 2** — Something you have, such as an ATM card or smart card
- **Type 3** — Something you are (physically), such as a fingerprint or retina scan

❖ Passwords

Because passwords can be compromised, they must be protected. In the ideal case, a password should be used only once. This “one-time password,” or OTP, provides maximum security because a new password is required for each new logon. A password that is the same for each logon is called a *static password*. A password that changes with each logon is termed a *dynamic password*.

❖ Tokens

Tokens, in the form of small, hand-held devices, are used to provide passwords.

❖ Memory Cards

Memory cards provide nonvolatile storage of information, but they do not have any processing capability. A memory card stores encrypted passwords and other related identifying information. A telephone calling card and an ATM card are examples of memory cards.

❖ Smart Cards

Smart cards provide even more capability than memory cards by incorporating additional processing power on the cards. These credit-card-size devices comprise microprocessor and memory and are used to store digital signatures, private keys, passwords, and other personal information.

❖ Biometrics

An alternative to using passwords for authentication in logical or technical access control is *biometrics*. Biometrics is based on the Type 3 authentication mechanism — something you are. Biometrics is defined as an automated means of identifying or authenticating the identity of a living person based on physiological or behavioral characteristics.

The following are typical biometric characteristics that are used to uniquely authenticate an individual's identity:

- **Fingerprints** Fingerprint characteristics are captured and stored. Typical CERs are 4–5%.
- **Retina scans** The eye is placed approximately two inches from a camera and an invisible light source scans the retina for blood vessel patterns. CERs are approximately 1.4%.

- **Iris scans** A video camera remotely captures iris patterns and characteristics. CER values are around 0.5%.
- **Hand geometry** Cameras capture three-dimensional hand characteristics. CERs are approximately 2%.
- **Voice** — Sensors capture voice characteristics, including throat vibrations and air pressure, when the subject speaks a phrase. CERs are in the range of 10%.
- **Handwritten signature dynamics** The signing characteristics of an individual making a signature are captured and recorded. Typical characteristics including writing pressure and pen direction. CERs are not published at this time.
Other types of biometric characteristics include facial and palm scans.

► Implementing Identity Management

Realizing effective identity management requires a high-level corporate commitment and dedication of sufficient resources to accomplish the task. Typical undertakings in putting identity management in place include the following:

- Establishing a database of identities and credentials
- Managing users' access rights
- Enforcing security policy
- Developing the capability to create and modify accounts
- Setting up monitoring of resource accesses
- Installing a procedure for removing access rights
- Providing training in proper procedures

Access Control

Access control is intrinsically tied to identity management and is necessary to preserve the confidentiality, integrity, and availability of cloud data.

Three things that must be considered for the planning and implementation of access control mechanisms are threats to the system, the system's vulnerability to these threats, and the risk that the threats might materialize. These concepts are defined as follows:

- **Threat** — An event or activity that has the potential to cause harm to the information systems or networks
- **Vulnerability** — A weakness or lack of a safeguard that can be exploited by a threat, causing harm to the information systems or networks
- **Risk** — The potential for harm or loss to an information system or network; the probability that a threat will materialize

► Controls

Controls are implemented to mitigate risk and reduce the potential for loss. Two important control concepts are *separation of duties* and the principle of *least privilege*.

Control measures can be administrative, logical (also called technical), and physical in their implementation.

- Administrative controls include policies and procedures, security awareness training, background checks, work habit checks, a review of vacation history, and increased supervision.
- Logical or technical controls involve the restriction of access to systems and the protection of information. Examples of these types of controls are encryption, smart cards, access control lists, and transmission protocols.

- Physical controls incorporate guards and building security in general, such as the locking of doors, the securing of server rooms or laptops, the protection of cables, the separation of duties, and the backing up of files.

Models for Controlling Access

Controlling access by a subject (an active entity such as an individual or process) to an object (a passive entity such as a file) involves setting up access rules. These rules can be classified into three categories or models.

Mandatory Access Control

The authorization of a subject's access to an object depends upon labels, which indicate the subject's *clearance*, and the *classification or sensitivity* of the object. *Rule-based access control* is a type of mandatory access control because rules determine this access (such as the correspondence of clearance labels to classification labels), rather than the identity of the subjects and objects alone.

Discretionary Access Control

With discretionary access control, the subject has authority, within certain limitations, to specify what objects are accessible. For example, access control lists (ACLs) can be used. An *access control triple* consists of the user, program, and file, with the corresponding access privileges noted for each user. This type of access control is used in local, dynamic situations in which the subjects must have the discretion to specify what resources certain users are permitted to access.

Nondiscretionary Access Control

A central authority determines which subjects can have access to certain objects based on the organizational security policy. The access controls might be based on the individual's role in the organization (role-based) or the subject's responsibilities and duties (task-based). Access control can also be characterized as *context-dependent* or *content-dependent*. Context-dependent access control is a function of factors such as location, time of day, and previous access history.

⇒ **Autonomic Security**

- ✓ Autonomic computing refers to a self-managing computing model in which computer systems reconfigure themselves in response to changing conditions and are self-healing.
- ✓ The promise of autonomic computing will take a number of years to fully materialize, but it offers capabilities that can improve the security of information systems and cloud computing in particular.
- ✓ The ability of autonomic systems to collect and interpret data and recommend or implement solutions can go a long way toward enhancing security and providing for recovery from harmful events.

► **Autonomic Systems**

Autonomic systems are based on the human autonomic nervous system, which is self-managing, monitors changes that affect the body, and maintains internal balances.

Examples of events that would have to be handled autonomously include the following:

- Malicious attacks
- Hardware or software faults
- Excessive CPU utilization
- Power failures

- Organizational policies
 - Inadvertent operator errors
 - Interaction with other systems
 - Software updates

IBM introduced the concept of autonomic computing and its eight defining characteristics⁵ as follows:

- 1) **Self-awareness** — An autonomic application/system “knows itself” and is aware of its state and its behaviors.
- 2) **Self-configuring** — An autonomic application/system should be able to configure and reconfigure itself under varying and unpredictable conditions.
- 3) **Self-optimizing** — An autonomic application/system should be able to detect sub-optimal behaviors and optimize itself to improve its execution.
- 4) **Self-healing** — An autonomic application/system should be able to detect and recover from potential problems and continue to function smoothly.
- 5) **Self-protecting** — An autonomic application/system should be capable of detecting and protecting its resources from both internal and external attack and maintaining overall system security and integrity.
- 6) **Context-aware** — An autonomic application/system should be aware of its execution environment and be able to react to changes in the environment.
- 7) **Open** — An autonomic application/system must function in a heterogeneous world and should be portable across multiple hardware and software architectures. Consequently, it must be built on standard and open protocols and interfaces.
- 8) **Anticipatory** — An autonomic application/system should be able to anticipate, to the extent possible, its needs and behaviors and those of its context, and be able to manage itself proactively.

Autonomic Protection

Autonomic self-protection involves detecting a harmful situation and taking actions that will mitigate the situation. These systems will also be designed to predict problems from analysis of sensory inputs and initiate corrective measures.

An autonomous system security response is based on network knowledge, capabilities of connected resources, information aggregation, the complexity of the situation, and the impact on affected applications.

Autonomous protection systems should, therefore, adhere to the following guidelines:

- Minimize overhead requirements.
 - Be consistent with security policies.
 - Optimize security-related parameters.
 - Minimize impact on performance.
 - Minimize potential for introducing new vulnerabilities.
- Conduct regression analysis and return to previous software versions if problems are introduced by changes.
- Ensure that reconfiguration processes are secure.

Autonomic Self-Healing: Autonomic self healing systems can provide the capability to detect and repair software problems and identify hardware faults without manual intervention. The objective of the autonomous self-healing process is to keep the elements operating according to their design specifications. **109**