Cloud Computing all unit notes

Cloud Computing (S A Engineering College)

PANIMALAR INSTITUTE OF TECHNOLOGY
(JAISAKTHI EDUCATIONAL TRUST)
CHENNAI 600 123

DEPARTMENT OF CSE

CS8791 CLOUD COMPUTING

IV YEAR – VII SEMESTER

LECTURE NOTES - UNIT I

UNIT I INTRODUCTION

Introduction to Cloud Computing – Definition of Cloud – Evolution of Cloud Computing – Underlying Principles of Parallel and Distributed Computing – Cloud Characteristics – Elasticity in Cloud – On-demand Provisioning.

1.1INTRODUCTION

EVOLUTION OF DISTRIBUTED COMPUTING

Grids enable access to shared computing power and storage capacity from your desktop. Clouds enable access to leased computing power and storage capacity from your desktop.

- Grids are an **open source** technology. Resource users and providers alike can understand and contribute to the management of their grid
- Clouds are a **proprietary** technology. Only the resource provider knows exactly how their cloud manages data, job queues, security requirements and so on.
- The concept of grids was proposed in 1995. The Open science grid (OSG) started in 1995 The EDG (European Data Grid) project began in 2001.
- In the late 1990`s Oracle and EMC offered early private cloud solutions . However the term cloud computing didn't gain prominence until 2007.

SCALABLE COMPUTING OVER THE INTERNET

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.

The Age of Internet Computing

- high-performance computing (HPC) applications is no longer optimal for measuring system performance
- The emergence of computing clouds instead demands high-throughput computing (HTC) systems built with parallel and distributed computing technologies
- We have to upgrade data centers using fast servers, storage systems, and high-bandwidth networks.

The Platform Evolution

- From 1950 to 1970, a handful of mainframes, including the IBM 360 and CDC 6400

- From 1960 to 1980, lower-cost minicomputers such as the DEC PDP 11 and VAX Series
- From 1970 to 1990, we saw widespread use of personal computers built with VLSI microprocessors.
- From 1980 to 2000, massive numbers of portable computers and pervasive devices appeared in both wired and wireless applications
- Since 1990, the use of both HPC and HTC systems hidden in clusters, grids, or Internet clouds has proliferated

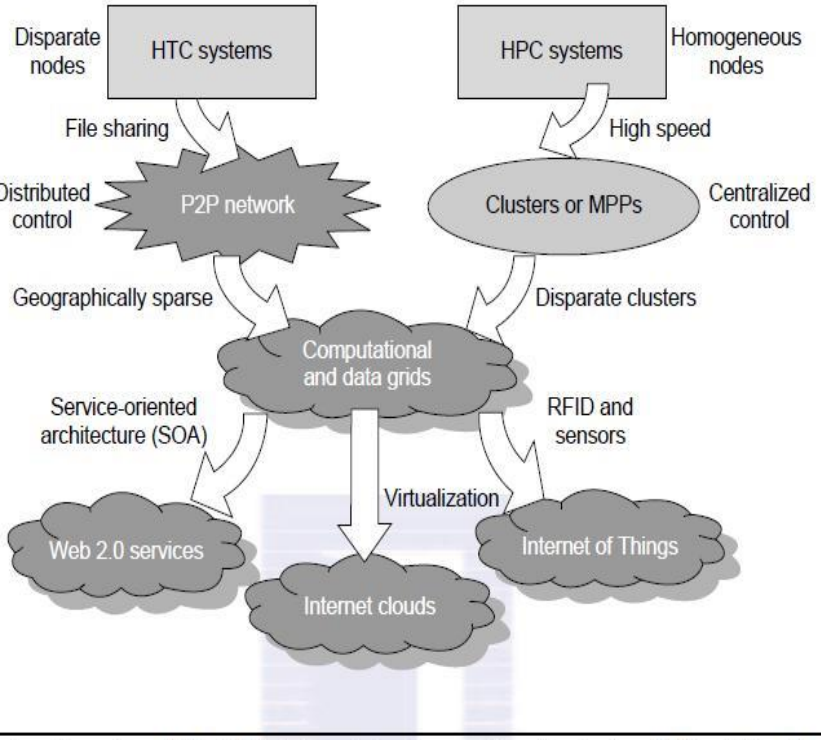


FIGURE 1.1

Evolutionary trend toward parallel, distributed, and cloud computing with clusters, MPPs, P2P networks, grids, clouds, web services, and the Internet of Things.

On the HPC side, supercomputers (massively parallel processors or MPPs) are gradually replaced by clusters of cooperative computers out of a desire to share computing resources. The cluster is often a collection of homogeneous compute nodes that are physically connected in close range to one another.

- On the HTC side, peer-to-peer (P2P) networks are formed for distributed file sharing and content delivery applications. A P2P system is built over many client machines (a concept we will discuss further in Chapter 5). 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.

- 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.
- The development of market-oriented high-end computing systems is undergoing a strategic change from an HPC paradigm to an HTC paradigm. This HTC paradigm pays more attention to high-flux computing. The main application for high-flux computing is in Internet searches and web services by millions or more users simultaneously. The performance goal thus shifts to measure high throughput or the number of tasks completed per unit of time. HTC technology needs to not only improve in terms of batch processing speed, but also address the acute problems of cost, energy savings, security, and reliability at many data and enterprise computing centers.
- 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). These new paradigms are only briefly introduced here.
- 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.

Terms

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. Inter processor communication is accomplished through shared memory or via message passing. Acomputer 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 . As an alternative to the preceding terms, some in the high-tech community prefer the term concurrent computing or concurrent programming. These terms typically refer to the union of parallel computing and distributing computing, although biased practitioners may interpret them differently.
- **Ubiquitous computing** refers to computing with pervasive devices at any place and time using wired or wireless communication. The Internet of Things (IoT) is a networked connection of everyday objects including computers, sensors, humans, etc. The IoT is supported by Internet clouds to achieve ubiquitous computing with any object at any place and time. Finally, the term Internet computing is even broader and covers all computing paradigms over the Internet. This book covers all the aforementioned computing paradigms, placing more emphasis on distributed and cloud computing and their working systems, including the clusters, grids, P2P, and cloud systems.

Internet of Things

- The traditional Internet connects machines to machines or web pages to web pages. The concept of the IoT was introduced in 1999 at MIT .

- The IoT refers to the networked interconnection of everyday objects, tools, devices, or computers. One can view the IoT as a wireless network of sensors that interconnect all things in our daily life.
- It allows objects to be sensed and controlled remotely across existing network infrastructure

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.

Clusters of Cooperative Computers

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

- In the past, clustered computer systems have demonstrated impressive results in handling heavy workloads with large data sets.

Cluster Architecture

cluster built around a low-latency, high bandwidth interconnection network. This network can be as simple as a SAN or a LAN (e.g., Ethernet).

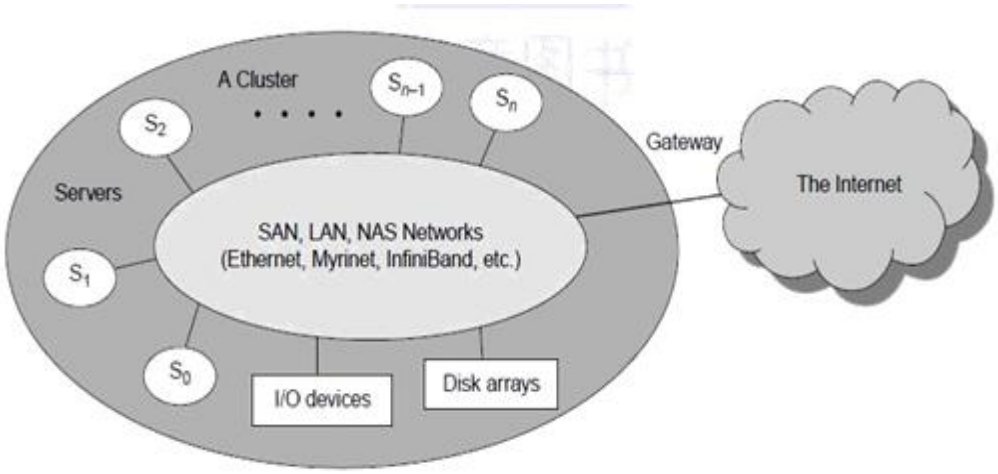


Figure 1.2 Clusters of Servers

Figure 1.2shows 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, or InfiniBand switches.
- Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes. The cluster is connected to the Internet via a virtual private network (VPN) gateway.
- The gateway IP address locates the cluster. The system image of a computer is decided by the way the OS manages the shared cluster resources.

Most clusters have loosely coupled node computers. All resources of a server node are managed by their own OS. Thus, most clusters have multiple system images as a result of having many autonomous nodes under different OS control.

1.3.1.2 Single-System Image(SSI)

- Ideal cluster should merge multiple system images into a single-system image (SSI).
- Cluster designers desire a cluster operating system or some middleware to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes.

An SSI is an illusion created by software or hardware that presents a collection of resources as one integrated, powerful resource. SSI makes the cluster appear like a single machine to the user. A cluster with multiple system images is nothing but a collection of independent computers.

1.3.1.3 Hardware, Software, and Middleware Support

- Clusters exploring massive parallelism are commonly known as MPPs. Almost all HPC clusters in the Top 500 list are also MPPs.
- The building blocks are computer nodes (PCs, workstations, servers, or SMP), special communication software such as PVM, and a network interface card in each computer node.

Most clusters run under the Linux OS. The computer nodes are interconnected by a high-bandwidth network (such as Gigabit Ethernet, Myrinet, InfiniBand, etc.). Special cluster middleware supports are needed to create SSI or high availability (HA). Both sequential and parallel applications can run on the cluster, and special parallel environments are needed to facilitate use of the cluster resources. For example, distributed memory has multiple images. Users may want all distributed memory to be shared by all servers by forming distributed shared

memory (DSM). Many SSI features are expensive or difficult to achieve at various cluster operational levels. Instead of achieving SSI, many clusters are loosely coupled machines. Using virtualization, one can build many virtual clusters dynamically, upon user demand.

Cloud Computing over the Internet

- 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.
- A cloud allows workloads to be deployed and scaled out quickly through rapid provisioning of virtual or physical machines.
- The cloud supports redundant, self-recovering, highly scalable programming models that allow workloads to recover from many unavoidable hardware/software failures.
- Finally, the cloud system should be able to monitor resource use in real time to enable rebalancing of allocations when needed.

a. Internet Clouds

- Cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and data sets dynamically .The idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers.
- Cloud computing leverages its low cost and simplicity to benefit both users and providers.
- Machine virtualization has enabled such cost-effectiveness. Cloud computing intends to satisfy many user applications simultaneously.

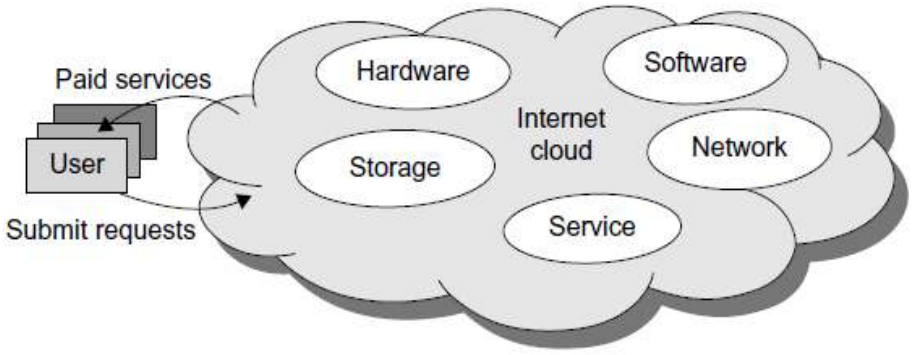


Figure 1.3 Internet Cloud

b. The Cloud Landscape

- The cloud ecosystem must be designed to be secure, trustworthy, and dependable. Some computer users think of the cloud as a centralized resource pool. Others consider the cloud to be a server cluster which practices distributed computing over all the servers Traditionally, a distributed computing system tends to be owned and operated by an autonomous administrative domain (e.g., a research laboratory or company) for on-premises computing needs.
- Cloud computing as an on-demand computing paradigm resolves or relieves us from these problems.

Three Cloud service Model in a cloud landscape

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 OS on specific applications.
- The user does not manage or control the underlying cloud infrastructure, but can specify when to request and release the needed resources.

Platform as a Service (PaaS)

• This model enables the user to deploy user-built applications onto a virtualized cloud platform. PaaS includes middleware, databases, development tools, and some runtime support such as Web 2.0 and Java.

• The platform includes both hardware and software integrated with specific programming interfaces.

• The provider supplies the API and software tools (e.g., Java, Python, Web 2.0, .NET). The user is freed from managing the cloud infrastructure.

Software as a Service (SaaS)

• This refers to browser-initiated application software over thousands of paid cloud customers. The SaaS model applies to business processes, industry applications, consumer relationship management (CRM), enterprise resources planning (ERP), human resources (HR), and collaborative applications. On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications.

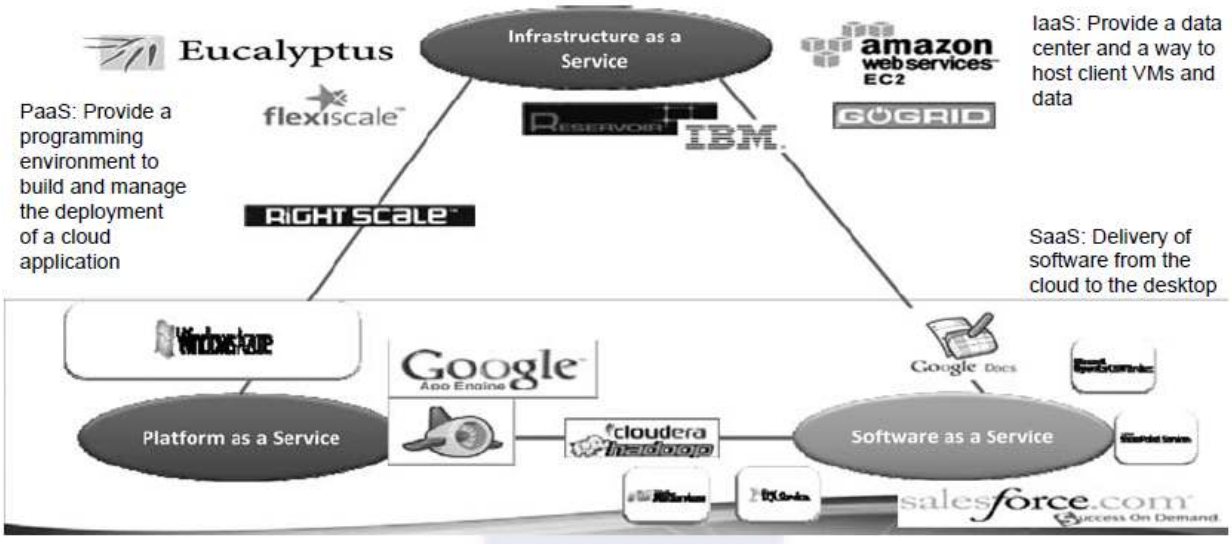


Figure 1.4 The Cloud Landscape in an application

Internet clouds offer four deployment modes: private, public, managed, and hybrid . These modes demand different levels of security implications. The different SLAs imply that the security responsibility is shared among all the cloud providers, the cloud resource consumers, and the third party cloud-enabled software providers. Advantages of cloud computing have been advocated by many IT experts, industry leaders, and computer science researchers.

Reasons to adapt the cloud for upgraded Internet applications and web services:

1. Desired location in areas with protected space and higher energy efficiency
2. Sharing of peak-load capacity among a large pool of users, improving overall utilization
3. Separation of infrastructure maintenance duties from domain-specific application development
4. Significant reduction in cloud computing cost, compared with traditional computing paradigms
5. Cloud computing programming and application development
6. Service and data discovery and content/service distribution
7. Privacy, security, copyright, and reliability issues
8. Service agreements, business models, and pricing policies

- Cloud computing is using the internet to access someone else's software running on someone else's hardware in someone else's data center.
- The user sees only one resource (HW, Os) but uses virtually multiple os. HW resources etc..
- Cloud architecture effectively uses virtualization
- A model of computation and data storage based on “pay as you go” access to “unlimited” remote data center capabilities
- A cloud infrastructure provides a framework to manage scalable, reliable, on-demand access to applications
- Cloud services provide the “invisible” backend to many of our mobile applications
- High level of elasticity in consumption
- Historical roots in today’s Internet apps
    - Search, email, social networks, e-com sites
    - File storage (Live Mesh, Mobile Me)

1.2 Definition

- **“The National Institute of Standards and Technology (NIST) defines cloud computing as a "pay-per-use model for enabling available, convenient and 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."**

Cloud Computing Architecture

- Architecture consists of 3 tiers
    - Cloud Deployment Model
    - Cloud Service Model
    - Essential Characteristics of Cloud Computing

Essential Characteristics 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 a service provider.

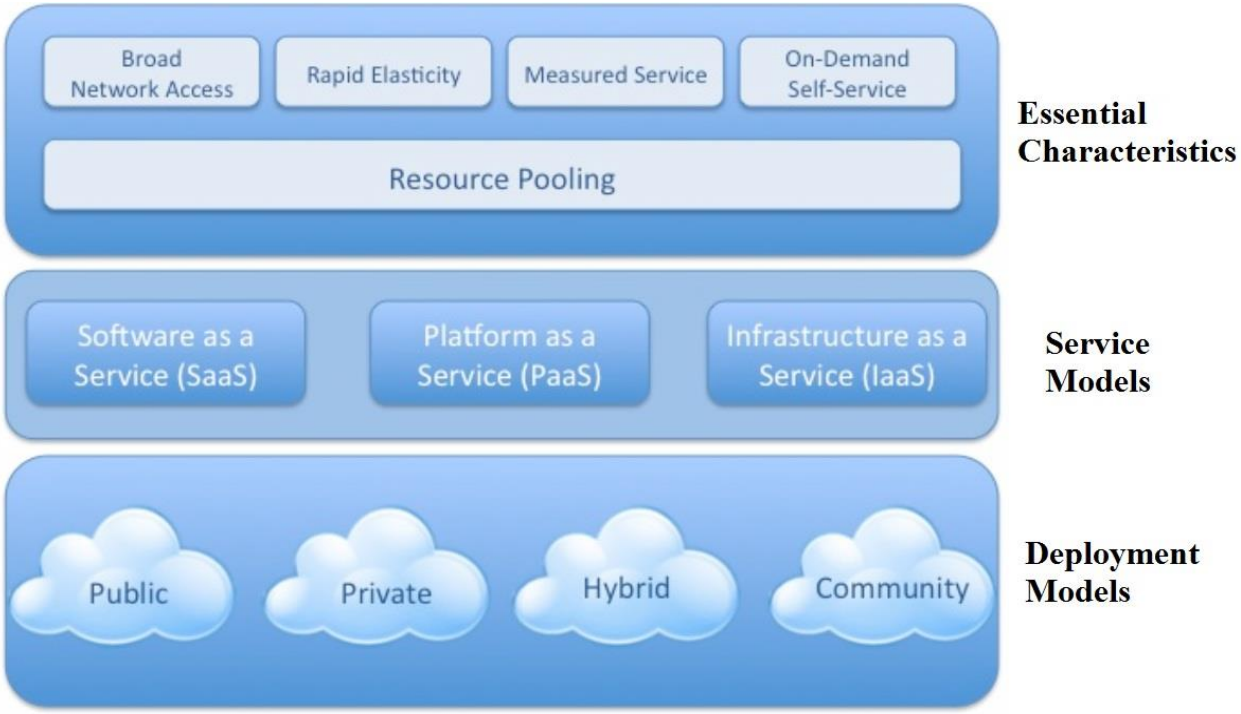


Figure 1.5 Cloud Computing Architecture

Essential Characteristics 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, laptops, and PDAs) as well as other traditional or cloudbased software services.

Essential Characteristics 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.

Essential Characteristics 4

- **Rapid elasticity.**
  - Capabilities can be rapidly and elastically provisioned - in some cases automatically - to quickly scale out; and rapidly released to quickly scale in.
  - To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

Essential Characteristics 5

- **Measured service.**
  - Cloud systems automatically control and optimize resource usage by leveraging a metering capability at some level of abstraction appropriate to the type of service.
  - Resource usage can be monitored, controlled, and reported - providing transparency for both the provider and consumer of the service.

Cloud Service Models

- Cloud **S**oftware as a Service (**S**aaS)
- Cloud **P**latform as a Service (**P**aaS)
- Cloud **I**nfrastructure as a Service (**I**aaS)

SaaS

- SaaS is a licensed software offering on the cloud and pay per use
- SaaS is a software delivery methodology that provides licensed multi-tenant access to software and its functions remotely as a Web-based service. Usually billed based on usage
  - Usually multi tenant environment
  - Highly scalable architecture
- Customers do not invest on software application programs
- 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 a thin client interface such as a web browser (e.g., web-based email).

- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, data or even individual application capabilities, with the possible exception of limited user specific application configuration settings.

SaaS providers

- Google's Gmail, Docs, Talk etc
- Microsoft's Hotmail, Sharepoint
- SalesForce,
- Yahoo, Facebook

Infrastructure as a Service (IaaS)

- IaaS is the delivery of technology infrastructure (mostly hardware) as an on demand, scalable service
  - Usually billed based on usage
  - Usually multi tenant virtualized environment
  - Can be coupled with Managed Services for **OS** and application support
  - User can choose his OS, storage, deployed app, networking components
  -

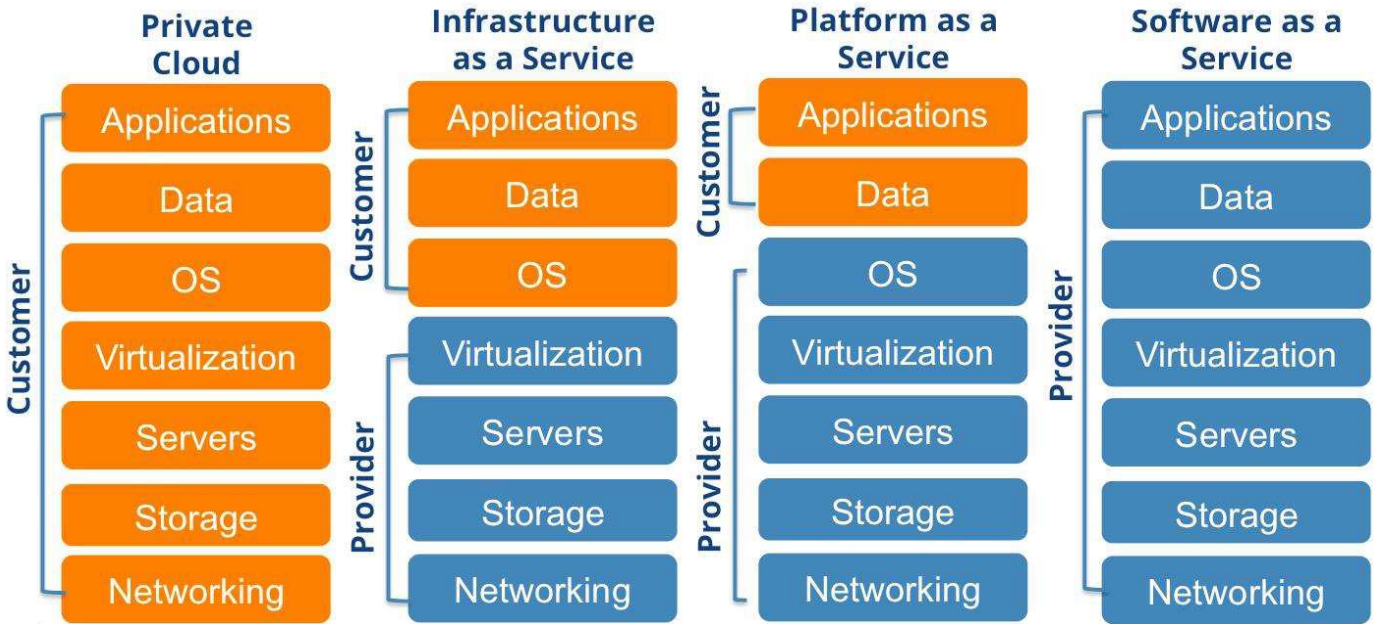


Figure 1.6 Cloud Service Model

- The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources.

- Consumer is able to deploy and run arbitrary software, which may include operating systems and applications.
- The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

IaaS providers

- Amazon Elastic Compute Cloud (EC2)
  - Each instance provides 1-20 processors, upto 16 GB RAM, 1.69TB storage
- RackSpace Hosting
  - Each instance provides 4 core CPU, upto 8 GB RAM, 480 GB storage
- Joyent Cloud
  - Each instance provides 8 CPUs, upto 32 GB RAM, 48 GB storage
- Go Grid
  - Each instance provides 1-6 processors, upto 15 GB RAM, 1.69TB storage

Platform as a Service (PaaS)

- PaaS provides all of the facilities required to support the complete life cycle of building, delivering and deploying web applications and services entirely from the Internet. Typically applications must be developed with a particular platform in mind
  - Multi tenant environments
  - Highly scalable multi tier architecture
- The capability provided to the consumer is to deploy onto the cloud infrastructure consumer created or acquired applications created using programming languages and tools supported by the provider.
- 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 application hosting environment configurations.

PaaS providers

- Google App Engine
  - Python, Java, Eclipse

- Microsoft Azure
  - .Net, Visual Studio
- Sales Force
  - Apex, Web wizard
- TIBCO,
- VMware,
- Zoho

Cloud Computing - Opportunities and Challenges

- It enables services to be used without any understanding of their infrastructure.
- Cloud computing works using economies of scale
- It potentially lowers the outlay expense for start up companies, as they would no longer need to buy their own software or servers.
- Cost would be by on-demand pricing.
- Vendors and Service providers claim costs by establishing an ongoing revenue stream.
- Data and services are stored remotely but accessible from "anywhere"

Cloud Computing – Pros

- Lower computer costs
- Instant software updates:
  - When the application is web-based, updates happen automatically
- Improved document format compatibility
- e capacity:
  - Cloud computing offers virtually limitless storage
  - • Increased data reliability:

Cloud Computing – Cons

- Need of Internet :
  - A dead Internet connection means no work and in areas where Internet connections are few or inherently unreliable, this could be a deal-breaker.
  - Requires a constant Internet connection

- Can be slow:
  - Even with a fast connection, web-based applications can sometimes be slower than accessing a similar software program on your desktop PC.
- Disparate Protocols :
  - Each cloud systems uses different protocols and different APIs – Standards yet to evolve.

1.3 Evolution of Cloud Computing

Evolution of Cloud Computing

- Cloud Computing Leverages dynamic resources to deliver a large number of services to end users.
- It is High Throughput Computing(HTC) paradigm
- It enables users to share access to resources from anywhere at any time

II Hardware Evolution

- In 1930, binary arithmetic was developed
  - computer processing technology, terminology, and programming languages.

• In 1939,Electronic computer was developed

- Computations were performed using vacuum-tube technology.

• In 1941, Konrad Zuse's Z3 was developed

- Support both floating-point and binary arithmetic.

There are four generations

- First Generation Computers
- Second Generation Computers
- Third Generation Computers
- Fourth Generation Computers

a.First Generation Computers

Time Period : 1942 to 1955

Technology : Vacuum Tubes

Size : Very Large System

Processing : Very Slow

Examples:

1.ENIAC (Electronic Numerical Integrator and Computer)

2.EDVAC(Electronic Discrete Variable Automatic Computer)

Advantages:

• It made use of vacuum tubes which was the advanced technology at that time

• Computations were performed in milliseconds.

Disadvantages:

• very big in size, weight was about 30 tones.

• very costly.

• Requires more power consumption

•Large amount heat was generated.

b.Second Generation Computers

Time Period : 1956 to 1965.

Technology : Transistors

Size : Smaller

Processing : Faster

o Examples

Honeywell 400

IBM 7094

Advantages

- Less heat than first generation.
- Assembly language and punch cards were used for input.
- Low cost than first generation computers.
- Computations was performed in microseconds.
- Better Portability as compared to first generation

Disadvantages:

- A cooling system was required.
- Constant maintenance was required.
- Only used for specific purposes

c.Third Generation Computers

Time Period : 1966 to 1975

Technology : ICs (Integrated Circuits)

Size : Small as compared to 2nd generation computers

Processing : Faster than 2nd generation computers

Examples

• PDP-8 (Programmed Data Processor)

• PDP-11

Advantages

• These computers were cheaper as compared to generation computers.

• They were fast and reliable.

• IC not only reduce the size of the computer but it also improves the performance of the computer

• Computations was performed in nanoseconds

Disadvantages

• IC chips are difficult to maintain.

• The highly sophisticated technology required for the manufacturing of IC chips.

•Air Conditioning is required

d.Fourth Generation Computers

Time Period : 1975 to Till Date

Technology : Microprocessor

Size : Small as compared to third generation computer

Processing : Faster than third generation computer

Examples

• IBM 4341

• DEC 10

Advantages:

- Fastest in computation and size get reduced as compared to the previous generation of computer. Heat generated is small.
- Less maintenance is required.

Disadvantages:

- The Microprocessor design and fabrication are very complex.
- Air Conditioning is required in many cases

III Internet Hardware Evolution

- Internet Protocol is the standard communications protocol used by every computer on the Internet.
- The conceptual foundation for creation of the Internet was significantly developed by three individuals.
  - Vannevar Bush — MEMIX (1930)
  - Norbert Wiener
  - Marshall McLuhan
- Licklider was founder for the creation of the AR PANET (Advanced Research Projects Agency Network)
- Clark deployed a minicomputer called an Interface Message Processor (IMP) at each site.
- Network Control Program (NCP)- first networking protocol that was used on the ARPANET

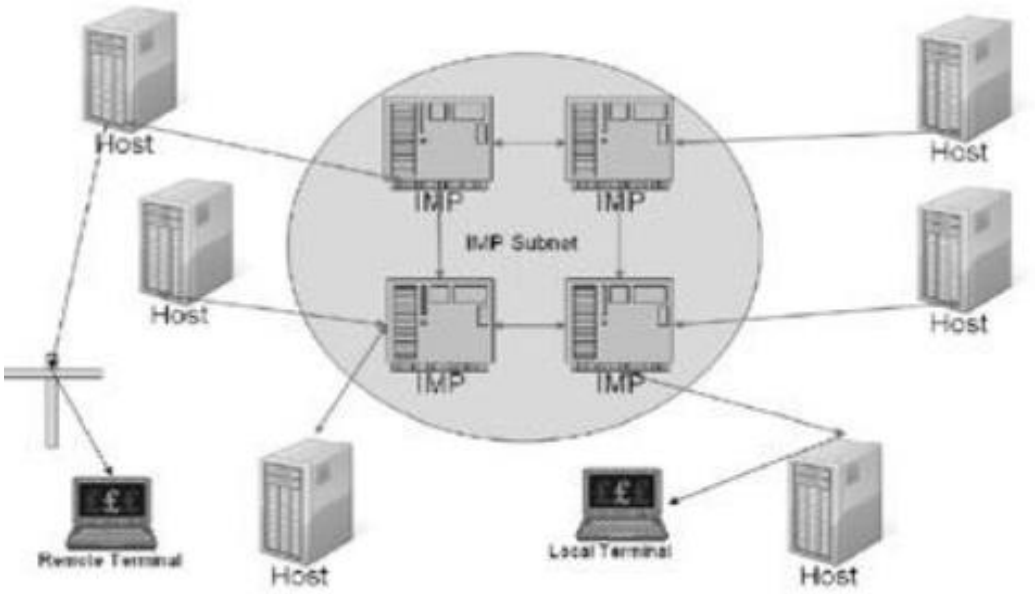


Figure 1.7 **IMP Architecture**

Internet Hardware Evolution

- Establishing a Common Protocol for the Internet
- Evolution of Ipv6
- Finding a Common Method to Communicate Using the Internet Protocol
- Building a Common Interface to the Internet
- The Appearance of Cloud Formations From One Computer to a Grid of Many

a.Establishing a Common Protocol for the Internet

- NCP essentially provided a transport layer consisting of the ARPANET Host-to-Host Protocol (AIIIIP) and the Initial Connection Protocol (ICP)
- Application protocols
  - File Transfer Protocol (FTP), used for file transfers,
  - Simple Mail Transfer Protocol (SMTP), used for sending email

Four versions of TCP/IP

- TCP vl
- TCP v2
- TCP v3 and IP v3,
- TCP v4 and IP v4

b.Evolution of Ipv6

- IPv4 was never designed to scale to global levels.
- To increase available address space, it had to process large data packets (i.e., more bits of data).
- To overcome these problems, Internet Engineering Task Force (IETF) developed IPv6, which was released in January 1995.
- Ipv6 is sometimes called the Next Generation Internet Protocol (IPNG) or TCP/IP v6.

c.Finding a Common Method to Communicate Using the Internet Protocol

- In the 1960s,the word ktpertext was created by Ted Nelson.
- In 1962, Engelbart's first project was Augment, and its purpose was to develop computer tools to augment human capabilities.
- He developed the mouse, Graphical user interface (GUI), and the first working hypertext system, named NLS (oN-Line System).

- NLS was designed to cross-reference research papers for sharing among geographically distributed researchers.
- In the 1980s, Web was developed in Europe by Tim Berners-Lee and Robert Cailliau

d.Building a Common Interface to the Internet

- Betters-Lee developed the first web browser featuring an integrated editor that could create hypertext documents.
- Following this initial success, Berners-Lee enhanced the server and browser by adding support for the FTP (File Transfer protocol)

Figure 1.8 First Web Browser

- Mosaic was the first widely popular web browser available to the general public. Mosaic support for graphics, sound, and video clips.
- In October 1994, Netscape released the first beta version of its browser, Mozilla 0.96b, over the Internet.
- In 1995, Microsoft Internet Explorer was developed that supports both a graphical Web browser and the name for a set of technologies.
- Mozilla Firefox. released in November 2004, became very popular almost immediately.

e.The Appearance of Cloud Formations From One Computer to a Grid of Many

- Two decades ago, computers were clustered together to form a single larger computer in order to simulate a supercomputer and greater processing power.
- In the early 1990s, Ian Foster and Carl Kesselman presented their concept of "The Grid." They used an analogy to the electricity grid, where users could plug in and use a (metered) utility service.
- A major problem in clustering model was data residency. Because of the distributed nature of a grid, computational nodes could be anywhere in the world.

- The Globus Toolkit is an open source software toolkit used for building grid systems and applications

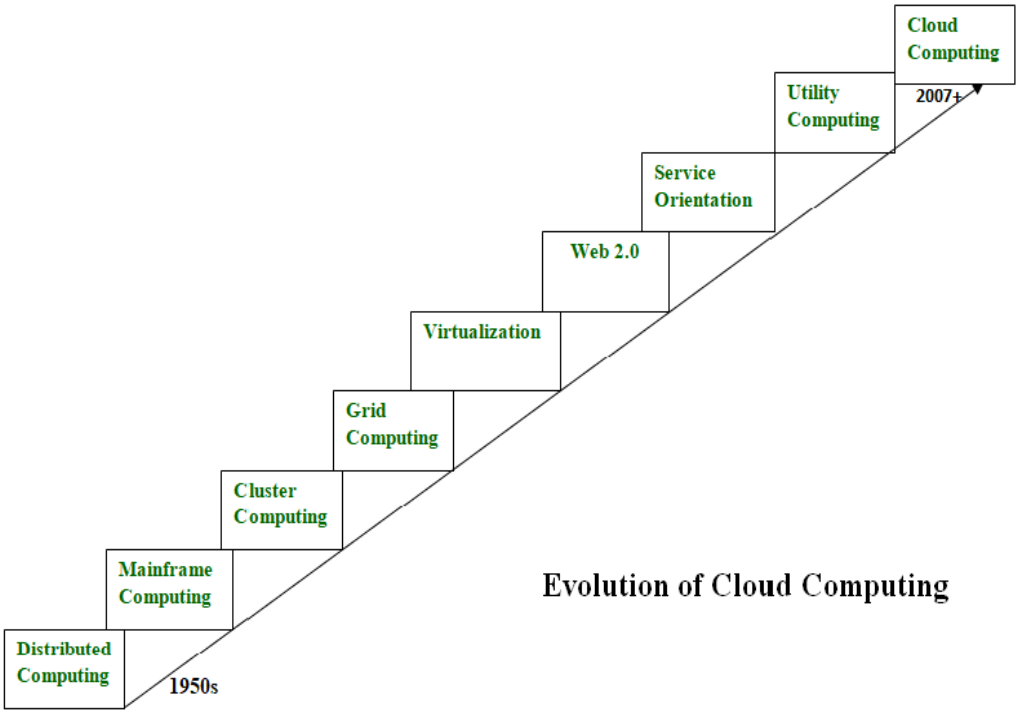


Figure 1.9 Evolution

Evolution of Cloud Services

2008-2009	**Google Application Engine** **Microsoft Azure**
2006	**S3 launches EC2**
2002	**Launch of Amazon Web Services**
1990	**The first milestone of cloud computing arrival of salesforce.com**
1960	**Super Computers** **Mainframes**

IV. SERVER VIRTUALIZATION

- Virtualization is a method of running multiple independent virtual operating systems on a single physical computer.
- This approach maximizes the return on investment for the computer.

- Virtualization technology is a way of reducing the majority of hardware acquisition and maintenance costs, which can result in significant savings for any company.
    - Parallel Processing
    - Vector Processing
    - Symmetric Multiprocessing Systems
    - Massively Parallel Processing Systems

a.Parallel Processing

- Parallel processing is performed by the simultaneous execution of program instructions that have been allocated across multiple processors.
- Objective: running a progran in less time.
- The next advancement in parallel processing-multiprogramming
- In a multiprogramming system, multiple programs submitted by users but each allowed to use the processor for a short time.
- This approach is known as "round-robin scheduling"(RR scheduling)

b.Vector Processing

- Vector processing was developed to increase processing performance by operating in a multitasking manner.
- Matrix operations were added to computers to perform arithmetic operations.
- This was valuable in certain types of applications in which data occurred in the form of vectors or matrices.
- In applications with less well-formed data, vector processing was less valuable.

c.Symmetric Multiprocessing Systems

- Symmetric multiprocessing systems (SMP) was developed to address the problem of resource management in master/slave models.
- In SMP systems, each processor is equally capable and responsible for managing the workflow as it passes through the system.
- The primary goal is to achieve sequential consistency

d.Massively Parallel Processing Systems

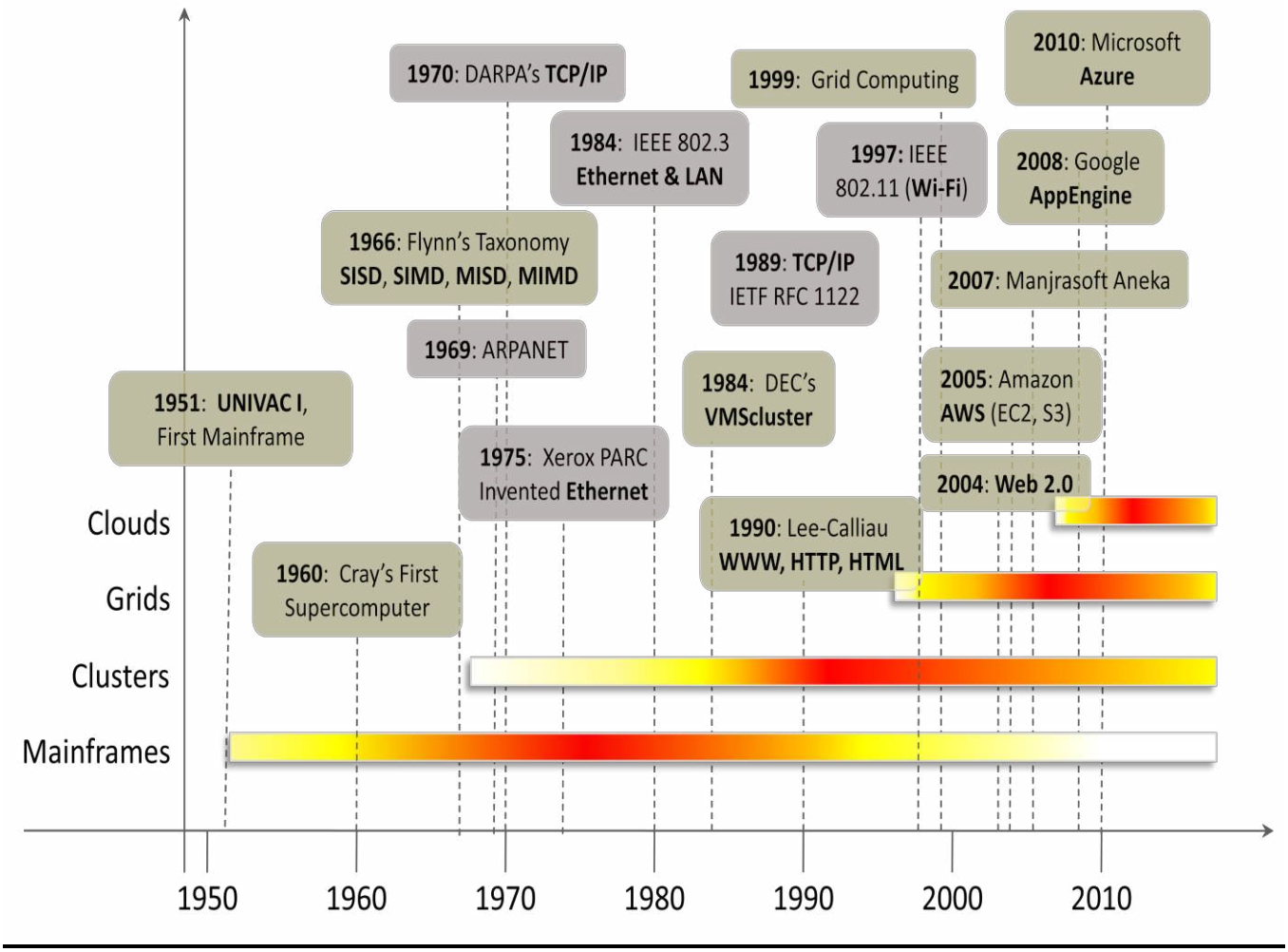
- In Massively Parallel Processing Systems, a computer system with many independent arithmetic units, which run in parallel.
- All the processing elements are interconnected to act as one very large computer.

- Early examples of MPP systems were the Distributed ArrayProcessor, the Goodyear MPP, the Connection Machine, and the Ultracomputer
- MPP machines are not easy to program, but for certain applications, such as data mining, they are the best solution

1.4 Principles of Parallel and Distributed Computing

- **Three major milestones have led to cloud computing evolution**
  - Mainframes: Large computational facilities leveraging multiple processing units. Even though mainframes cannot be considered as distributed systems, they offered large computational power by using multiple processors, which were presented as a single entity to users.

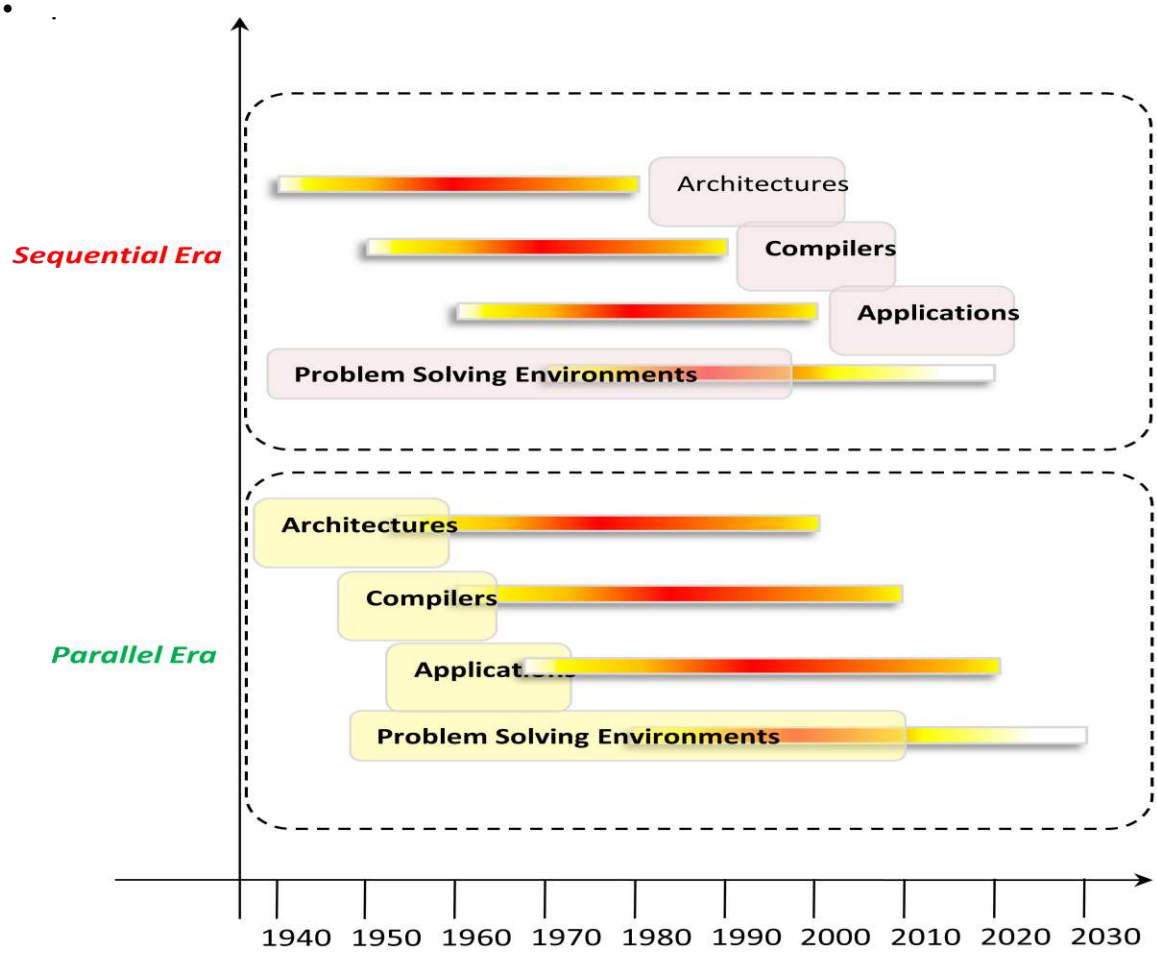
Mile Stones to Cloud computing Evolution



- Clusters: An alternative technological advancement to the use of mainframes and super computers.
- Grids
- Clouds

1.4.1Eras of Computing

- Two fundamental and dominant models of computing are ***sequential*** and ***parallel***.
  - The sequential era began in the 1940s, and Parallel(and distributed) computing era followed it within a decade.
- Four key elements of computing developed during three eras are
  - Architecture
  - Compilers
  - Applications
  - Problem solving environments
- .



- The computing era started with development in ***hardware architectures***, which actually enabled the creation of ***system software*** – particularly in the area of **compilers and operating systems** – which support the management of such systems and the development of ***applications***

- The term parallel computing and distributed computing are **often used interchangeably**, even though they mean **slightly different things**.
- The term **parallel implies a tightly coupled system**, where as **distributed systems refers to a wider class of system, including those that are tightly coupled**.
- More precisely, the term **parallel computing** refers to a model in which **the computation is divided among several processors sharing the same memory**.
- The architecture of **parallel computing system** is often characterized by the **homogeneity of components**: **each processor is of the same type and it has the same capability as the others.**

- The shared memory has a single address space, which is accessible to all the processors.
- Parallel programs are then broken down into several units of execution that can be allocated to different processors and can communicate with each other by means of shared memory.
- Originally parallel systems are considered as those architectures that featured multiple processors sharing the same physical memory and that were considered a single computer.
  - Over time, these restrictions have been relaxed, and parallel systems now include all architectures that are based on the concept of shared memory, whether this is physically present or created with the support of libraries, specific hardware, and a highly efficient networking infrastructure.
  - For example: a cluster of which of the nodes are connected through an InfiniBand network and configured with distributed shared memory system can be considered as a parallel system.
- The term distributed computing encompasses any architecture or system that allows the computation to be broken down into units and executed concurrently on different

computing elements, whether these are processors on different nodes, processors on the same computer, or cores within the same processor.

- Distributed computing includes a wider range of systems and applications than parallel computing and is often considered a more general term.
- Even though it is not a rule, the term distributed often implies that the locations of the computing elements are not the same and such elements might be heterogeneous in terms of hardware and software features.
- Classic examples of distributed computing systems are
  - Computing Grids
  - Internet Computing Systems

1.4.2 Elements of Parallel computing

- Silicon-based processor chips are reaching their physical limits. Processing speed is constrained by the speed of light, and the density of transistors packaged in a processor is constrained by thermodynamics limitations.
- A viable solution to overcome this limitation is to connect multiple processors working in coordination with each other to solve "Grand Challenge" problems.
- The first step in this direction led
  - To the development of parallel computing, which encompasses techniques, architectures, and systems for performing multiple activities in parallel.

a.Parallel Processing

- Processing of multiple tasks simultaneously on multiple processors is called ***parallel processing***.
- The parallel program consists of multiple active processes (tasks) simultaneously solving a given problem.
- A given task is divided into multiple subtasks using a divide-and-conquer technique, and each subtask is processed on a different central processing unit (CPU).
- Programming on multi processor system using the divide-and-conquer technique is called ***parallel programming***.
- Many applications today require more computing power than a traditional sequential computer can offer.

- Parallel Processing provides a cost effective solution to this problem by increasing the number of CPUs in a computer and by adding an efficient communication system between them.
- The workload can then be shared between different processors. This setup results in higher computing power and performance than a single processor a system offers.

Parallel Processing influencing factors

- The development of parallel processing is being influenced by many factors. The prominent among them include the following:
  - Computational requirements are ever increasing in the areas of both scientific and business computing. The technical computing problems, which require high-speed computational power, are related to
    - life sciences, aerospace, geographical information systems, mechanical design and analysis etc.
  - Sequential architectures are reaching mechanical physical limitations as they are constrained by the speed of light and thermodynamics laws.
    - The speed which sequential CPUs can operated is reaching saturation point (no more vertical growth), and hence an alternative way to get high computation speed is to connect multiple CPUs (opportunity for horizontal growth).
  - Hardware improvements in pipelining , super scalar, and the like are non scalable and require sophisticated compiler technology.
    - Developing such compiler technology is a difficult task.
  - Vector processing works well for certain kinds of problems. It is suitable mostly for scientific problems (involving lots of matrix operations) and graphical processing.
    - It is not useful for other areas, such as databases.
  - The technology of parallel processing is mature and can be exploited commercially
    - here is already significant R&D work on development tools and environments.
  - Significant development in networking technology is paving the way for

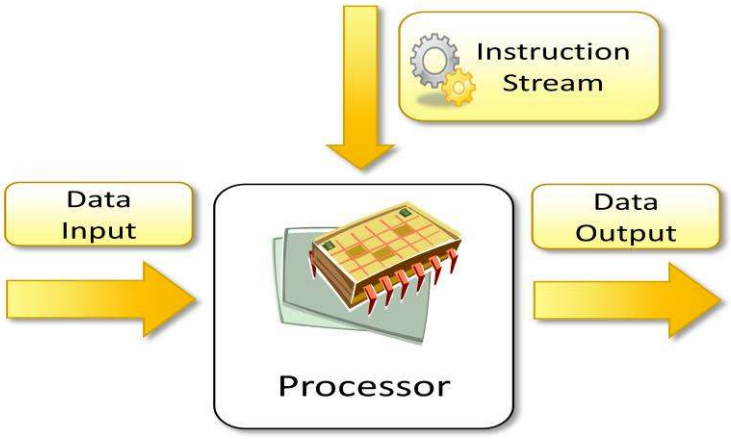
- heterogeneous computing.

b.Hardware architectures for parallel Processing

- The core elements of parallel processing are CPUs. Based on the number of instructions and data streams, that can be processed simultaneously, computing systems are classified into the following four categories:
  - Single-instruction, Single-data (SISD) systems
  - Single-instruction, Multiple-data (SIMD) systems
  - Multiple-instruction, Single-data (MISD) systems
  - Multiple-instruction, Multiple-data (MIMD) systems

(i)Single – Instruction , Single Data (SISD) systems

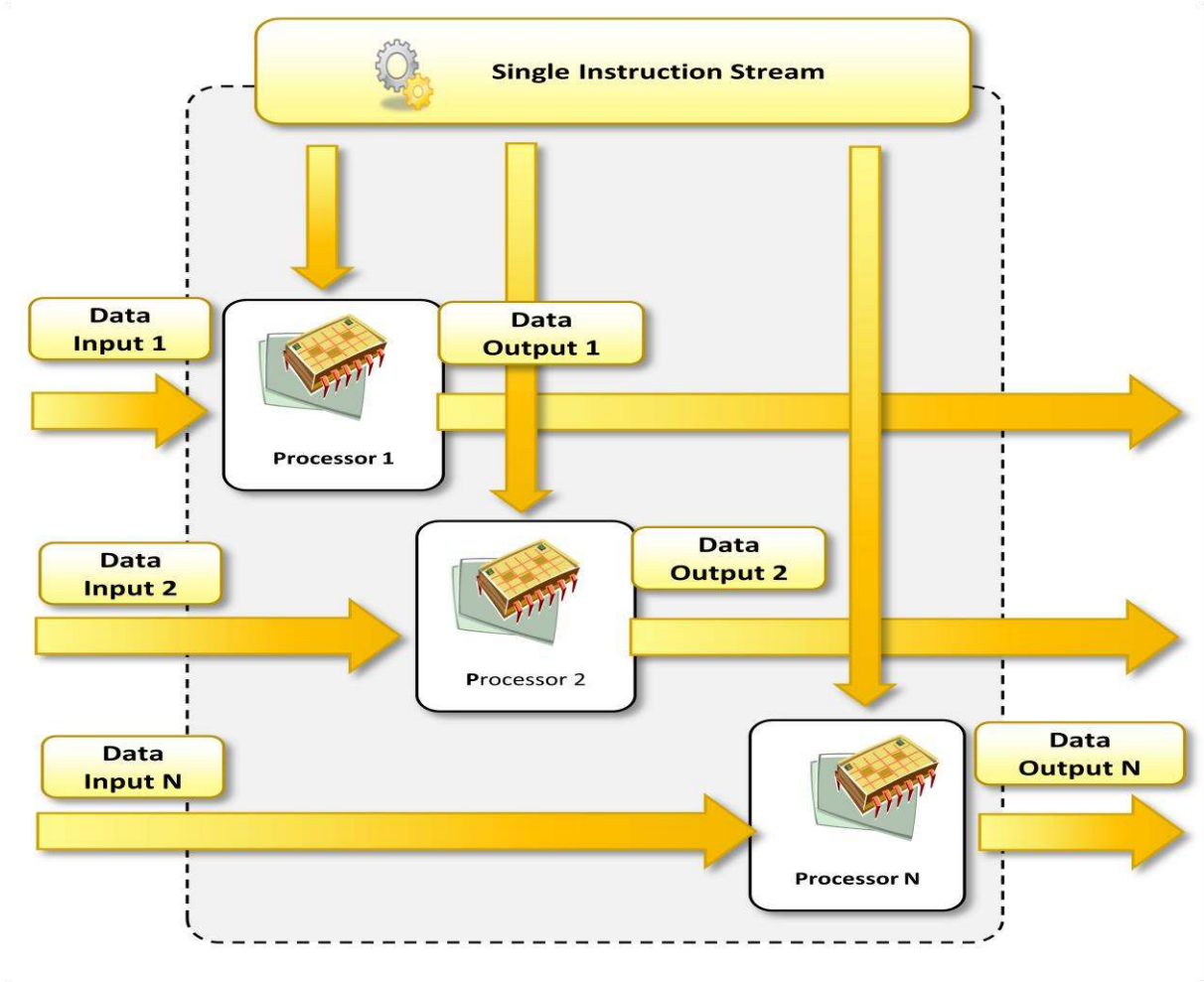
- SISD computing system is a uni-processor machine capable of executing a single instruction, which operates on a single data stream.
- Machine instructions are processed sequentially, hence computers adopting this model are popularly called sequential computers.
- Most conventional computers are built using SISD model.
- All the instructions and data to be processed have to be stored in primary memory.
- The speed of processing element in the SISD model is limited by the rate at which the computer can transfer information internally.
- Dominant representative SISD systems are IBM PC, Macintosh, and workstations.



(ii)Single – Instruction , Multiple Data (SIMD) systems

- SIMD computing system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams.
- Machines based on this model are well suited for scientific computing since they involve lots of vector and matrix operations.
- For instance statement Ci = Ai * Bi, can be passed to all the processing elements (PEs), organized data elements of vectors A and B can be divided into multiple sets (N- sets for N PE systems), and each PE can process one data set.

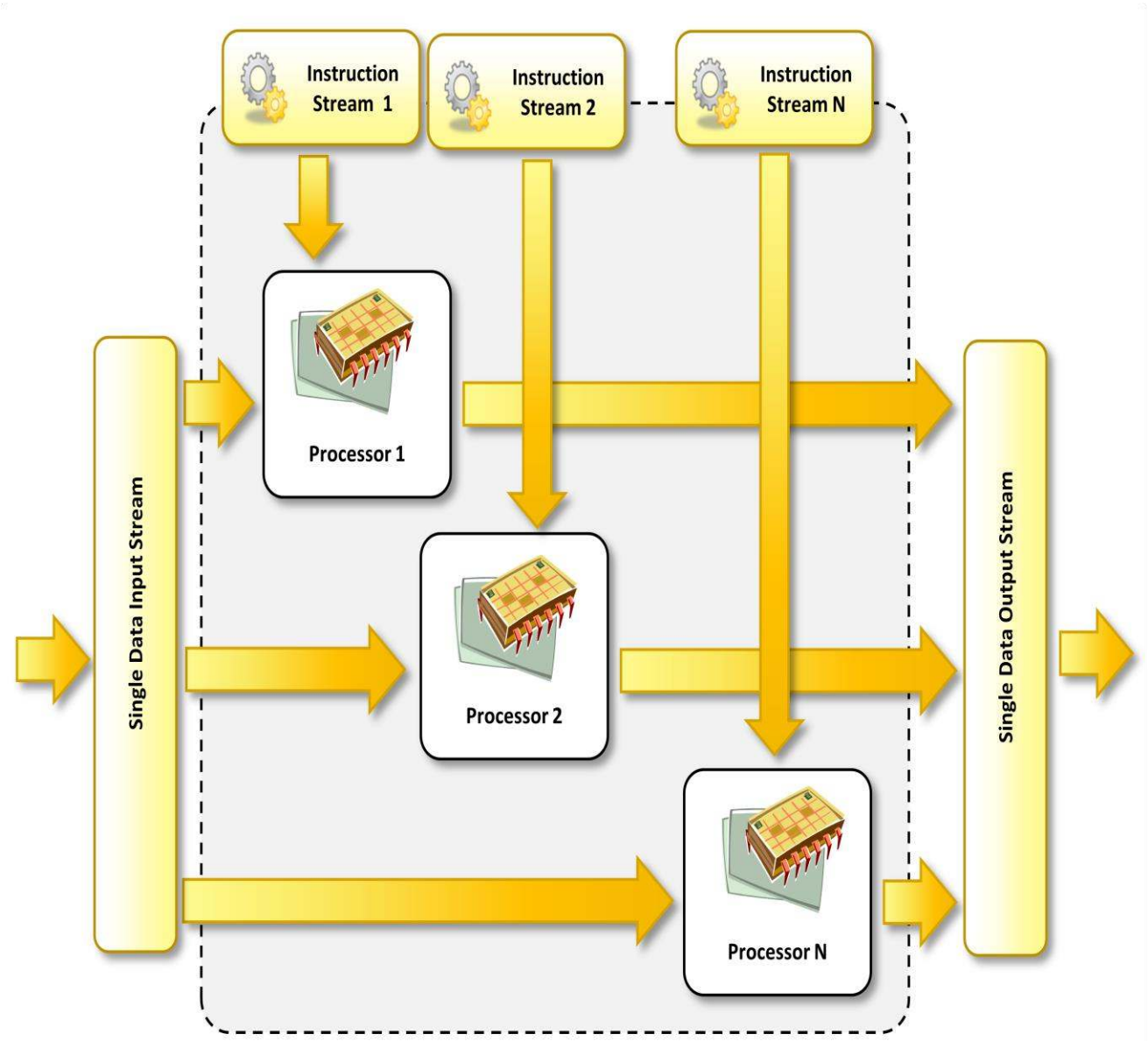
Dominant representative SIMD systems are Cray's Vector processing machine and Thinking Machines Cm*, and GPGPU accelerators



-

(iii)Multiple – Instruction , Single Data (MISD) systems

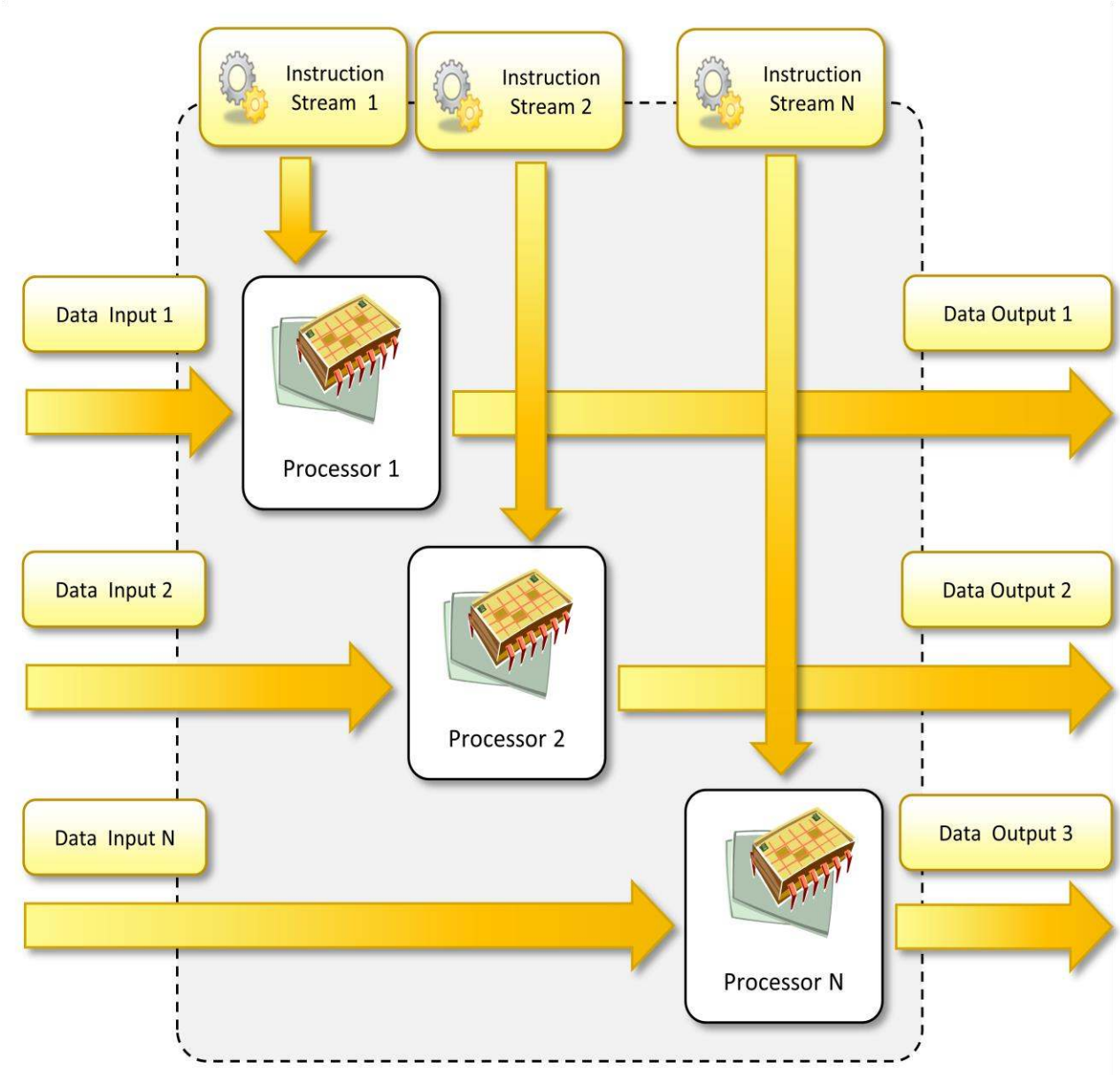
- MISD computing system is a multi processor machine capable of executing different instructions on different Pes all of them operating on the same data set.
- Machines built using MISD model are not useful in most of the applications.
- Few machines are built but none of them available commercially.
- This type of systems are more of an intellectual exercise than a practical configuration.



(iv)Multiple – Instruction , Multiple Data (MIMD) systems

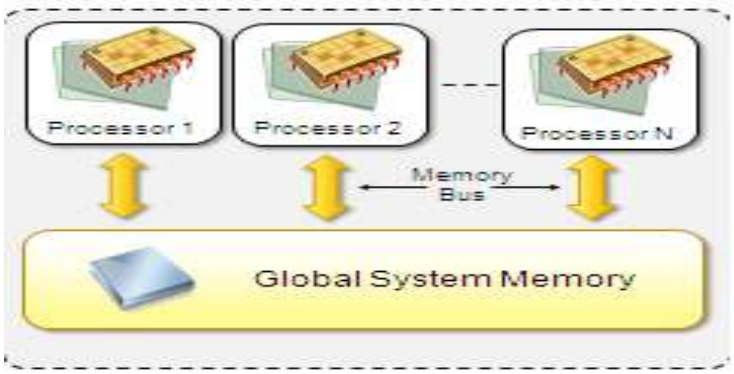
- MIMD computing system is a multi processor machine capable of executing multiple instructions on multiple data sets.
- Each PE in the MIMD model has separate instruction and data streams, hence machines built using this model are well suited to any kind of application.
- Unlike SIMD, MISD machine, PEs in MIMD machines work asynchronously,

MIMD machines are broadly categorized into shared-memory MIMD and distributed memory MIMD based on the way PEs are coupled to the main memory



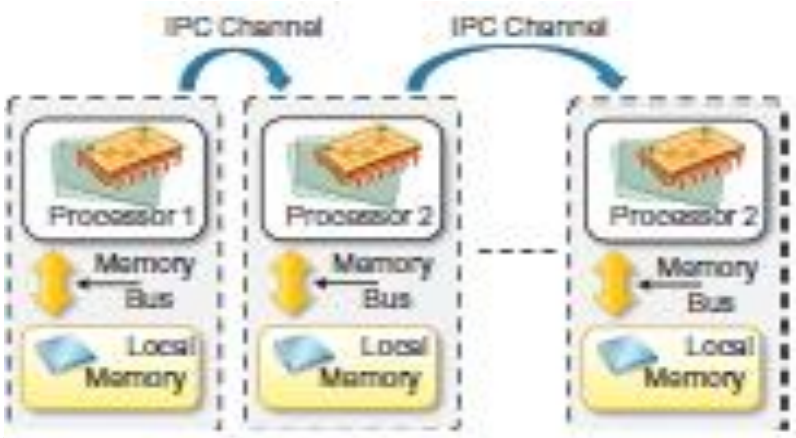
Shared Memory MIMD machines

- All the PEs are connected to a single global memory and they all have access to it.
- Systems based on this model are also called tightly coupled multi processor systems.
- The communication between PEs in this model takes place through the shared memory.
- Modification of the data stored in the global memory by one PE is visible to all other PEs.
- Dominant representative shared memory MIMD systems are silicon graphics machines and Sun/IBM SMP (Symmetric Multi-Processing).



Distributed Memory MIMD machines

- All PEs have a local memory. Systems based on this model are also called loosely coupled multi processor systems.
- The communication between PEs in this model takes place through the interconnection network, the inter process communication channel, or IPC.
- The network connecting PEs can be configured to tree, mesh, cube, and so on.
- Each PE operates asynchronously, and if communication/synchronization among tasks is necessary, they can do so by exchanging messages between them.



Shared Vs Distributed MIMD model

- The shared memory MIMD architecture is easier to program but is less tolerant to failures and harder to extend with respect to the distributed memory MIMD model.
- Failures, in a shared memory MIMD affect the entire system, whereas this is not the case of the distributed model, in which each of the PEs can be easily isolated.
- Moreover, shared memory MIMD architectures are less likely to scale because the addition of more PEs leads to memory contention.
- This is a situation that does not happen in the case of distributed memory, in which each PE has its own memory.

As a result, distributed memory MIMD architectures are most popular today

c.Approaches to Parallel Programming

- A sequential program is one that runs on a single processor and has a single line of control.
- To make many processors collectively work on a single program, the program must be divided into smaller independent chunks so that each processor can work on separate chunks of the problem.
- The program decomposed in this way is a parallel program.
- A wide variety of parallel programming approaches are available.
- The most prominent among them are the following.
- Data Parallelism
- Process Parallelism
- Farmer-and-worker model
- The above said three models are suitable for task-level parallelism. In the case of data level parallelism, the divide-and-conquer technique is used to split data into multiple sets, and each data set is processed on different PEs using the same instruction.
- This approach is highly suitable to processing on machines based on the SIMD model.
- In the case of Process Parallelism, a given operation has multiple (but distinct) activities that can be processed on multiple processors.
- In the case of Farmer-and-Worker model, a job distribution approach is used, one processor is configured as master and all other remaining PEs are designated as slaves,

the master assigns the jobs to slave PEs and, on completion, they inform the master, which in turn collects results.

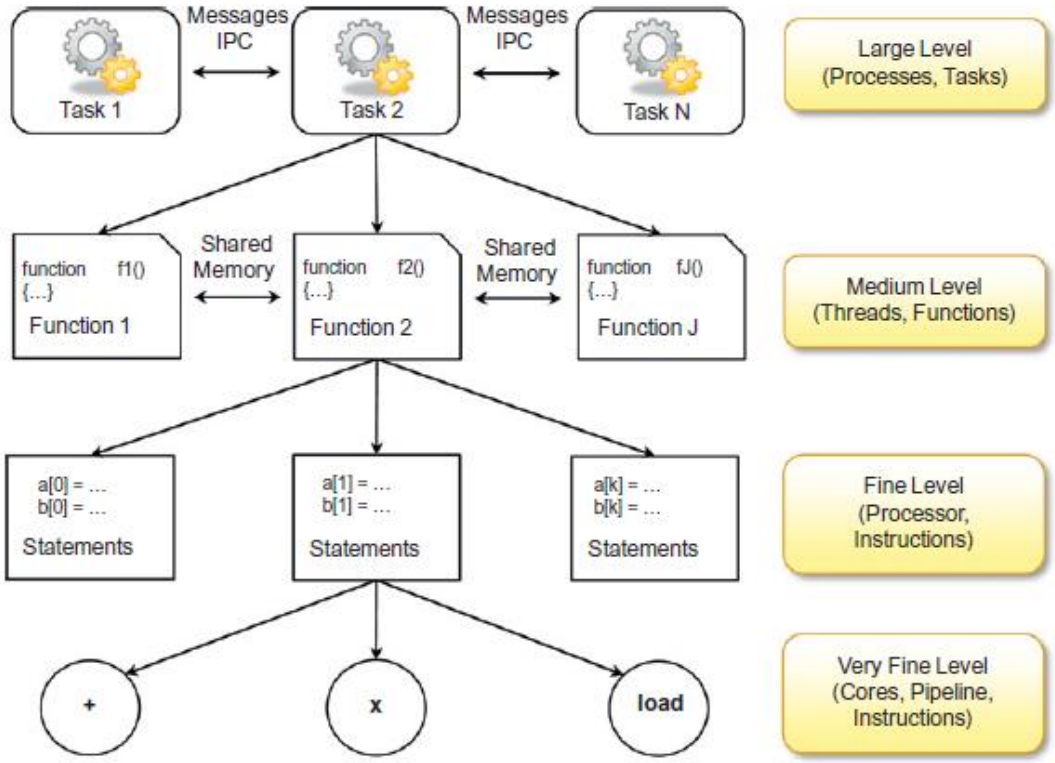
- These approaches can be utilized in different levels of parallelism.

d. Levels of Parallelism

- Levels of Parallelism are decided on the lumps of code (grain size) that can be a potential candidate of parallelism.
- The table shows the levels of parallelism.
- All these approaches have a common goal
  - To boost processor efficiency by hiding latency.
  - To conceal latency, there must be another thread ready to run whenever a lengthy operation occurs.
- The idea is to execute concurrently two or more single-threaded applications. Such as compiling, text formatting, database searching, and device simulation.

Grain Size	Code Item	Parallelized By
Large	Separate and heavy weight process	Programmer
Medium	Function or procedure	Programmer
Fine	Loop or instruction block	Parallelizing compiler
Very Fine	Instruction	Processor

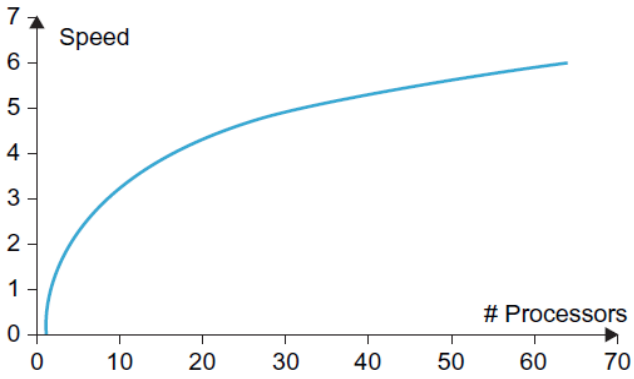
Levels of Parallelism



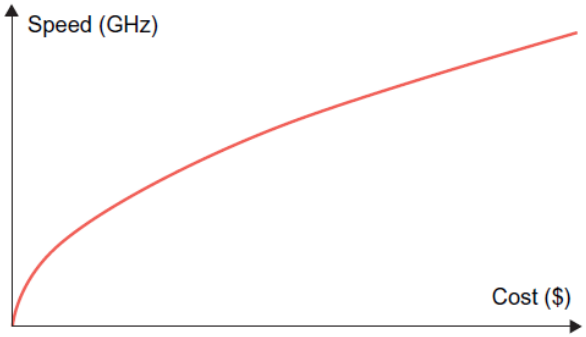
e. Laws of Caution

- Studying how much an application or a software system can gain from parallelism.
- In particular what need to keep in mind is that parallelism is used to perform multiple activities together so that the system can increase its throughput or its speed.
- But the relations that control the increment of speed are not linear.
- For example: for a given n processors, the user expects speed to be increase by in times. This is an ideal situation, but it rarely happens because of the communication overhead.
- Here two important guidelines to take into account.
  - Speed of computation is proportional to the square root of the system cost; they never increase linearly. Therefore, the faster a system becomes, the more expensive it is to increase its speed

– Speed by a parallel computer increases as the logarithm of the number of processors (i.e. y=k*log(N)).



Number processors versus speed



Cost versus speed

1.4.3 Elements of Distributed Computing

a.General concepts and definitions

- Distributed computing studies the models, architectures, and algorithms used for building and managing distributed systems.
- As general definition of the term distributed system, we use the one proposed by Tanenbaum
  - A distributed system is a collection of independent computers that appears to its users as a single coherent system.
- This definition is general enough to include various types of distributed computing systems that are especially focused on unified usage and aggregation of distributed resources.
- Communications is another fundamental aspect of distributed computing. Since distributed systems are composed of more than one computer that collaborate together, it is necessary to provide some sort of data and information exchange between them, which generally occurs through the network.
  - A distributed system is one in which components located at networked computers communicate and coordinate their action only by passing messages.
- As specified in this definition, the components of a distributed system communicate with some sort of message passing. This is a term that encompasses several communication models.

b.Components of distributed System

- A distributed system is the result of the interaction of several components that traverse the entire computing stack from hardware to software.
- It emerges from the collaboration of several elements that- by working together- give users the illusion of a single coherent system.
- The figure provides an overview of the different layers that are involved in providing the services of a distributed system.

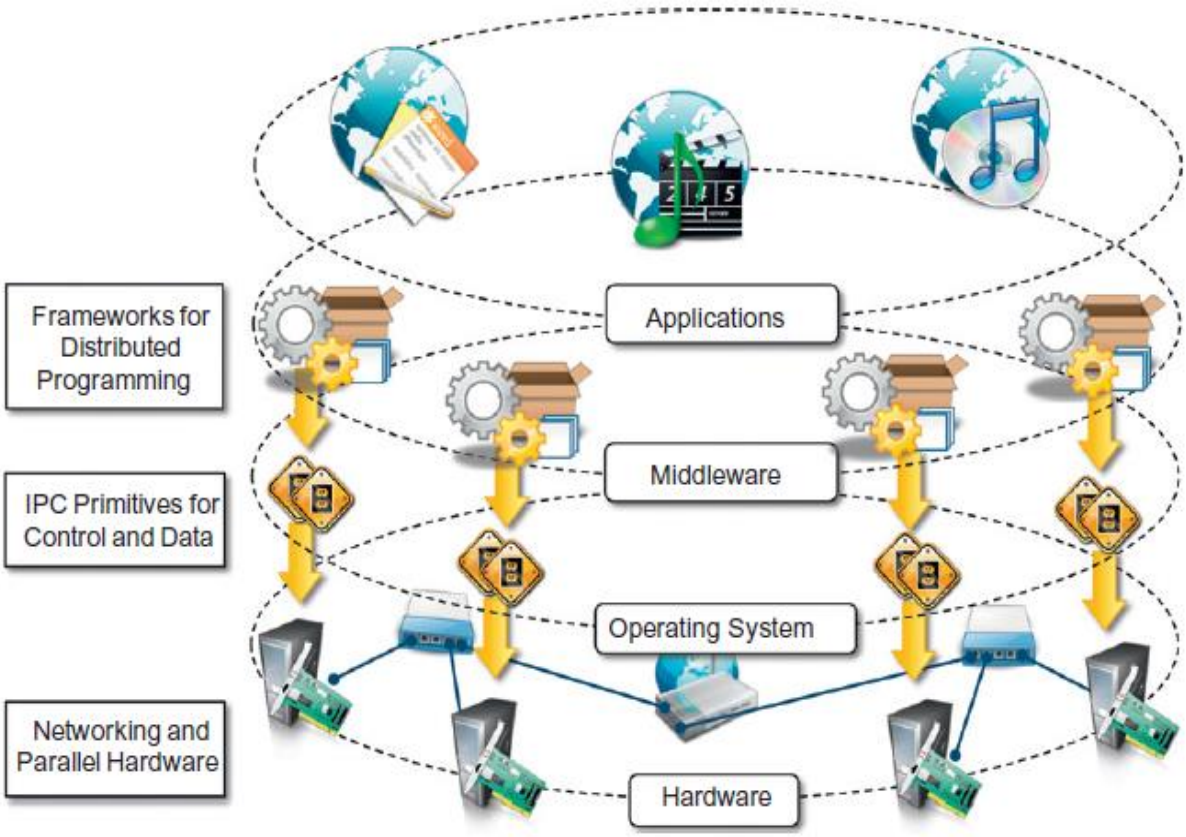


Figure 1.10 A layered view of a distributed system.

c.Architectural styles for distributed computing

- At the very bottom layer, computer and network hardware constitute the physical infrastructure; these components are directly managed by the operating system, which

provides the basic services for inter process communication (IPC), process scheduling and management, and resource management in terms of file system and local devices.

- Taken together these two layers become the platform on top of which specialized software is deployed to turn a set of networked computers into a distributed system
- Although a distributed system comprises the interaction of several layers, the middleware layer is the one that enables distributed computing, because it provides a coherent and uniform runtime environment for applications.
- There are many different ways to organize the components that, taken together, constitute such an environment.
- The interactions among these components and their responsibilities give structure to the middleware and characterize its type or, in other words, define its architecture.
- Architectural styles aid in understanding the classifying the organization of the software systems in general and distributed computing in particular.
- The use of well-known standards at the operating system level and even more at the hardware and network levels allows easy harnessing of heterogeneous components and their organization into a coherent and uniform system.
- For example; network connectivity between different devices is controlled by standards, which allow them into interact seamlessly.
- Design patterns help in creating a common knowledge within the community of software engineers and developers as to how to structure the relevant of components within an application and understand the internal organization of software applications.
- Architectural styles do the same for the overall architecture of software systems.
- The architectural styles are classified into two major classes
    - Software Architectural styles : Relates to the logical organization of the software.
    - System Architectural styles: styles that describe the physical organization of distributed software systems in terms of their major components.

Software Architectural Styles

- Software architectural styles are based on the logical arrangement of software components.
- They are helpful because they provide an intuitive view of the whole system, despite its physical deployment.

- They also identify the main abstractions that are used to shape the components of the system and the expected interaction patterns between them.

Data Centered Architectures

- These architectures **identify the data as the fundamental element of the software system,** and access to shared data is the core characteristics of the data-centered architectures.
- Within the context of distributed and parallel computing systems, **integrity of data is overall goal for such systems.**
- The **repository architectural style** is the most relevant reference model in this category. It is characterized by two main components – the central data structure, which represents the current state of the system, and a collection of independent component, which operate on the central data.
- The ways in which the independent components interact with the central data structure can be very heterogeneous.
- In particular repository based architectures differentiate and specialize further into subcategories according to the choice of control discipline to apply for the shared data structure. Of particular interest are databases and blackboard systems.

Black board Architectural Style

- The black board architectural style is characterized by three main components:
  - Knowledge sources: These are entities that update the knowledge base that is maintained in the black board.
  - Blackboard: This represents the data structure that is shared among the knowledge sources and stores the knowledge base of the application.
  - Control: The control is the collection of triggers and procedures that govern the interaction with the blackboard and update the status of the knowledge base.

Data Flow Architectures

- Access to data is the core feature; data-flow styles explicitly incorporate the pattern of data-flow, since their design is determined by an orderly motion of data from component to component, which is the form of communication between them.
- Styles within this category differ in one of the following ways: how the control is exerted, the degree of concurrency among components, and the topology that describes the flow of data.

- **Batch Sequential:** The batch sequential style is characterized by an ordered sequence of separate programs executing one after the other. These programs are chained together by providing as input for the next program the output generated by the last program after its completion, which is most likely in the form of a file. This design was very popular in the mainframe era of computing and still finds applications today. For example, many distributed applications for scientific computing are defined by jobs expressed as sequence of programs that, for example, pre-filter, analyze, and post process data. It is very common to compose these phases using the batch sequential style.
- **Pipe-and-Filter Style:** It is a variation of the previous style for expressing the activity of a software system as sequence of data transformations. Each component of the processing chain is called a filter, and the connection between one filter and the next is represented by a data stream.

Virtual Machine architectures

- The virtual machine class of architectural styles is characterized by the presence of an abstract execution environment (generally referred as a virtual machine) that simulates features that are not available in the hardware or software.
- Applications and systems are implemented on top of this layer and become portable over different hardware and software environments.
- The general interaction flow for systems implementing this pattern is – the program (or the application) defines its operations and state in an abstract format, which is interpreted by the virtual machine engine. The interpretation of a program constitutes its execution. It is quite common in this scenario that the engine maintains an internal representation of the program state.
- Popular examples within this category are rule based systems, interpreters, and command language processors.
- **Rule-Based Style:**
    - This architecture is characterized by representing the abstract execution environment as an inference engine. Programs are expressed in the form of rules or predicates that hold true. The input data for applications is generally represented by a set of assertions or facts that the inference engine uses to activate rules or to apply predicates, thus transforming data. The examples of rule-based systems can be found in the networking domain: Network Intrusion Detection

Systems (NIDS) often rely on a set of rules to identify abnormal behaviors connected to possible intrusion in computing systems.

- Interpreter Style: The presence of engine to interpret the style.

Call and return architectures

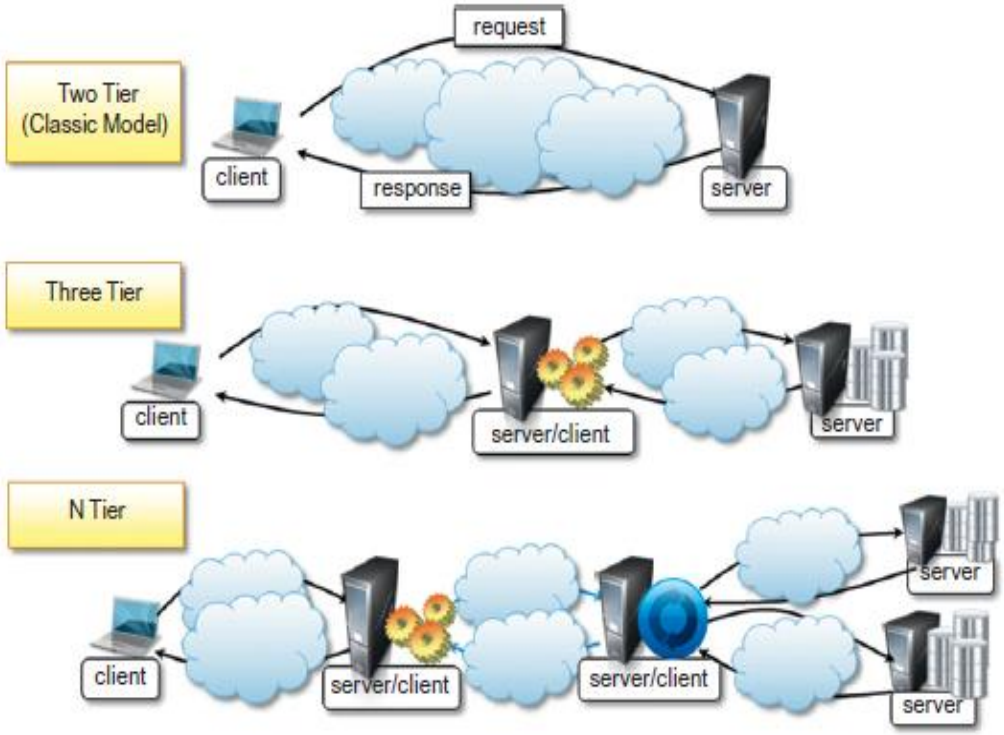
- This identifies all systems that are organized into components mostly connected together by method calls.
- The activity of systems modeled in this way is characterized by a chain of method calls whose overall execution and composition identify the execution one or more operations.
- There are three categories in this
  - Top down Style : developed with imperative programming
  - Object Oriented Style: Object programming models
  - Layered Style: provides the implementation in different levels of abstraction of the system.

System Architectural Styles

- System architectural styles cover the physical organization of components and processes over a distributed infrastructure.
- Two fundamental reference style
  - Client / Server
  - Peer- to -Peer
  - The information and the services of interest can be centralized and accessed through a single access point: the server.
  - Multiple clients are interested in such services and the server must be appropriately designed to efficiently serve requests coming from different clients.

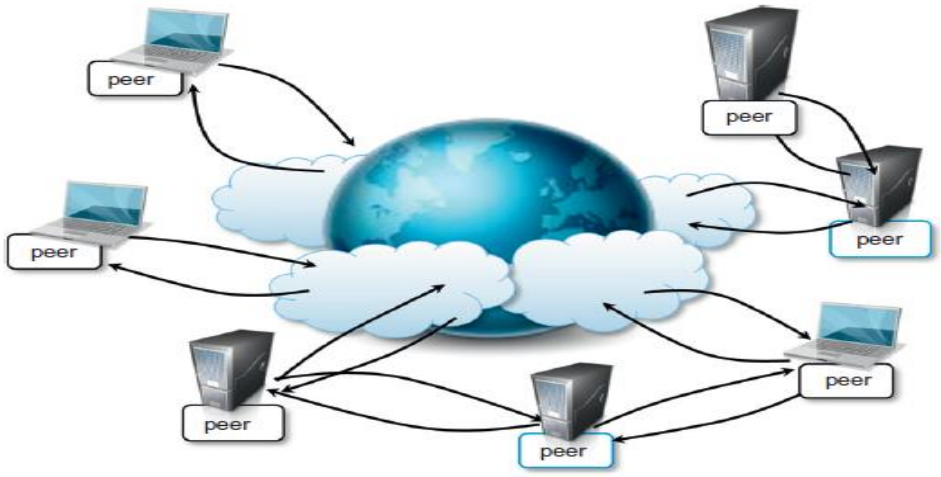
Client / Server architectural Styles

-
-
-
-
-



- Symmetric architectures in which all the components, called peers, play the same role and incorporate both client and server capabilities of the client/server model.
- More precisely, each peer acts as a server when it processes requests from other peers and as a client when it issues requests to other peers.

Peer-to-Peer architectural Style



d.Models for Inter process Communication

- Distributed systems are composed of a collection of concurrent processes interacting with each other by means of a network connection.
- IPC is a fundamental aspect of distributed systems design and implementation.
- IPC is used to either exchange data and information or coordinate the activity of processes.
- IPC is what ties together the different components of a distributed system, thus making them act as a single system.
- There are several different models in which processes can interact with each other – these maps to different abstractions for IPC.
- Among the most relevant that we can mention are shared memory, remote procedure call (RPC), and message passing.
- At lower level, IPC is realized through the fundamental tools of network programming.
- Sockets are the most popular IPC primitive for implementing communication channels between distributed processes.

Message-based communication

- The abstraction of message has played an important role in the evolution of the model and technologies enabling distributed computing.
- The definition of distributed computing – is the one in which components located at networked computers communicate and coordinate their actions only by passing messages. The term messages, in this case, identify any discrete amount of information that is passed from one entity to another. It encompasses any form of data representation that is limited in size and time, whereas this is an invocation to a remote procedure or a serialized object instance or a generic message.
- The term message-based communication model can be used to refer to any model for IPC.
- Several distributed programming paradigms eventually use message-based communication despite the abstractions that are presented to developers for programming the interactions of distributed components.
- Here are some of the most popular and important:

Message Passing: This paradigm introduces the concept of a message as the main abstraction of the model. The entities exchanging information explicitly encode in the form of a message the data to be exchanged. The structure and the content of a message vary according to the model. Examples of this model are the Message-Passing-Interface (MPI) and openMP.

- **Remote Procedure Call (RPC):** This paradigm extends the concept of procedure call beyond the boundaries of a single process, thus triggering the execution of code in remote processes.
- **Distributed Objects:** This is an implementation of the RPC model for the object-oriented paradigm and contextualizes this feature for the remote invocation of methods exposed by objects. Examples of distributed object infrastructures are Common Object Request Broker Architecture (CORBA), Component Object Model (COM, DCOM, and COM+), Java Remote Method Invocation (RMI), and .NET Remoting.
- **Distributed agents and active Objects:** Programming paradigms based on agents and active objects involve by definition the presence of instances, whether they are agents of objects, despite the existence of requests.
- **Web Service**: An implementation of the RPC concept over HTTP; thus allowing the interaction of components that are developed with different technologies. A Web service is exposed as a remote object hosted on a Web Server, and method invocation are transformed in HTTP requests, using specific protocols such as Simple Object Access Protocol (SOAP) or Representational State Transfer (REST).

e. Technologies for distributed computing

- Remote Procedure Call (RPC)
  - RPC is the fundamental abstraction enabling the execution procedures on clients' request.
  - RPC allows extending the concept of a procedure call beyond the boundaries of a process and a single memory address space.
  - The called procedure and calling procedure may be on the same system or they may be on different systems..
- Distributed Object Frameworks

- Extend object-oriented programming systems by allowing objects to be distributed across a heterogeneous network and provide facilities so that they can be coherently act as though they were in the same address space.

Web Services

- Web Services are the prominent technology for implementing SOA systems and applications.
- They leverage Internet technologies and standards for building distributed systems.
- Several aspects make Web Services the technology of choice for SOA.
- First, they allow for interoperability across different platforms and programming languages.
- Second, they are based on well-known and vendor-independent standards such as HTTP, SOAP, and WSDL.
- Third, they provide an intuitive and simple way to connect heterogeneous software systems, enabling quick composition of services in distributed environment.

1.5 ELASTICITY IN CLOUD COMPUTING

- Elasticity is defined as the ability of a system to add and remove resources (such as CPU cores, memory, VM and container instances) to adapt to the load variation in real time.
- Elasticity is a dynamic property for cloud computing.
- Elasticity is the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible.

Elasticity = Scalability + Automation + Optimization

- Elasticity is built on top of scalability.
- It can be considered as an automation of the concept of scalability and aims to optimize at best and as quickly as possible the resources at a given time.
- Another term associated with elasticity is the efficiency, which characterizes how cloud resource can be efficiently utilized as it scales up or down.
- It is the amount of resources consumed for processing a given amount of work, the lower this amount is, the higher the efficiency of a system.

- Elasticity also introduces a new important factor, which is the speed.
- Rapid provisioning and deprovisioning are key to maintaining an acceptable performance in the context of cloud computing
- Quality of service is subjected to a service level agreement

Classification

Elasticity solutions can be arranged in different classes based on

- Scope
- Policy
- Purpose
- Method

a.Scope

- Elasticity can be implemented on any of the cloud layers.
- Most commonly, elasticity is achieved on the IaaS level, where the resources to be provisioned are virtual machine instances.
- Other infrastructure services can also be scaled
- On the PaaS level, elasticity consists in scaling containers or databases for instance.
- Finally, both PaaS and IaaS elasticity can be used to implement elastic applications, be it for private use or in order to be provided as a SaaS
- The elasticity actions can be applied either at the infrastructure or application/platform level.
- The elasticity actions perform the decisions made by the elasticity strategy or management system to scale the resources.
- Google App Engine and Azure elastic pool are examples of elastic Platform as a Service (PaaS).
- Elasticity actions can be performed at the infrastructure level where the elasticity controller monitors the system and takes decisions.
- The cloud infrastructures are based on the virtualization technology, which can be VMs or containers.
- In the embedded elasticity, elastic applications are able to adjust their own resources according to runtime requirements or due to changes in the execution flow.
- There must be a knowledge of the source code of the applications.

- Application Map: The elasticity controller must have a complete map of the application components and instances.
- Code embedded: The elasticity controller is embedded in the application source code.
- The elasticity actions are performed by the application itself.
- While moving the elasticity controller to the application source code eliminates the use of monitoring systems
- There must be a specialized controller for each application.

b.Policy

- Elastic solutions can be either manual or automatic.
- A manual elastic solution would provide their users with tools to monitor their systems and add or remove resources but leaves the scaling decision to them.

Automatic mode: All the actions are done automatically, and this could be classified into reactive and proactive modes.

Elastic solutions can be either reactive or predictive

Reactive mode: The elasticity actions are triggered based on certain thresholds or rules, the system reacts to the load (workload or resource utilization) and triggers actions to adapt changes accordingly.

- An elastic solution is reactive when it scales a posteriori, based on a monitored change in the system.
- These are generally implemented by a set of Event-Condition-Action rules.

Proactive mode: This approach implements forecasting techniques, anticipates the future needs and triggers actions based on this anticipation.

- A predictive or proactive elasticity solution uses its knowledge of either recent history or load patterns inferred from longer periods of time in order to predict the upcoming load of the system and scale according to it.

c.Purpose

- An elastic solution can have many purposes.
- The first one to come to mind is naturally performance, in which case the focus should be put on their speed.
- Another purpose for elasticity can also be energy efficiency, where using the minimum amount of resources is the dominating factor.

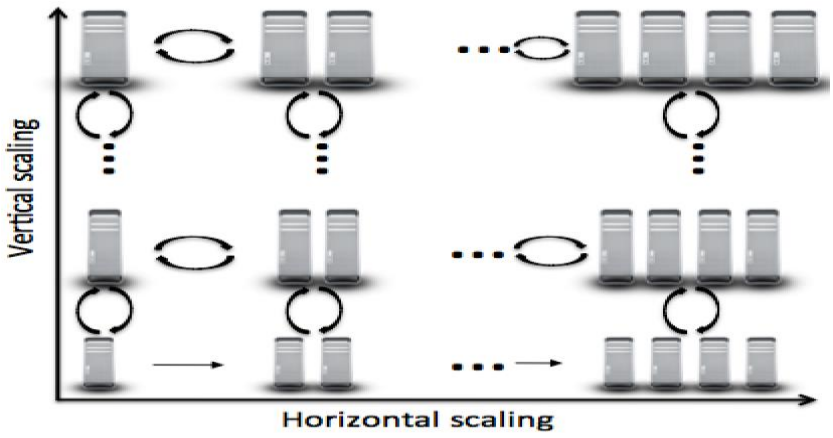
- Other solutions intend to reduce the cost by multiplexing either resource providers or elasticity methods
- Elasticity has different purposes such as improving performance, increasing resource capacity, saving energy, reducing cost and ensuring availability.
- Once we look to the elasticity objectives, there are different perspectives.
- Cloud IaaS providers try to maximize the profit by minimizing the resources while offering a good Quality of Service (QoS),
- PaaS providers seek to minimize the cost they pay to the

Cloud.

- The customers (end-users) search to increase their Quality of Experience (QoE) and to minimize their payments.
- QoE is the degree of delight or annoyance of the user of an application or service

d.Method

- Vertical elasticity, changes the amount of resources linked to existing instances on-the-fly.
- This can be done in two manners.
- The first method consists in explicitly redimensioning a virtual machine instance, i.e., changing the quota of physical resources allocated to it.
- This is however poorly supported by common operating systems as they fail to take into account changes in CPU or memory without rebooting, thus resulting in service interruption.
- The second vertical scaling method involves VM migration: moving a virtual machine instance to another physical machine with a different overall load changes its available resources



- Horizontal scaling is the process of adding/removing instances, which may be located at different locations.
- Load balancers are used to distribute the load among the different instances.
- Vertical scaling **is** the process of modifying resources (CPU, memory, storage or both) size for an instance at run time.
- It gives more flexibility for the cloud systems to cope with the varying workloads

Migration

- Migration can be also considered as a needed action to further allow the vertical scaling when there is no enough resources on the host machine.
- It is also used for other purposes such as migrating a VM to a less loaded physical machine just to guarantee its performance.
- Several types of migration are deployed such as live migration and no-live migration.
- Live migration has two main approaches
    - post-copy
    - pre-copy
- Post-copy migration suspends the migrating VM, copies minimal processor state to the target host, resumes the VM and then begins fetching memory pages from the source.
- In pre-copy approach, the memory pages are copied while the VM is running on the source.
- If some pages are changed (called dirty pages) during the memory copy process, they will be recopied until the number of recopied pages is greater than dirty pages, or the source VM will be stopped.
- The remaining dirty pages will be copied to the destination VM.

Architecture

- The architecture of the elasticity management solutions can be either centralized or decentralized.
- Centralized architecture has only one elasticity controller, i.e., the auto scaling system that provisions and deprovisions resources.

- In decentralized solutions, the architecture is composed of many elasticity controllers or application managers, which are responsible for provisioning resources for different cloud-hosted platforms

Provider

- Elastic solutions can be applied to a single or multiple cloud providers.
- A single cloud provider can be either public or private with one or multiple regions or datacenters.
- Multiple clouds in this context means more than one cloud provider.
- It includes hybrid clouds that can be private or public, in addition to the federated clouds and cloud bursting.
- Most of the elasticity solutions support only a single cloud provider

1.6 On-demand Provisioning.

- Resource Provisioning means the selection, deployment, and run-time management of software (e.g., database server management systems, load balancers) and hardware resources (e.g., CPU, storage, and network) for ensuring guaranteed performance for applications.
- Resource Provisioning is an important and challenging problem in the large-scale distributed systems such as Cloud computing environments.
- There are many resource provisioning techniques, both static and dynamic each one having its own advantages and also some challenges.
- These resource provisioning techniques used must meet Quality of Service (QoS) parameters like availability, throughput, response time, security, reliability etc., and thereby avoiding Service Level Agreement (SLA) violation.
- Over provisioning and under provisioning of resources must be avoided.
- Another important constraint is power consumption.
- The ultimate goal of the cloud user is to minimize cost by renting the resources and from the cloud service provider's perspective to maximize profit by efficiently allocating the resources.

- In order to achieve the goal, the cloud user has to request cloud service provider to make a provision for the resources either statically or dynamically.
- So that the cloud service provider will know how many instances of the resources and what resources are required for a particular application.
- By provisioning the resources, the QoS parameters like availability, throughput, security, response time, reliability, performance etc must be achieved without violating SLA

There are two types

- **Static Provisioning**
- **Dynamic Provisioning**

Static Provisioning

- For applications that have predictable and generally unchanging demands/workloads, it is possible to use “static provisioning" effectively.
- With advance provisioning, the customer contracts with the provider for services.
- The provider prepares the appropriate resources in advance of start of service.
- The customer is charged a flat fee or is billed on a monthly basis.

Dynamic Provisioning

- In cases where demand by applications may change or vary, “dynamic provisioning" techniques have been suggested whereby VMs may be migrated on-the-fly to new compute nodes within the cloud.
- The provider allocates more resources as they are needed and removes them when they are not.
- The customer is billed on a pay-per-use basis.
- When dynamic provisioning is used to create a hybrid cloud, it is sometimes referred to as cloud bursting.

Parameters for Resource Provisioning

- Response time
- Minimize Cost
- Revenue Maximization
- Fault tolerant
- Reduced SLA Violation
- Reduced Power Consumption

Response time: The resource provisioning algorithm designed must take minimal time to respond when executing the task.

Minimize Cost: From the Cloud user point of view cost should be minimized.

Revenue Maximization: This is to be achieved from the Cloud Service Provider's view.

Fault tolerant: The algorithm should continue to provide service in spite of failure of nodes.

Reduced SLA Violation: The algorithm designed must be able to reduce SLA violation.

Reduced Power Consumption: VM placement & migration techniques must lower power consumption

Dynamic Provisioning Types

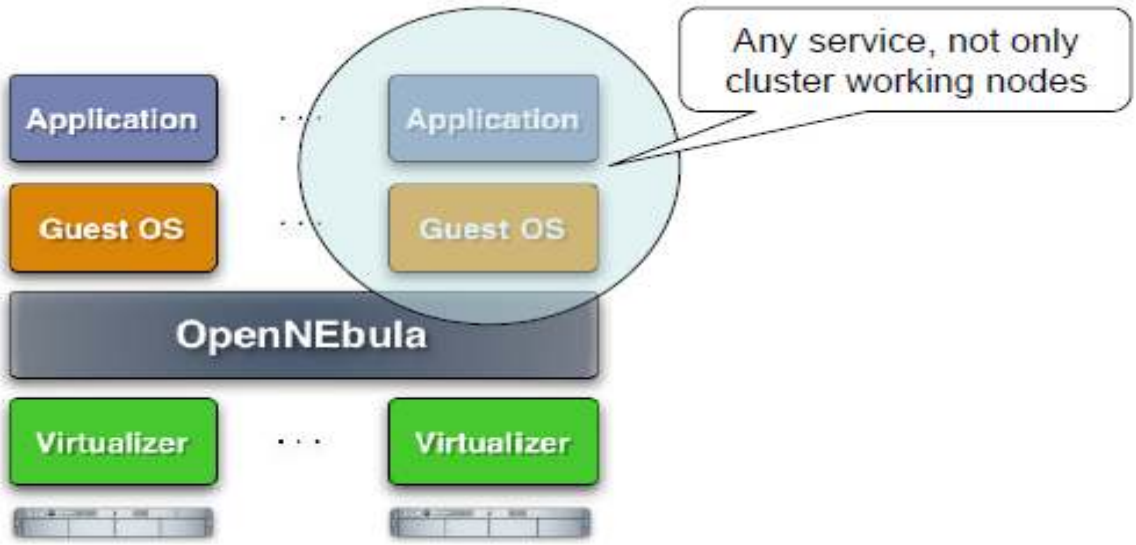
1. Local On-demand Resource Provisioning
2. Remote On-demand Resource Provisioning

Local On-demand Resource Provisioning

1. The Engine for the Virtual Infrastructure

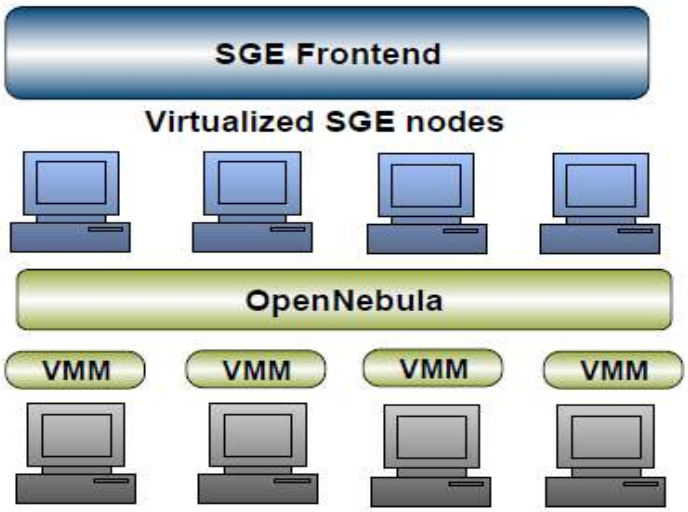
The OpenNebula Virtual Infrastructure Engine

- OpenNEbula creates a distributed virtualization layer
  - Extend the benefits of VM Monitors from one to multiple resources
  - Decouple the VM (service) from the physical location
- Transform a distributed physical infrastructure into a flexible and elastic virtual infrastructure, which adapts to the changing demands of the VM (service) workloads



Separation of Resource Provisioning from Job Management

- New virtualization layer between the service and the infrastructure layers
- Seamless integration with the existing middleware stacks.
- Completely transparent to the computing service and so end users



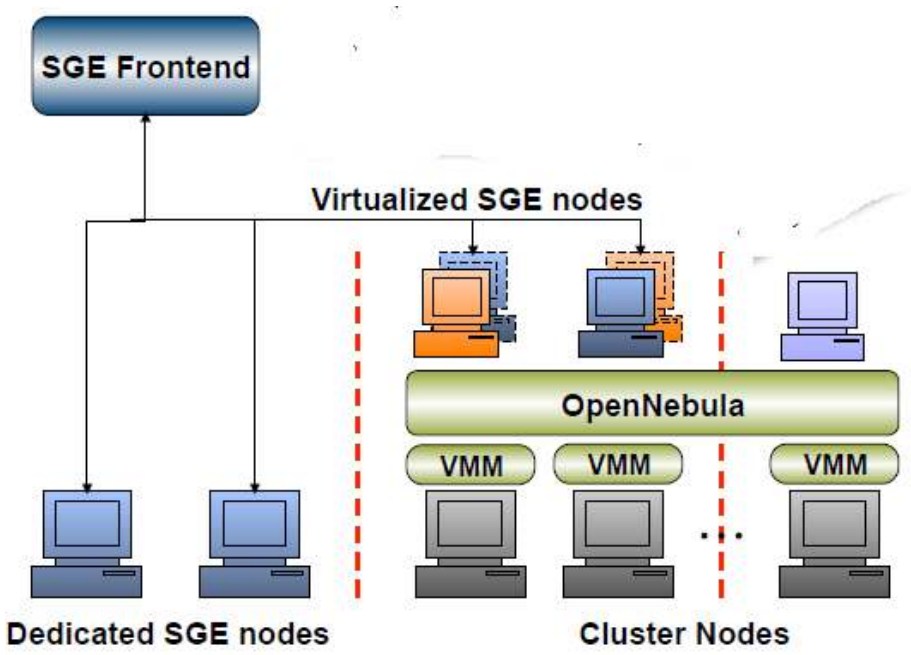
Cluster Partitioning

- Dynamic partition of the infrastructure
- Isolate workloads (several computing clusters)
- Dedicated HA partitions

Benefits for Existing Grid Infrastructures

- The **virtualization of the local infrastructure** supports a virtualized alternative to contribute resources to a Grid infrastructure
    - Simpler deployment and operation of new middleware distributions
    - Lower operational costs
    - Easy provision of resources to more than one infrastructure
    - Easy support for VO-specific worker nodes

  Performance partitioning between local and grid clusters



•

Other Tools for VM Management

- VMware DRS, Platform Orchestrator, IBM Director, Novell ZENworks, Enomalism, Xenoserver
- **Advantages**:
  - Open-source (Apache license v2.0)
  - Open and flexible architecture to integrate new virtualization technologies
  - Support for the definition of any scheduling policy (consolidation, workload balance, affinity, SLA)
  - LRM-like CLI and API for the integration of third-party tools

Remote on-Demand Resource Provisioning

Access to Cloud Systems

- Provision of virtualized resources as a service

VM Management Interfaces

The processes involved are

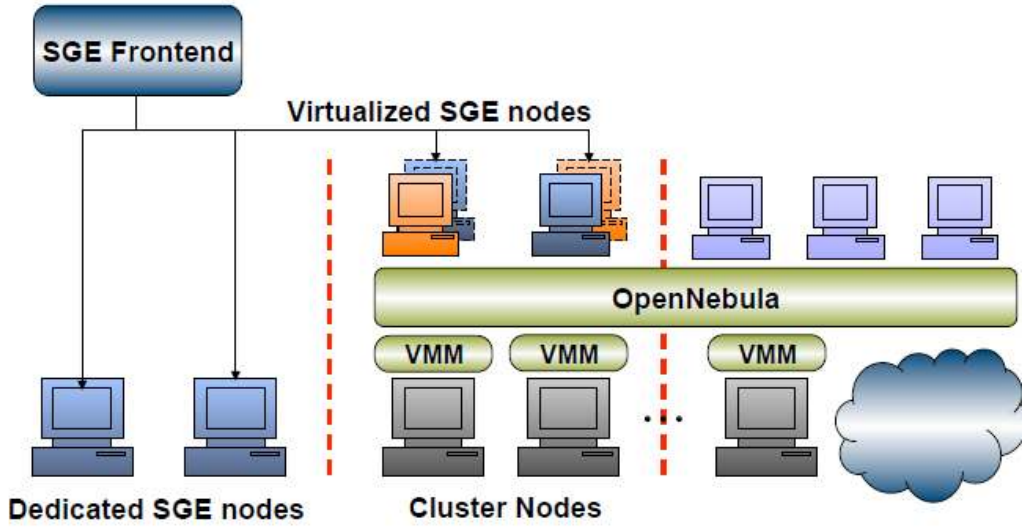
- Submission
- Control
- Monitoring

Infrastructure Cloud Computing Solutions

- Commercial Cloud: Amazon EC2
- Scientific Cloud: Nimbus (University of Chicago)
- Open-source Technologies
    - Globus VWS (Globus interfaces)
    - Eucalyptus (Interfaces compatible with Amazon EC2)
    - OpenNEbula (Engine for the Virtual Infrastructure)

On-demand Access to Cloud Resources

- Supplement local resources with cloud resources to satisfy peak or fluctuating demands



UNIT II CLOUD ENABLING TECHNOLOGIES

Service Oriented Architecture – REST and Systems of Systems – Web Services – Publish-Subscribe Model – Basics of Virtualization – Types of Virtualization – Implementation Levels of Virtualization – Virtualization Structures – Tools and Mechanisms – Virtualization of CPU – Memory – I/O Devices –Virtualization Support and Disaster Recovery.

2.1 Introduction

Web Service

- **Generic definition**

- Any application accessible to other applications over the Web.

- **Definition of the UDDI consortium**

- Web services are self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces.

- **Definition of the World Wide Web Consortium (W3C)**

- A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.
- It has an interface described in a machine-processable format (specifically WSDL).
- Other systems interact with the Web service using SOAP messages.

What is a Web Service?

- Web Services are Classes/Methods, NOT Servlets
- Loosely-coupled
- Encapsulate functionality to logical entities
- Reuse of code and functionality
- Distributed Architecture
- Standardized interface & established Internet protocols

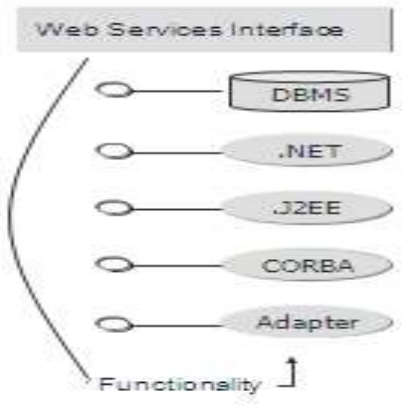


Figure 2.1 Service Interfaces

Characteristics of a Web Service

- A web service interface generally consists of a collection of operations that can be used by a client over the Internet.
- The operations in a web service may be provided by a variety of different resources, for example, programs, objects, or databases.
- The key characteristic of (most) web services is that they can process XML-formatted SOAP messages. An alternative is the REST approach.
- Each web service uses its own service description to deal with the service-specific characteristics of the messages it receives. Commercial examples include Amazon, Yahoo, Google and eBay.

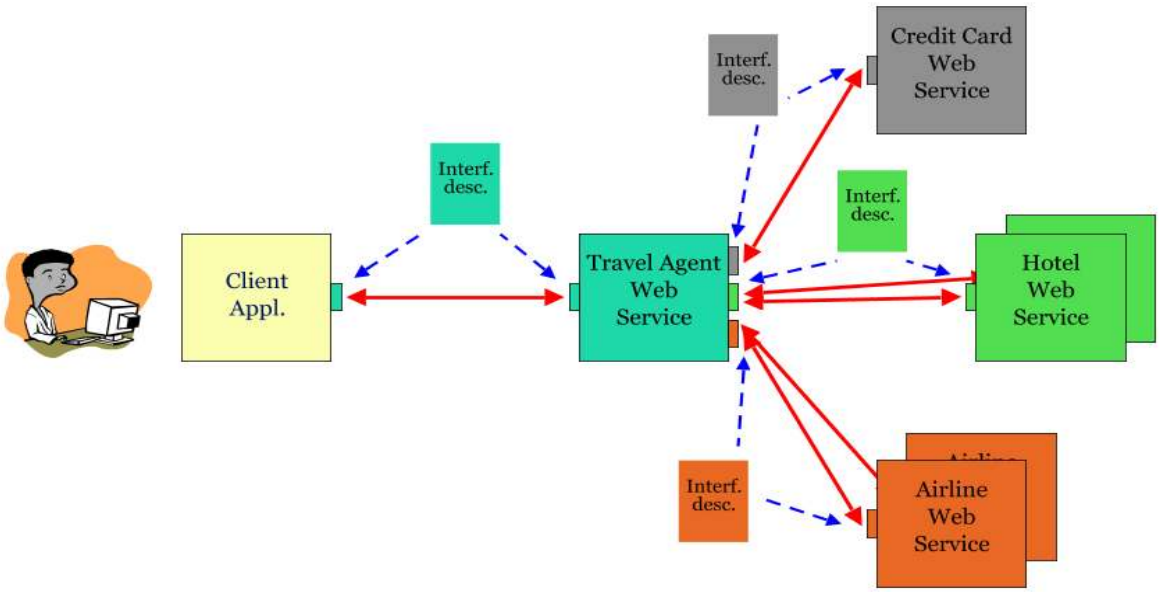


Figure 2.2 Example-Travel Agent Service

Remote Access

- IP Address - Locate a Computer
- URI - Uniform Resource Identifier
    - Locate a file in that Computer
- Socket and Port- Binding to a Method

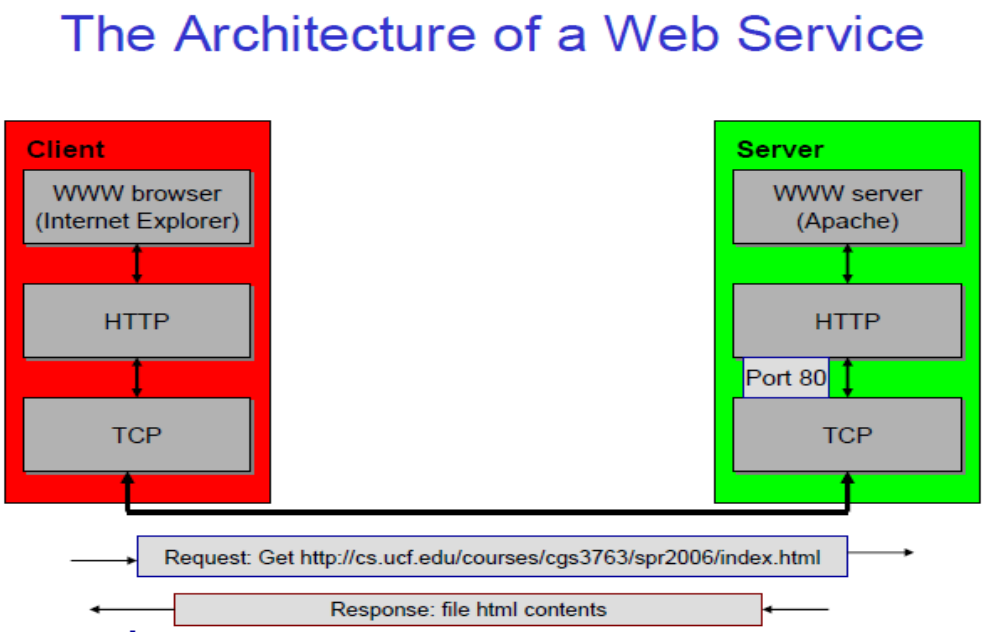
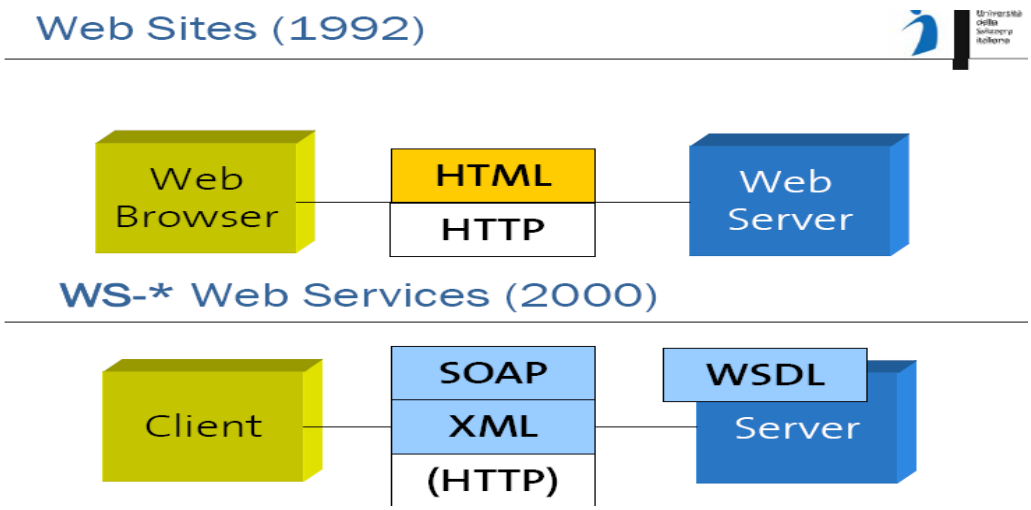


Figure 2.3 Web Service Architecture



2.1.1 SOA – Service Oriented Architecture

- Service provider publishes service description (WSDL), e.g. on a service broker
- Service Requester finds service (on service broker) and dynamically binds to service
- Enables ad-hoc collaboration and Enterprise Application Integration (EAI) within web-based information systems
- SOA is about how to design a software system that makes use of services of new or legacy applications through their published or discoverable interfaces.

- These applications are often distributed over the networks.
- SOA also aims to make service interoperability extensible and effective.
- It prompts architecture styles such as loose coupling, published interfaces and a standard communication model in order to support this goal.

Properties of SOA

- Logical view
- Message orientation
- Description orientation

Logical view

- The SOA is an abstracted, logical view of actual programs, databases, business processes.
- Defined in terms of what it does, typically carrying out a business-level operation.
- The service is formally defined in terms of the messages exchanged between provider agents and requester agents.

Message Orientation

- The internal structure of providers and requesters include the implementation language, process structure, and even database structure.
- These features are deliberately abstracted away in the SOA
- Using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed.
- The key benefit of this concerns legacy systems.
- By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application to adhere to the formal service definition.

Description orientation

- A service is described by machine-executable metadata.
- The description supports the public nature of the SOA.
- Only those details that are exposed to the public and are important for the use of the service should be included in the description.

- The semantics of a service should be documented, either directly or indirectly, by its description.

SOA Realization (Two ways)

- XML - SOAP Based Web Services
- Extensible Markup Language (XML) is a markup language designed as a standard way to encode documents and data
- SOAP is an acronym for Simple Object Access Protocol. It is an XML-based messaging protocol for exchanging information among computers
- RESTful Web services

Web service is the terminology used everywhere

Service Oriented Architecture model implemented by XML Web Services

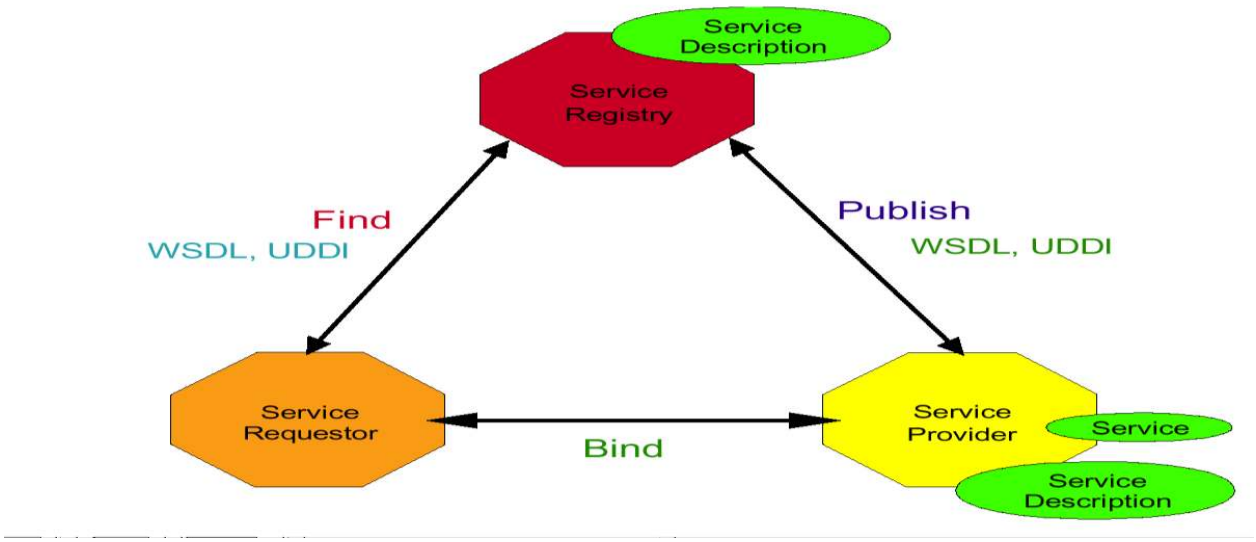


Figure 2.4 SOA Model

WSDL – Web Services Description Languages

- Provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.
- Used in combination with SOAP and an XML Schema to provide Web services over the Internet.

UDDI – Universal Description, Discovery and Integration

- White Pages — address, contact, and known identifiers;

- Yellow Pages — industrial categorizations based on standard taxonomies;
- Green Pages — technical information about services exposed by the business.

2.2 Representational State Transfer (REST)

- APIs are for software components; a way for software to interact with other software.
- Web Services are a set of rules and technologies that enable two or more components on the web to talk to each other.
- Not every API is a web service.
- REST API is a web service.
- REST API is an API that follows the rules of REST specification.
- A web service is defined by rules:

a) How software components will talk?

b) What kind of messages they will send to each other?

c) How requests and responses will be handled?

HTTP and REST

- HTTP is an application layer protocol for sending and receiving messages over a network.
- REST is a specification that dictates how distributed systems on the web should communicate.
- REST is a way to implement and use the HTTP protocol.

REST and Systems of Systems

- SOA focuses on loosely coupled software applications running across different administrative domains, based on common protocols and technologies, such as HTTP and XML.
- SOA is related to early efforts on the architecture style of large scale distributed systems, particularly Representational State Transfer (REST).
- REST still provides an alternative to the complex standard-driven web services technology.
- Used in many Web 2.0 services.

- REST is a software architecture style for distributed systems, particularly distributed hypermedia systems, such as the World Wide Web.

Applications:

Google, Amazon, Yahoo, Facebook and Twitter

Advantage:

- Simplicity
- Ease of being published and consumed by clients.

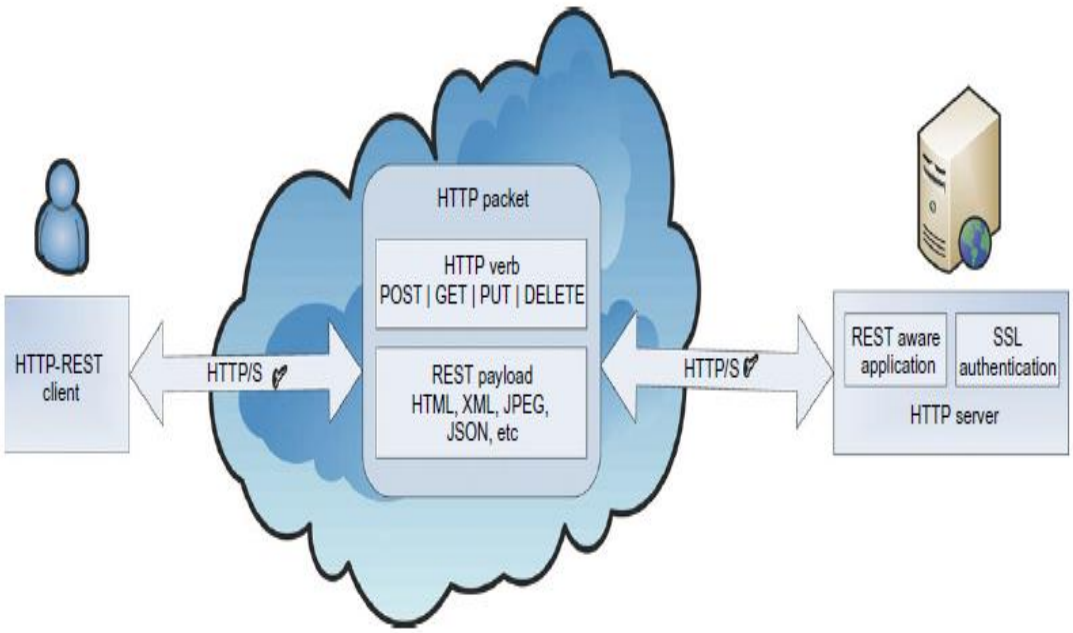


Figure 2.5 A simple REST interaction between user and server in HTTP specification

REST Principles

The REST architectural style is based on four principles:

- **Resource Identification through URIs**
- **Uniform, Constrained Interface**
- **Self-Descriptive Message**
- **Stateless Interactions**

Resource Identification through URIs

- The RESTful web service exposes a set of resources which identify targets of interaction with its clients.
- The key abstraction of information in REST is a resource.
- Any information that can be named can be a resource, such as a document or image or a temporal service.
- A resource is a conceptual mapping to a set of entities.
- Each particular resource is identified by a unique name, or more precisely, a Uniform Resource Identifier (URI).
- URI is of type URL, providing a global addressing space for resources involved in an interaction between components as well as facilitating service discovery.
- The URIs can be bookmarked or exchanged via hyperlinks.
- URIs provide more readability and the potential for advertisement.

Uniform, Constrained Interface

- Interaction with RESTful web services is done via the HTTP standard, client/server cacheable protocol.
- Resources are manipulated using a fixed set of four CRUD

  (create, read, update, delete) verbs or operations:

  - PUT
  - GET
  - POST
  - DELETE

- **PUT** creates a new resource.
- The resource can then be destroyed by using **DELETE.**
- **GET retrieves the current state of a resource.**
- **POST** transfers a new state onto a resource**.**

Self-Descriptive Message

- **A** REST message includes enough information to describe how to process the message.
- This enables intermediaries to do more with the message without parsing the message contents.
- In REST, resources are decoupled from their representation so that their content can be accessed in a variety of standard formats

  Eg:- HTML, XML, MIME, plain text, PDF, JPEG, JSON, etc.
- REST provides multiple/alternate representations of each resource.
- Metadata about the resource is available and can be used for various purposes.
  - ❖ Cache control
  - ❖ Transmission error detection
  - ❖ Authentication or authorization
  - ❖ Access control.

Stateless Interactions

- The REST interactions are "stateless"
- Message does not depend on the state of the conversation.
- Stateless communications improve visibility, reliability and increases scalability
- Decrease network performance by increasing the repetitive data

REST - Advantages

- RESTful web services can be considered an alternative to SOAP stack or "big web services
- Simplicity
- Lightweight nature
- Integration with HTTP

REST Architectural Elements

REST Elements	Elements	Example
Data elements	Resource	The intended conceptual target of a hypertext reference
	Resource identifier	URL
	Representation	HTML document, JPEG image, XML, etc.
	Representation metadata	Media type, last-modified time
	Resource metadata	Source link, alternates, vary
	Control data	If-modified-since, cache-control
Connectors	Client	libwww, libwww-perl
	Server	libwww, Apache API, NSAPI
	Cache	Browser cache, Akamai cache network
	Resolver	Bind (DNS lookup library)
	Tunnel	SSL after HTTP CONNECT
Components	Origin server	Apache httpd, Microsoft IIS
	Gateway	Squid, CGI, Reverse Proxy
	Proxy	CERN Proxy, Netscape Proxy, Gauntlet
	User agent	Netscape Navigator, Lynx, MOMspider

2.3 Web Services

- Web service is often referred to a self-contained, self-describing, modular application designed to be used and accessible by other software applications across the web.
- This allows client software to dynamically determine what a service does, the data types that a service uses, how to invoke operations on the service, and the responses that the service may return.
- Once a web service is deployed, other applications and other web services can discover and invoke the deployed service.
- A web service is defined as a software system designed to support interoperable machine-to-machine interaction over a network
- A web service has an interface described in a machine-executable format (specifically Web Services Description Language or WSDL).
- Web services are remotely executed, they do not depend on resources residing on the client system that calls them.

- Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization

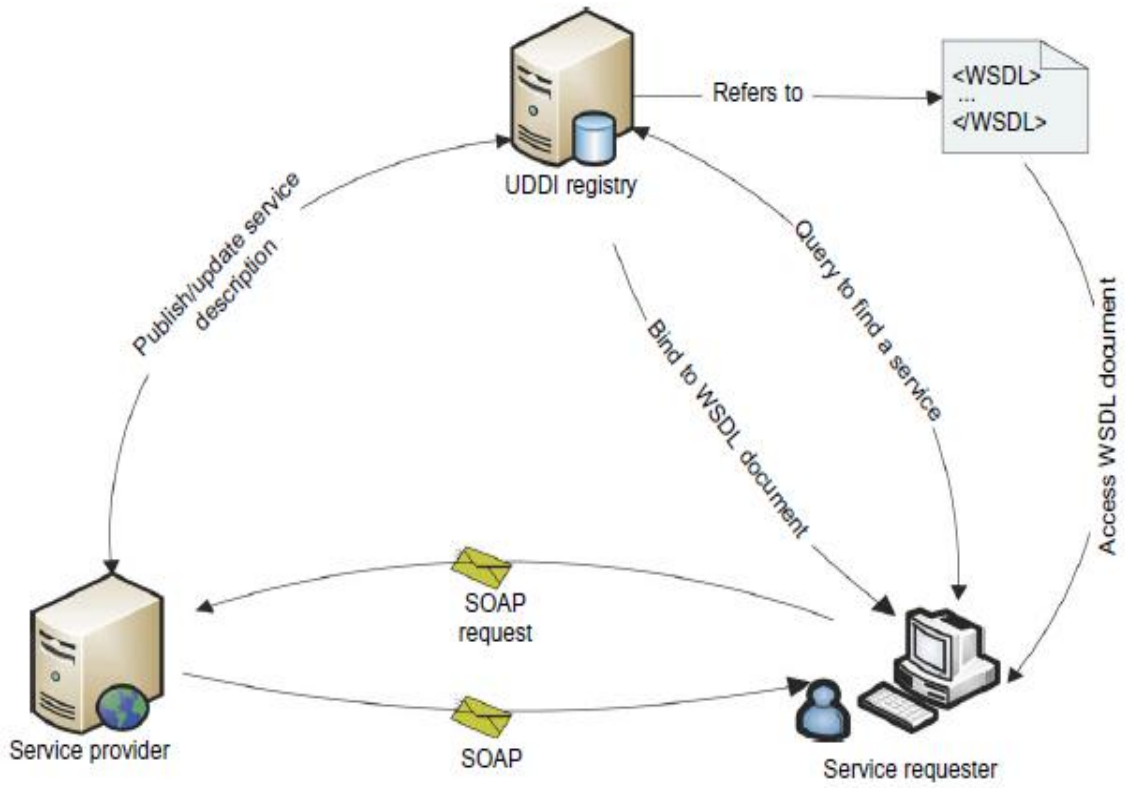


Figure 2.6 Web Services

Web Services- Technologies

- Simple Object Access Protocol (SOAP)
- Web Services Description Language (WSDL)
- Universal Description, Discovery and Integration (UDDI)

Simple Object Access Protocol (SOAP)

- SOAP provides a standard packaging structure for transmission of XML documents over various Internet protocols, such as SMTP, HTTP, and FTP.
- A SOAP message consists of a root element called envelope. Envelope contains a header: a container that can be extended by intermediaries.

- Additional application-level elements such as routing information, authentication, transaction management, message parsing instructions and Quality of Service (QoS) configurations are also included.
- Body element that carries the payload of the message.
- The content of the payload will be marshaled by the sender's SOAP engine and unmarshaled at the receiver side, based on the XML schema that describes the structure of the SOAP message.

Web Services Description Language (WSDL)

WSDL describes the interface, a set of operations supported by a web service in a standard format.

- It standardizes the representation of input and output parameters of its operations as well as the service's protocol binding, the way in which the messages will be transferred on the wire.
- Using WSDL enables disparate clients to automatically understand how to interact with a web service.

Universal Description, Discovery, and Integration (UDDI)

- UDDI provides a global registry for advertising and discovery of web services.
- Performed by searching for names, identifiers, categories or the specification implemented by the web service

SOAP is an extension, and an evolved version of XML-RPC.

A simple and effective remote procedure call protocol which uses XML for encoding its calls and HTTP as a transport mechanism.

- A procedure executed on the server and the value it returns was formatted in XML.
- SOAP mainly describes the protocols between interacting parties
- Data format of exchanging messages is left to XML schema to be handled.

Figure 2.7 WS-I Protocol Stack

- **Business Process Execution Language for Web Services (BPEL4WS**), a standard executable language for specifying interactions between web services.
  - Web services can be composed together to make more complex web services and workflows.
  - BPEL4WS is an XML-based language, built on top of web service specifications, which is used to define and manage long-lived service orchestrations or processes.
  - In BPEL, a business process is a large-grained stateful service, which executes steps to complete a business goal.
  - That goal can be the completion of a business transaction, or fulfillment of the job of a service.
- **Web Service WS-Notification** enables web services to use the publish and subscribe messaging pattern.
- **Web Services Security (WS-Security) are** set of protocols that ensure security for SOAP-based messages by implementing the principles of confidentiality, integrity and authentication.

- **Web Services Reliable Messaging (WS-Reliable Messaging)** describes a protocol that allows **messages** to be delivered reliably between distributed applications in the presence of software component, system, or network failures
- **WS-ResourceLifetime** specification standardizes the means by which a WS-Resource can be destroyed.
- **WS-Policy** is a specification that allows web services to use XML to advertise their **policies** (on security, quality of service, etc.) and for web service consumers to specify their **policy** requirements.
- **WS-ResourceProperties** defines a standard set of message exchanges that allow a requestor to query or update the property values of the WS-Resource.
- **WS-Addressing** is a specification of transport-neutral mechanism that allows web services to communicate addressing information.
- **WS-Transaction WS-Transaction** is a specification developed that indicates how transactions will be handled and controlled in Web Services.
- The transaction specification is divided into two parts - short atomic transactions (AT) and long business activity (BA).
- **Web Services Coordination (WS-Coordination)** describes an extensible framework for providing protocols that coordinate the actions of distributed applications.
- **The Java Message Service (JMS) API** is a messaging standard that allows application components based on the Java Platform Enterprise Edition (Java EE) to create, send, receive, and read messages.
- **Internet Inter-ORB Protocol (IIOP)** is an object-oriented protocol
- Used to facilitate network interaction between distributed programs written in different programming languages.
- IIOP is used to enhance Internet and intranet communication for applications and services.

Table 5.3 Sample SOAP Request-Response for Creating an S3 Bucket

SOAP Request	SOAP Response
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" soap:encodingStyle= "http://www.w3.org/2001/12/soap-encoding"> <soap:Body> <CreateBucket xmlns="http://doc.s3.amazonaws .com/2010-03-15"> <Bucket>SampleBucket</Bucket> <AWSAccessKeyId> 1B9FVRAYCP1VJEXAMPLE= </AWSAccessKeyId> <Timestamp>2010-03-15T14:40:00.165Z </Timestamp> <Signature>luyz3d3P0aTou39dzbqaEXAMPLE =</Signature> </CreateBucket> </soap:Body> </soap:Envelope>	<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" soap:encodingStyle= "http://www.w3.org/2001/12/soap-encoding"> <soap:Body> <CreateBucket xmlns="http://doc.s3.amazonaws .com/2010-03-15"> <Bucket>SampleBucket</Bucket> <AWSAccessKeyId>1B9FVRAYCP1VJEXAMPLE= </AWSAccessKeyId> <Timestamp>2010-03-15T14:40:00.165Z </Timestamp> <Signature>luyz3d3P0aTou39dzbqaEXAMPLE =</Signature> </CreateBucket> </soap:Body> </soap:Envelope>

- A SOAP message consists of an envelope used by the applications to enclose information that need to be sent.
- An envelope contains a header and a body block.
- The EncodingStyle element refers to the URI address of an XML schema for encoding elements of the message.
- Each element of a SOAP message may have a different encoding, but unless specified, the encoding of the whole message is as defined in the XML schema of the root element.
- The header is an optional part of a SOAP message that may contain auxiliary information.
- The body of a SOAP request-response message contains the main information of the conversation, formatted in one or more XML blocks.
- In example, the client is calling CreateBucket of the Amazon S3 web service interface.
- In case of an error in service invocation, a SOAP message including a Fault element in the body.

Table 5.4 The 10 Areas Covered by the Core WS-* Specifications

WS-* Specification Area	Examples
1. Core Service Model	XML, WSDL, SOAP
2. Service Internet	WS-Addressing, WS-MessageDelivery, Reliable WSRM, Efficient MOTM
3. Notification	WS-Notification, WS-Eventing (Publish-Subscribe)
4. Workflow and Transactions	BPEL, WS-Choreography, WS-Coordination
5. Security	WS-Security, WS-Trust, WS-Federation, SAML, WS-SecureConversation
6. Service Discovery	UDDI, WS-Discovery
7. System Metadata and State	WSRF, WS-MetadataExchange, WS-Context
8. Management	WSDM, WS-Management, WS-Transfer
9. Policy and Agreements	WS-Policy, WS-Agreement
10. Portals and User Interfaces	WSRP (Remote Portlets)

2.4 PUBLISH SUBSCRIBE MODEL

- Publish/Subscribe systems are nowadays considered a key technology for information diffusion.
- Each participant in a publish/subscribe communication system can play the role of a publisher or a subscriber of information.
- Publishers produce information in form of events, which are then consumed by subscribers.
- Subscribers can declare their interest on a subset of the whole information issuing subscriptions.
- There are two major roles:
  - Publisher
  - Subscriber
- The former provides facilities for the later to register its interest in a specific topic or event.

- Specific conditions holding true on the publisher side can trigger the creation of messages that are attached to a specific event.
- Message will be available to all the subscribers that registered for the corresponding event.
- There are two major strategies for dispatching the event to the subscribers.

Push strategy:

- It is the responsibility of the publisher to notify all the subscribers. Eg: Method invocation.

Pull strategy :

- The publisher simply makes available the message for a specific event.
- It is the responsibility of the subscribers to check whether there are messages on the events that are registered.
- Subscriptions are used to filter out part of the events produced by publishers.
- In Software Architecture, Publish/Subscribe pattern is a message pattern and a network oriented architectural pattern
- It describes how two different parts of a message passing system connect and communicate with each other.
- There are three main components to the Publish Subscribe Model:
    - Publishers
    - Eventbus/broker
    - Subscribers

Publishers:

- Broadcast messages, with no knowledge of the subscribers.

Subscribers:

- They 'listen' out for messages regarding topic/categories that they are interested in without any knowledge of who the publishers are.

Event Bus:

- Transfers the messages from the publishers to the subscribers.

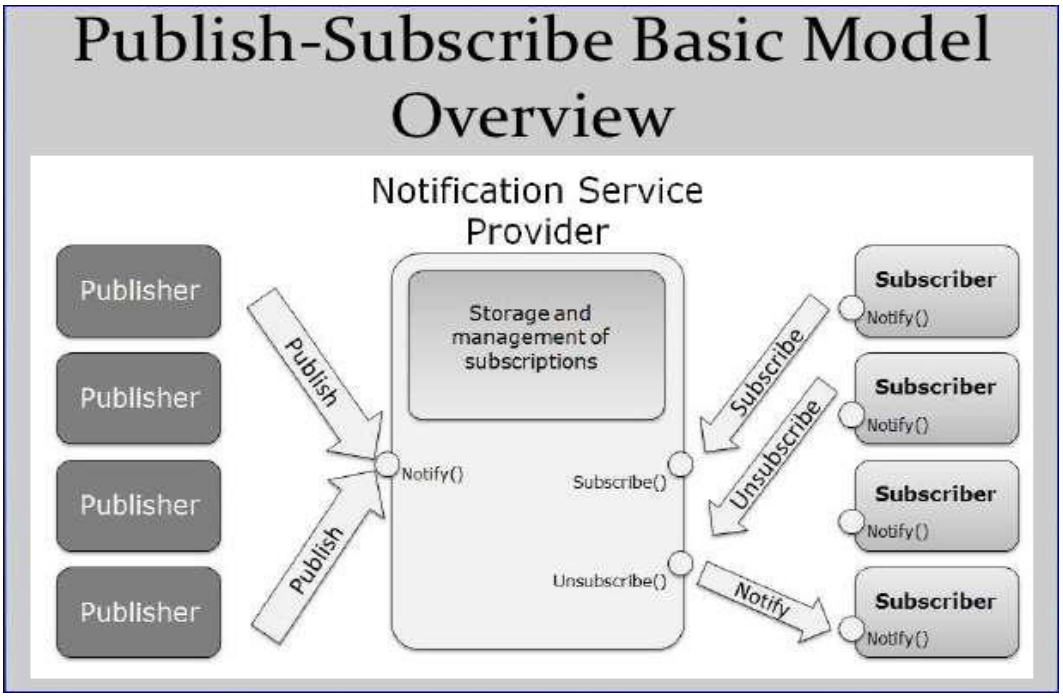


Figure 2.8 Publish Subscribe Model

- Each subscriber only receives a subset of the messages that have been sent by the Publisher.
- Receive the message topics or categories they have subscribed to.
- There are two methods of filtering out unrequired messages:
  - Topic based filter
  - Content based filter
  -

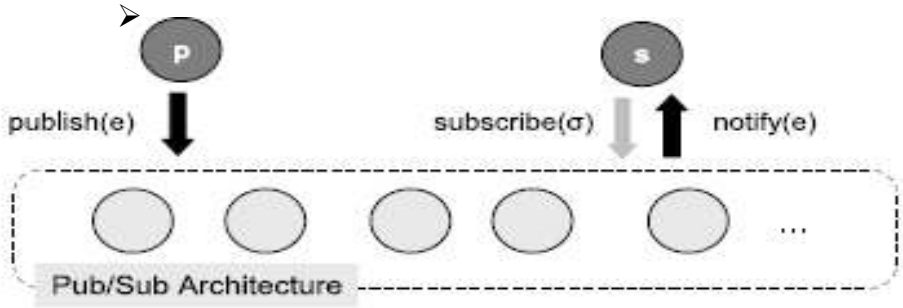


Figure 2.9 High Level View of A Publish/Subscribe System

- A generic pub/sub communication system is often referred as Event Service or Notification Service.

- System composed of a set of nodes distributed over a communication network.
- The clients of this system are divided according to their role into publishers and subscribers.
- Clients are not required to communicate directly among themselves.
- The interaction takes place through the nodes of the pub/sub system.

Elements of a Publish/Subscribe System

- A publisher submits a piece of information e (i.e., an event) to the pub/sub system by executing the publish(e) operation.
- An event is structured as a set of attribute-value pairs.
- Each attribute has a *name, a simple character* string, and a *type.*
- The type is generally one of the common primitive data types defined in programming languages or query languages (e.g. integer, real, string, etc.).
- On the subscriber's side, interest in specific events is expressed through subscriptions.
- *A* subscription is a filter over a portion of the event content (or the whole of it).
- Expressed through a set of constraints that depend on the subscription language.
- A subscriber installs and removes a subscription from the pub/sub system by executing the subscribe() and unsubscribe() operations respectively.
- An event e matches a subscription if it satisfies all the declared constraints on the corresponding attributes.
- The task of verifying whenever an event e matches a subscription is called matching.

Semantics of a Publish/subscribe System

- When a process issues a subscribe/unsubscribe operation, the pub/sub system is not immediately aware of the occurred event.
- The registration (resp. cancellation) of a subscription takes a certain amount of time, denoted as Tsub, to be stored into the system.

This time encompass the update of the internal data structures of the pub/sub system and the network delay due to the routing of the subscription.

Three properties:

- Safety (Legality): A subscriber cannot be notified for an information it is not interested in.
- Safety (Validity): A subscriber cannot be notified for an event that has not been previously published.
- Liveness: The delivery of a notification for an event is guaranteed only for those subscribers that subscribed at a time at least Tsub before the event was published.

Quality of Service in Publish/Subscribe Systems

- Reliable delivery
- Timeliness
- Security and trust

Reliable delivery

- Reliable delivery of an event means determining the subscribers that have to receive a published event, as stated by the liveness property and delivering the event to all of them.

Timeliness

- Real-time applications often require strict control over the time elapsed by a piece of information to reach all its consumers.
- They are typically deployed over dedicated infrastructures or simply managed environments where synchronous message delivery can be safely assumed.

Security and trust

- A subscriber wants to trust authenticity of the events it receives from the system.
- Generated by a trusty publisher and the information they contains have not been corrupted.
- Subscribers have to be trusted for what concerns the subscriptions they issue.
- Since an event is in general delivered to several subscribers, the producer/consumer trust relationship that commonly occur in a point-to-point communication, in pub/sub system must involve multiple participants

Subscription Models

- Topic based Model
- Type based Model
- Concept based Model
- Content based Model

Topic-based Model

- Events are grouped in topics.
- A subscriber declares its interest for a particular topic to receive all events pertaining to that topic.
- Each topic corresponds to a logical channel ideally connecting each possible publisher to all interested subscribers.
- Requires the messages to be broadcasted into logical channels.
- Subscribers only receive messages from logic channels they care about (and have subscribed to).

Type based Model

- Pub/sub variant events are actually objects belonging to a specific type, which can thus encapsulate attributes as well as methods.
- Types represent a more robust data model for application developer.
- Enforce type-safety at the pub/sub system, rather than inside the application.
- The declaration of a desired type is the main discriminating attribute.

Concept based Model

- Allows to describe event schema at a higher level of abstraction by using ontologies.
- Provide a knowledge base for an unambiguous interpretation of the event structure, by using metadata and mapping functions.

Content based Model

- System allows subscribers to receive messages based on the content of the messages.

Subscribers themselves must sort out junk messages from the ones they want.

Benefits

Loose coupling

- The publisher is not aware of the number of subscribers, of the identities of the subscribers, or of the message types that the subscribers are subscribed to.

Improved security

- The communication infrastructure transports the published messages only to the applications that are subscribed to the corresponding topic.
- Specific applications can exchange messages directly, excluding other applications from the message exchange.

Improved testability.

- Topics usually reduce the number of messages that are required for testing.

Separation of concerns

- Due to the simplistic nature of the architecture, developers can exercise fine grained separation of concerns by dividing up message types to serve a single simple purpose each.
- Eg. data with a topic "/cats" should only contain information about cats.

Reduced cognitive load for subscribers

- Subscribers need not concern themselves with the inner workings of a publisher.
- They do not even have to access to the source code.
- Subscribers only interact with the publisher through the public API exposed by the publisher.

Drawbacks

Increased complexity.

Publish/Subscribe requires you to address the following:

- To design a message classification scheme for topic implementation.
- To implement the subscription mechanism.
- To modify the publisher and the subscribers.

Increased maintenance effort.

- Managing topics requires maintenance work.
- Organizations that maintain many topics usually have formal procedures for their use.

Decreased performance

- Subscription management adds overhead.
- This overhead increases the latency of message exchange, and this latency decreases performance.

Inflexibility of data sent by publisher

- The publish/subscribe model introduces high semantic coupling in the messages passed by the publishers to the subscribers.
- Once the structure of the data is established, it becomes difficult to change.
- In order to change the structure of the messages, all of the subscribers must be altered to accept the changed format

Instability of Delivery

- The publisher does not have perfect knowledge of the status of the systems listening to the messages.
- For instance, publish/subscribe is commonly used for logging systems.
- If a logger subscribing to the 'Critical' message type crashes or gets stuck in an error state, then the 'Critical' messages may be lost!
- Then any services depending on the error messages will be unaware of the problems with the publisher.

Applications

Used in a wide range of group communication applications including

- Software Distribution
- Internet TV
- Audio or Video-conferencing
- Virtual Classroom

- Multi-party Network Games
- Distributed Cache Update

It can also be used in even larger size group communication applications, such as broadcasting and content distribution.

- News and Sports Ticker Services
- Real-time Stock Quotes and Updates
- Market Tracker
- Popular Internet Radio Sites

2.5 VIRTUALIZATION

- Virtualization is a technique, which allows sharing single physical instance of an application or resource among multiple organizations or tenants (customers).
- Virtualization is a proved technology that makes it possible to run multiple operating system and applications on the same server at same time.
- Virtualization is the process of creating a logical(virtual) version of a server operating system, a storage device, or network services.
- The technology that work behind virtualization is known as a virtual machine monitor(VM), or virtual manager which separates compute environments from the actual physical infrastructure.
- Virtualization -- the abstraction of computer resources.
- Virtualization hides the physical characteristics of computing resources from their users, applications, or end users.
- This includes making a single physical resource (such as a server, an operating system, an application, or storage device) appear to function as multiple virtual resources.
- It can also include making multiple physical resources (such as storage devices or servers) appear as a single virtual resource.
- In computing, virtualization refers to the act of creating a virtual (rather than actual) version of something, like computer hardware platforms, operating systems, storage devices, and computer network resources

- Creation of a virtual machine over existing operating system and hardware.
- **Host machine:** The machine on which the virtual machine is created.
- **Guest machine:** virtual machines referred as a guest machine.
- **Hypervisor:** Hypervisor is a firmware or low-level program that acts as a Virtual Machine Manager.

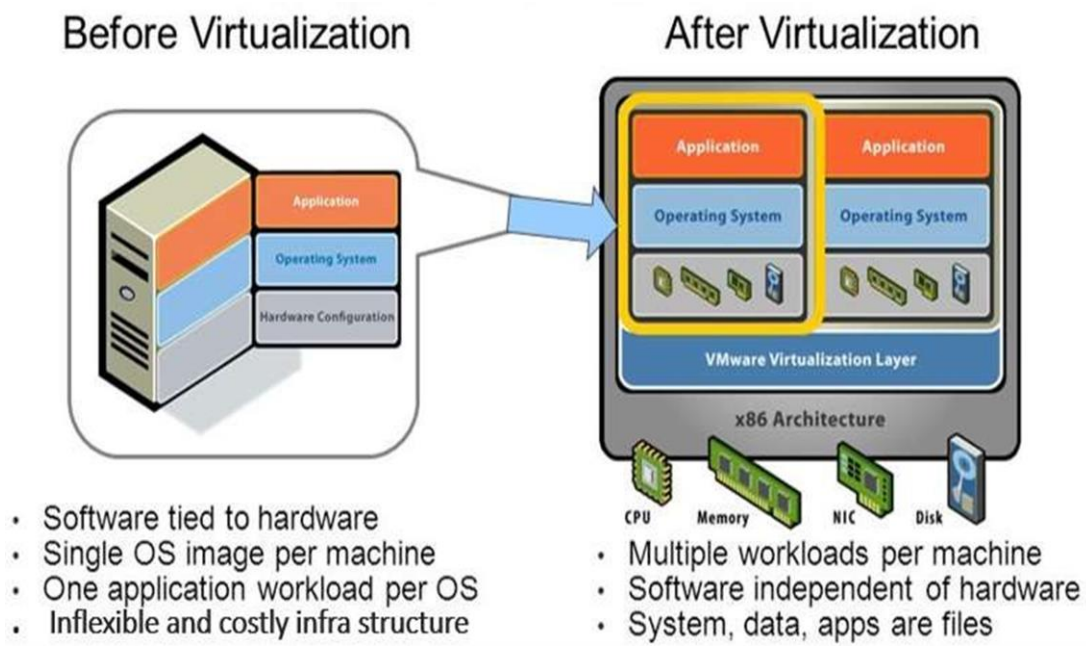


Figure 2.10 Virtualization Example

Advantages of Virtualization:

1. Reduced Costs.
2. Efficient hardware Utilization.
3. Virtualization leads to better resource Utilization and increase performance
4. Testing for software development.
5. Increase Availability
6. Save energy
7. Shifting all your Local Infrastructure to Cloud in a day
8. Possibility to Divide Services
9. Running application not supported by the host.

Disadvantages of Virtualization:

1. Extra Costs.
2. Software Licensing.

2.6 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.

The idea is to separate the hardware from the software to yield better system efficiency. For example, computer users gained access to much enlarged memory space when the concept of virtual memory was introduced. Similarly, virtualization techniques can be applied to enhance the use of compute engines, networks and storage.

2.6.1 Levels of Virtualization:

A traditional computer runs with host operating system specially tailored for its hardware architecture, as shown in Figure 2.11 (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 2.11 (b). This virtualization layer is known as hypervisor or virtual machine monitor (VMM) .The VMs are shown in the upper boxes, where applications run with their own guest OS over the virtualized CPU, memory, and I/O resources. The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs, exclusively. The virtualization software creates the abstraction of VMs by interposing a virtualization layer at various levels of a computer system. Common virtualization layers include the instruction set architecture (ISA) level, hardware level, operating system level, library support level, and application level.

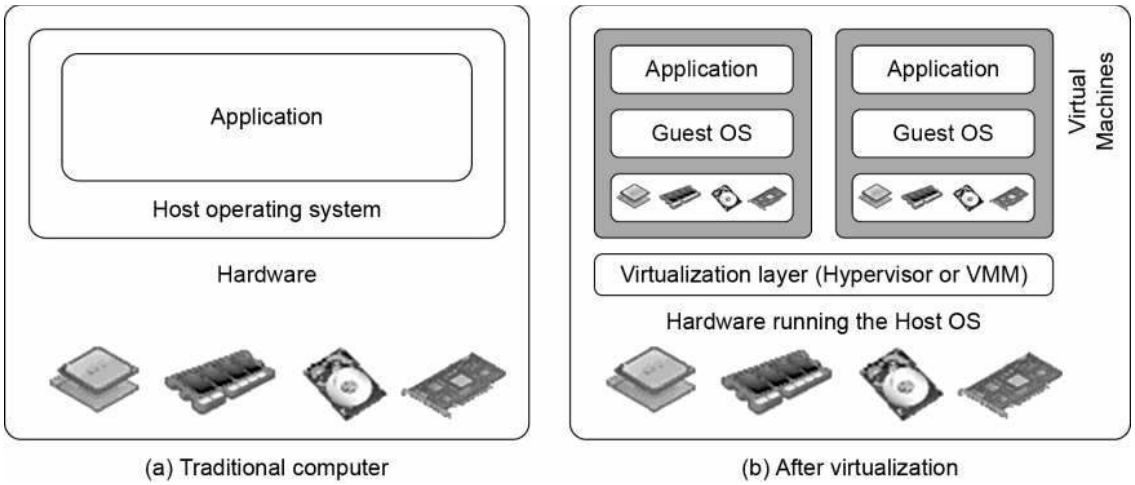


Figure 2.11 The architecture of a computer system before and after Virtualization

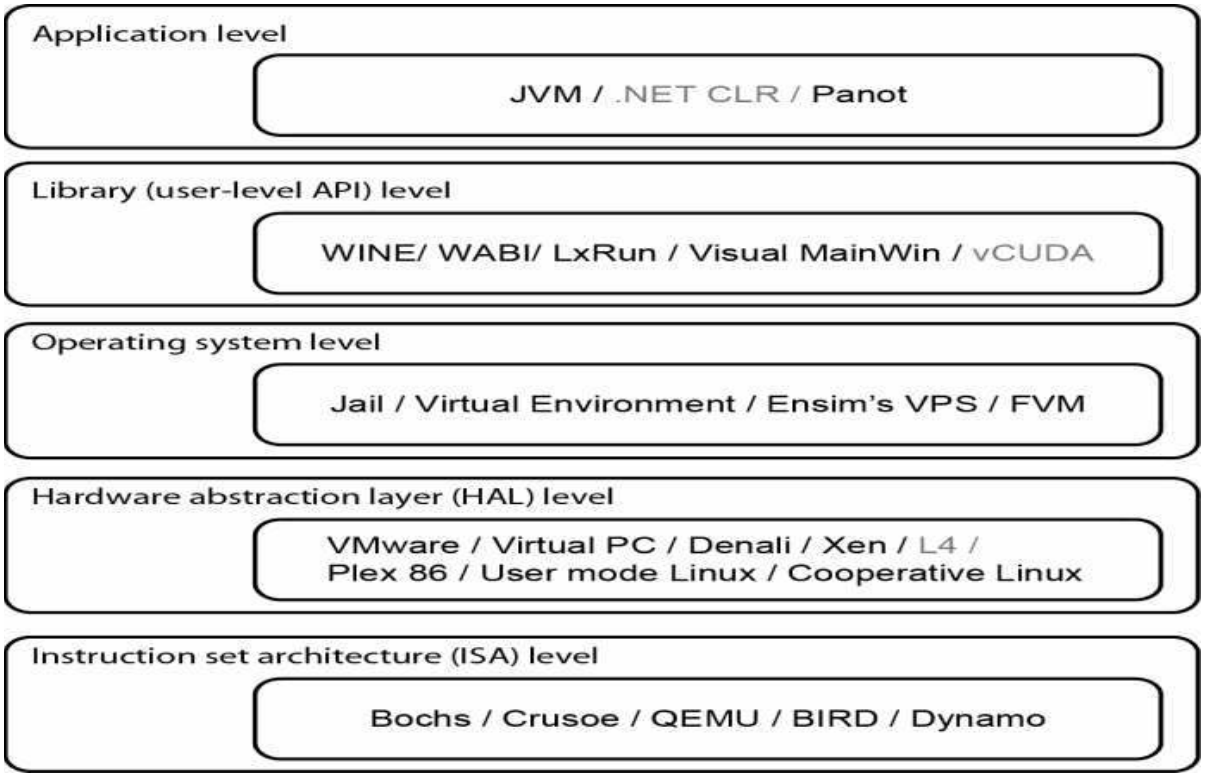


Figure 2.12 Virtualization ranging from hardware to applications in five abstraction levels.

Instruction Set Architecture Level:

At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. For example, MIPS binary code can run on an x86-based host machine with the help of ISA emulation. With this approach, it is possible to run a large amount of legacy binary

code written for various processors on any given new hardware host machine. Instruction set emulation leads to virtual ISAs created on any hardware machine.

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

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

Hardware Abstraction Level:

Hardware-level virtualization is performed right on top of the bare hardware. The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices. The intention is to upgrade the hardware utilization rate by multiple users concurrently.

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 datacenters.

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

Library Support Level:

Most applications use APIs exported by user level libraries rather than using lengthy system calls by the OS. Since most systems provide well documented APIs, such an interface becomes another candidate for virtualization.

Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks. The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts. Another

example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.

User-Application Level:

Virtualization at the application level virtualizes an application as a VM. On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process-level virtualization. The most popular approach is to deploy high level language (HLL)VMs.

2.6.2 VMM Design Requirements and Providers

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. Each time programs access the hardware the VMM captures the process. VMM acts as a traditional OS.

One hardware component, such as the CPU, can be virtualized as several virtual copies. Therefore, several traditional operating systems which are the same or different can sit on the same set of hardware simultaneously.

Three requirements for a VMM

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

Table 3.2 Comparison of Four VMM and Hypervisor Software Packages

Provider and References	Host CPU	Host OS	Guest OS	Architecture
VMware Workstation [71]	x86, x86-64	Windows, Linux	Windows, Linux, Solaris, FreeBSD, Netware, OS/2, SCO, BeOS, Darwin	Full Virtualization
VMware ESX Server [71]	x86, x86-64	No host OS	The same as VMware Workstation	Para-Virtualization
Xen [7,13,42]	x86, x86-64, IA-64	NetBSD, Linux, Solaris	FreeBSD, NetBSD, Linux, Solaris, Windows XP and 2003 Server	Hypervisor
KVM [31]	x86, x86-64, IA-64, S390, PowerPC	Linux	Linux, Windows, FreeBSD, Solaris	Para-Virtualization

2.6.3 Virtualization Support at the OS Level

With the help of VM technology, a new computing mode known as cloud computing is emerging. Cloud computing is transforming the computing landscape by shifting the hardware and staffing costs of managing a computational center to third parties, just like banks. However, cloud computing has at least two challenges.

- The first is the ability to use a variable number of physical machines and VM instances depending on the needs of a problem.
- The second challenge concerns the slow operation of instantiating new VMs.

Currently, new VMs originate either as fresh boots or as replicates of a template VM, unaware of the current application state. Therefore, to better support cloud computing, a large amount of research and development should be done.

Why OS-Level Virtualization?

To reduce the performance overhead of hardware-level virtualization, even hardware modification is needed. OS-level virtualization provides a feasible solution for these hardware-level virtualization issues. Operating system virtualization inserts a virtualization layer inside an operating system to partition a machine's physical resources. It enables multiple isolated VMs within a single operating system kernel. This kind of VM is often called a virtual execution environment (VE), Virtual Private System (VPS), or simply container. From the user's point of view, VEs look like real servers. This means a VE has its own set of processes, file system, user accounts, network interfaces with IP addresses, routing tables, firewall rules, and other personal settings. Although VEs can be customized for different people, they share the same operating system kernel.

Advantages of OS Extensions

(1) VMs at the operating system level have minimal startup/shutdown costs, low resource requirements, and high scalability.

(2) For an OS-level VM, it is possible for a VM and its host environment to synchronize state changes when necessary.

These benefits can be achieved via two mechanisms of OS-level virtualization:

(1) All OS-level VMs on the same physical machine share a single operating system kernel

(2) The virtualization layer can be designed in a way that allows processes in VMs to access as many resources of the host machine as possible, but never to modify them.

Virtualization on Linux or Windows Platforms

Virtualization support on the Windows-based platform is still in the research stage. The Linux kernel offers an abstraction layer to allow software processes to work with and operate on resources without knowing the hardware details. New hardware may need a new Linux kernel to support. Therefore, different Linux platforms use patched kernels to provide special support for extended functionality.

Table 3.3 Virtualization Support for Linux and Windows NT Platforms

Virtualization Support and Source of Information	Brief Introduction on Functionality and Application Platforms
Linux vServer for Linux platforms (http://linux-vserver.org/)	Extends Linux kernels to implement a security mechanism to help build VMs by setting resource limits and file attributes and changing the root environment for VM isolation
OpenVZ for Linux platforms [65]; http://ftp.openvz.org/doc/OpenVZ-Users-Guide.pdf)	Supports virtualization by creating *virtual private servers (VPSes)*; the VPS has its own files, users, process tree, and virtual devices, which can be isolated from other VPSes, and checkpointing and live migration are supported
FVM (Feather-Weight Virtual Machines) for virtualizing the Windows NT platforms [78])	Uses system call interfaces to create VMs at the NY kernel space; multiple VMs are supported by virtualized namespace and copy-on-write

2.6.4 Middleware Support for Virtualization

Library-level virtualization is also known as user-level Application Binary Interface (ABI) or API emulation. This type of virtualization can create execution environments for running alien programs on a platform rather than creating a VM to run the entire operating system. API call interception and remapping are the key functions performed. This provides an overview of several library-level virtualization systems: namely the Windows Application Binary Interface (WABI), lxrun, WINE, Visual MainWin, and Vcuda.

Table 3.4 Middleware and Library Support for Virtualization

Middleware or Runtime Library and References or Web Link	Brief Introduction and Application Platforms
WABI (http://docs.sun.com/app/docs/doc/802-6306)	Middleware that converts Windows system calls running on x86 PCs to Solaris system calls running on SPARC workstations
Lxrun (Linux Run) (http://www.ugcs.caltech.edu/~steven/lxrun/)	A system call emulator that enables Linux applications written for x86 hosts to run on UNIX systems such as the SCO OpenServer
WINE (http://www.winehq.org/)	A library support system for virtualizing x86 processors to run Windows applications under Linux, FreeBSD, and Solaris
Visual MainWin (http://www.mainsoft.com/)	A compiler support system to develop Windows applications using Visual Studio to run on Solaris, Linux, and AIX hosts
vCUDA (Example 3.2) (IEEE IPDPS 2009 [57])	Virtualization support for using general-purpose GPUs to run data-intensive applications under a special guest OS

2.7 Virtualization Structures/Tools and Mechanisms

There are three typical classes of VM architecture. Before virtualization, the operating system manages the hardware. After virtualization, a virtualization layer is inserted between the hardware and the operating system. In such a case, the virtualization layer is responsible for converting portions of the real hardware into virtual hardware. Therefore, different operating systems such as Linux and Windows can run on the same physical machine, simultaneously.

Depending on the position of the virtualization layer, there are several classes of VM architectures, namely the hypervisor architecture, para-virtualization, and host based virtualization. The hypervisor is also known as the VMM (Virtual Machine Monitor). They both perform the same virtualization operations.

2.7.1 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. The hypervisor provides hypercalls for the guest OSes and applications. Depending on the functionality, a hypervisor can assume a micro-kernel architecture like the Microsoft Hyper-V. Or it can assume monolithic hypervisor architecture like the VMware ESX for server virtualization.

A micro-kernel hypervisor includes only the basic and unchanging functions (such as physical memory management and processor scheduling). The device drivers and other changeable components are outside the hypervisor. A monolithic hypervisor implements all the aforementioned functions, including those of the device drivers.

Therefore, the size of the hypervisor code of a micro-kernel hypervisor is smaller than that of a monolithic hypervisor. Essentially, a hypervisor must be able to convert physical devices into virtual resources dedicated for the deployed VM to use.

The Xen Architecture:

The core components of a Xen system are the hypervisor, kernel, and applications. The organization of the three components is important. Like other virtualization systems, many guest OSes can run on top of the hypervisor. However, not all guest OSes are created equal, and one in particular controls the others.

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

2.7.2 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. The guest OSes and their applications consist of noncritical and critical instructions. In a host-based system, both a host OS and a guest OS are used. A virtualization software layer is built between the host OS and guest OS.

Full Virtualization:

With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software. Both the hypervisor and VMM approaches are considered full virtualization.

Binary Translation of Guest OS Requests Using a VMM :

VMware puts the VMM at Ring 0 and the guest OS at Ring 1. The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions.

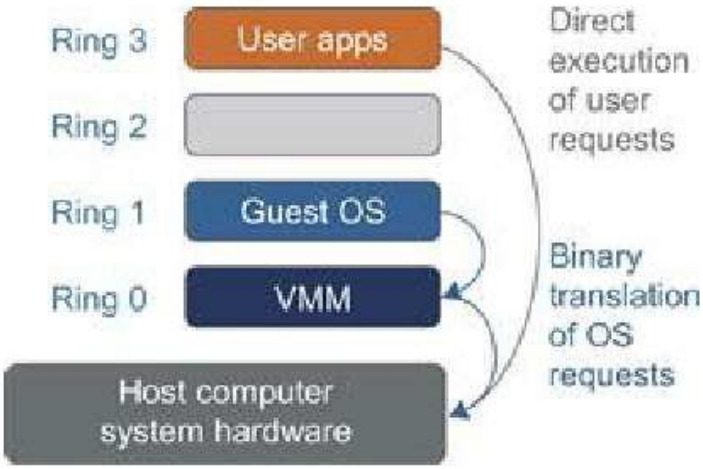


Figure 2.13 Indirect execution of complex instructions via binary translation of guest OS requests using the VMM plus direct execution of simpleinstructions on the same host.

The method used in this emulation is called binary translation. Therefore, full virtualization combines binary translation and direct execution. The guest OS is completely decoupled from the underlying hardware. Consequently, the guest OS is unaware that it is being virtualized. Binary translation employs a code cache to store translated hot instructions to improve performance, but it increases the cost of memory usage.

Host-Based Virtualization:

An alternative VM architecture is to install a virtualization layer on top of the host OS. This host OS is still responsible for managing the hardware. The guest OSes are installed and run on top of the virtualization layer. Dedicated applications may run on the VMs. Certainly, some other applications can also run with the host OS directly. This host-based architecture has some distinct advantages, as enumerated next. First, the user can install this VM architecture without modifying the host OS. The virtualizing software can rely on the host OS to provide device drivers and other low level services. This will simplify the VM design and ease its deployment. Second, the host-based approach appeals to many host machine configurations.

Compared to the hypervisor/VMM architecture, the performance of the host based architecture may also be low. When an application requests hardware access, it involves four layers of mapping which downgrades performance significantly.

2.7.3 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. Performance degradation is a critical issue of a virtualized system. No one wants to use a VM if it is much slower than using a physical machine.

The virtualization layer can be inserted at different positions in a machine software stack. However, para-virtualization attempts to reduce the virtualization overhead, and thus improve performance by modifying only the guest OS kernel. The guest operating systems are para-virtualized. 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.

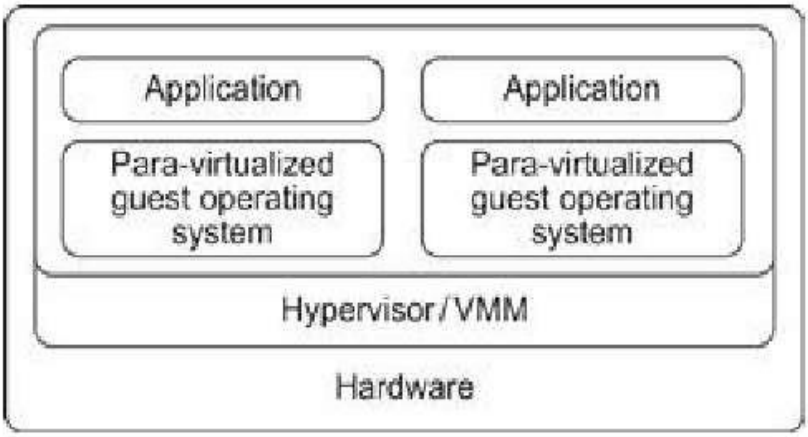


Figure 2.14 Para-virtualized VM architecture

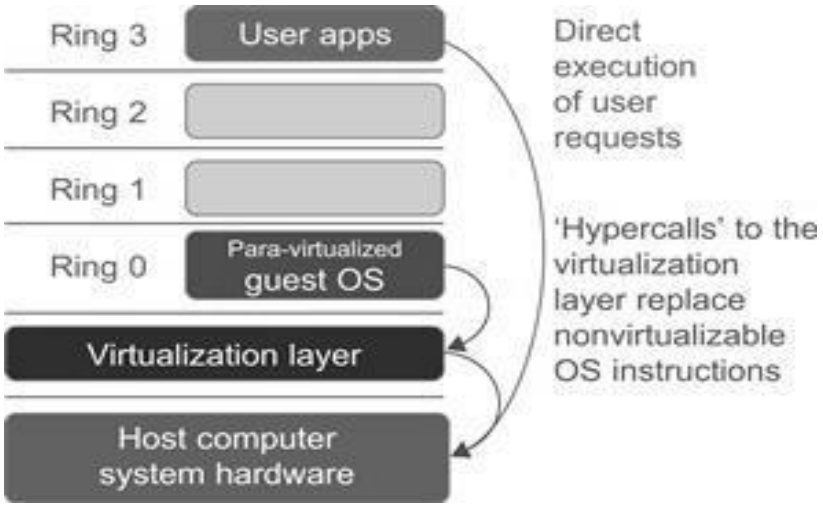


Figure 2.15 The use of a para-virtualized guest OS assisted by an intelligent compiler to replace non virtualizable OS instructions by hyper calls.

Para-Virtualization Architecture:

When the x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS. According to the x86 ring definitions, the virtualization layer should also be installed at Ring 0. The para-virtualization replaces non virtualizable instructions with hyper calls that communicate directly with the hypervisor or VMM. However, when the guest OS kernel is modified for virtualization, it can no longer run on the hardware directly.

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

KVM (Kernel-Based VM):

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

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

The guest OS kernel is modified to replace the privileged and sensitive instructions with hyper calls to the hypervisor or VMM. Xen assumes such a para virtualization architecture. The guest OS running in a guest domain may run at Ring 1instead of at Ring 0. This implies that the guest OS may not be able to execute some privileged and sensitive instructions. The privileged instructions are implemented by hypercalls to the hypervisor. After replacing the instructions with hyper calls, the modified guest OS emulates the behavior of the original guest OS.

2.8 VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

To support virtualization, processors such as the x86 employ a special running mode and instructions, known as hardware-assisted virtualization. In this way, the VMM and guest OS run in different modes and all sensitive instructions of the guest OS and its applications are trapped in the VMM. To save processor states, modes witching are completed by hardware. For the x86architecture, Intel and AMD have proprietary technologies for hardware-assisted virtualization.

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

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

CPU Virtualization:

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

Privileged instructions - Privileged instructions execute in a privileged mode and will be trapped if executed outside this mode.

Control sensitive instructions - Control-sensitive instructions attempt to change the configuration of resources used.

Behavior-sensitive instructions - Behavior-sensitive instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.

A CPU architecture is virtualizable if it supports the ability to run the VM's privileged and privileged instructions in the CPU's user mode while the VMM runs in supervisor mode. When the privileged instructions including control- and behavior sensitive instructions of a VM are executed, they are trapped in the VMM. In this case, the VMM acts as a unified mediator for hardware access from different VMs to guarantee the correctness and stability of the whole system. RISC CPU architectures can be naturally virtualized because all control- and behavior-sensitive instructions are privileged instructions.

Hardware-Assisted CPU Virtualization:

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

Memory Virtualization:

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

However, in a virtual execution environment, virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical

memory of the VMs. That means a two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory. Furthermore, MMU virtualization should be supported, which is transparent to the guest OS. The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs. But the guest OS cannot directly access the actual machine memory. The VMM is responsible for mapping the guest physical memory to the actual machine memory. Figure 2.16 shows the two-level memory mapping procedure.

I/O Virtualization:

I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware. There are three ways to implement I/O virtualization:

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

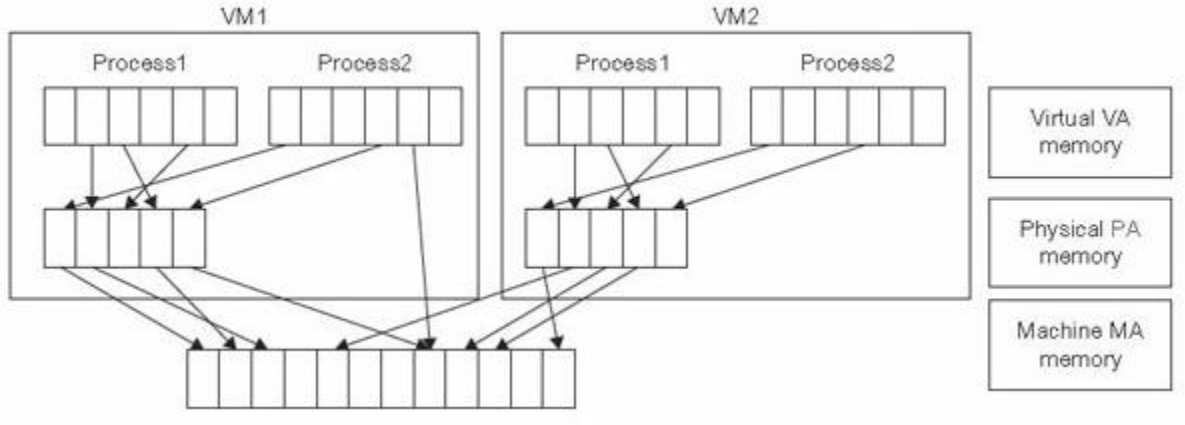


Figure 2.16 Two-level memory mapping procedure.

Full device emulation is the first approach for I/O virtualization. Generally, this approach emulates well known, real-world devices. All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated in software. This software is located in the VMM and acts as a virtual device. The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices.

A single hardware device can be shared by multiple VMs that run concurrently. However, software emulation runs much slower than the hardware it emulates. The para

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

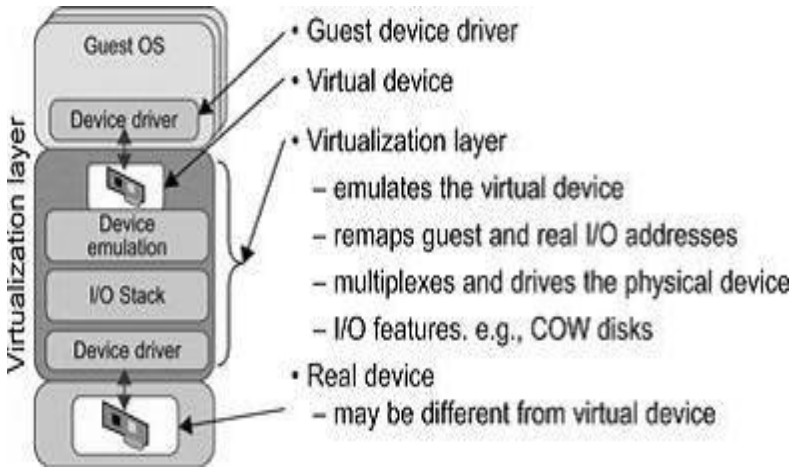


Figure 2.17 Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices into the virtual devices for the guest device driver to use.

Virtualization in Multi-Core Processors:

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

There are mainly two difficulties: Application programs must be parallelized to use all cores fully, and software must explicitly assign tasks to the cores, which is a very complex problem.

2.9 Virtualization Support and Disaster Recover:

One very distinguishing feature of cloud computing infrastructure is the use of system virtualization and the modification to provisioning tools. Virtualizations of servers on a shared cluster can consolidate web services. As the VMs are the containers of cloud services, the provisioning tools will first find the corresponding physical machines and deploy the VMs to those nodes before scheduling the service to run on the virtual nodes.

In addition, in cloud computing, virtualization also means the resources and fundamental infrastructure are virtualized. The user will not care about the computing resources that are used for pro-viding the services. Cloud users do not need to know and have no way to discover physical resources that are involved while processing a service request.

Also, application developers do not care about some infrastructure issues such as scalability and fault tolerance (i.e., they are virtualized). Application developers focus on service logic. Figure 2.18 shows the infrastructure needed to virtualize the servers in a data center for implementing specific cloud applications.

2.9.1 Hardware Virtualization

In many cloud computing systems, virtualization software is used to virtualize the hardware. System virtualization software is a special kind of software which simulates the execution of hardware and runs even unmodified operating systems. Cloud computing systems use virtualization software as the running environment for legacy software such as old operating systems and unusual applications. Virtualization software is also used as the platform for developing new cloud applications that enable developers to use any operating systems and programming environments they like.

The development environment and deployment environment can now be the same, which eliminates some runtime problems. Some cloud computing providers have used virtualization technology to provide this service for developers. As mentioned before, system virtualization software is considered the hardware analog mechanism to run an unmodified operating system, usually on bare hardware directly, on top of software. Table 4.4 lists some of the system virtualization software in wide use at the time of this writing. Currently, the VMs installed on a cloud computing platform are mainly used for hosting third-party programs. VMs provide flexible runtime services to free users from worrying about the system environment

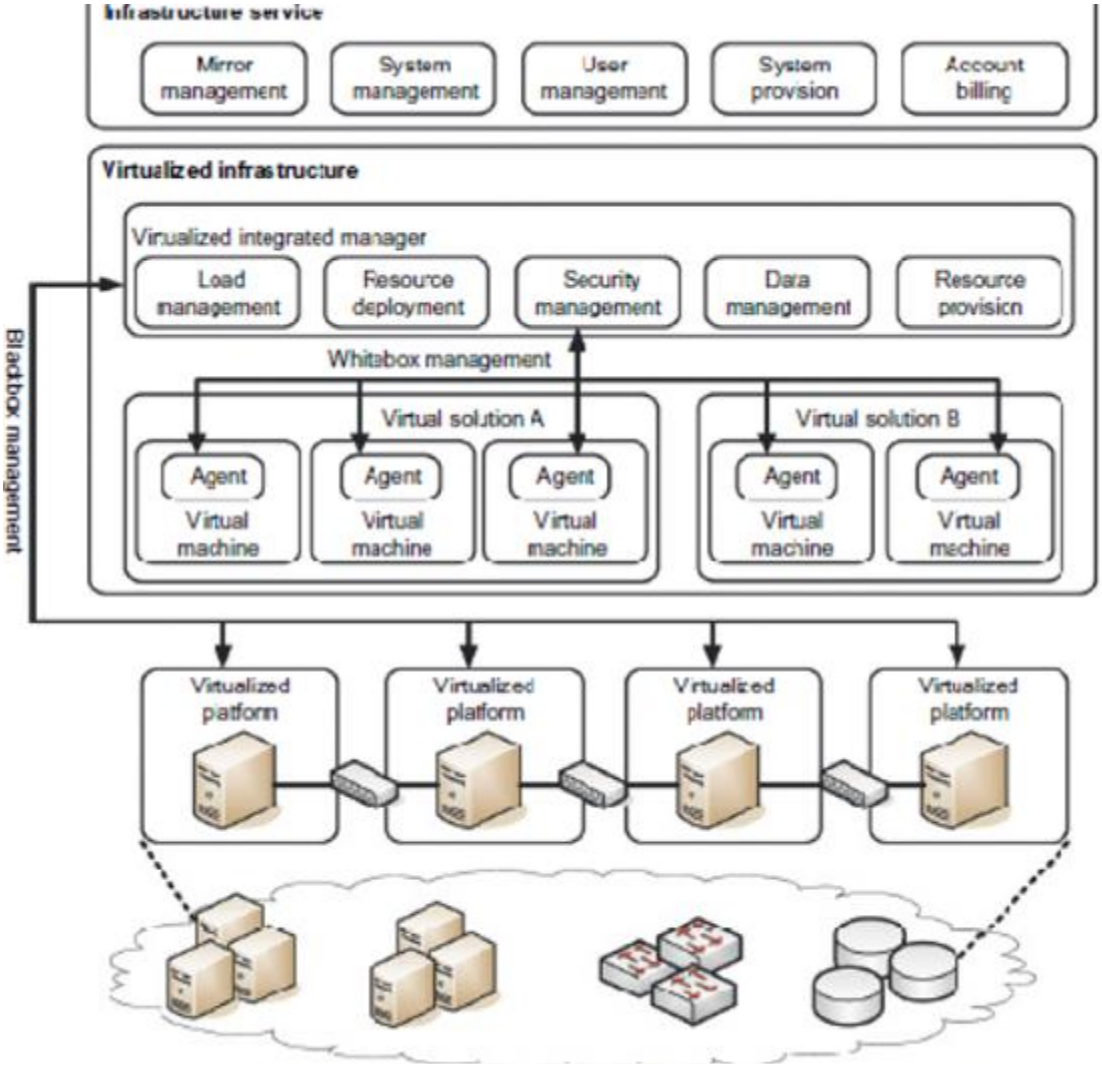
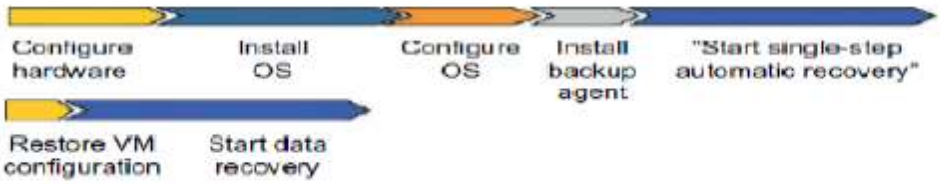


Figure 2.18 Virtualized Storage server and Networks

Using VMs in a cloud computing platform ensures extreme flexibility for users. As the computing resources are shared by many users, a method is required to maximize the users' privileges and still keep them separated safely. Traditional sharing of cluster resources depends on the user and group mechanism on a system. Such sharing is not flexible. Users cannot customize the system for their special purposes. Operating systems cannot be changed. The separation is not complete.

Table 4.4 Virtualized Resources in Compute, Storage, and Network Clouds [4]

Provider	AWS	Microsoft Azure	GAE
Compute cloud with virtual cluster of servers	x86 instruction set, Xen VMs, resource elasticity allows scalability through virtual cluster, or a third party such as RightScale must provide the cluster	Common language runtime VMs provisioned by declarative descriptions	Predefined application framework handlers written in Python, automatic scaling up and down, server failover inconsistent with the web applications
Storage cloud with virtual storage	Models for block store (EBS) and augmented key/blob store (SimpleDB), automatic scaling varies from EBS to fully automatic (SimpleDB, S3)	SQL Data Services (restricted view of SQL Server), Azure storage service	MegaStore/BigTable
Network cloud services	Declarative IP-level topology; placement details hidden, security groups restricting communication, availability zones isolate network failure, elastic IP applied	Automatic with user's declarative descriptions or roles of app. components	Fixed topology to accommodate three-tier web app. structure, scaling up and down is automatic and programmer invisible



Recovery overhead of a conventional disaster recovery scheme, compared with that required to recover from live migration of VMs.

An environment that meets one user's requirements often cannot satisfy another user. Virtualization allows users to have full privileges while keeping them separate. Users have full access to their own VMs, which are completely separate from other users' VMs. Multiple VMs can be mounted on the same physical server. Different VMs may run with different OSes. We also need to establish the virtual disk storage and virtual networks needed by the VMs. The virtualized resources form a resource pool.

The virtualization is carried out by special servers dedicated to generating the virtualized resource pool. The virtualized infrastructure (black box in the middle) is built with many virtualizing integration managers. These managers handle loads, resources, security, data, and provisioning functions.

2.9.2 Virtualization Support in Public Clouds

AWS provides extreme flexibility (VMs) for users to execute their own applications. GAE provides limited application-level virtualization for users to build applications only based on the services that are created by Google. Microsoft provides programming-level virtualization (.NET virtualization) for users to build their applications. The VMware tools apply to workstations, servers, and virtual infrastructure. The Microsoft tools are used on PCs and some special servers. The Xen Enterprise tool applies only to Xen-based servers.

Everyone is interested in the cloud; the entire IT industry is moving toward the vision of the cloud. Virtualization leads to HA, disaster recovery, dynamic load leveling, and rich provisioning support. Both cloud computing and utility computing leverage the benefits of virtualization to provide a scalable and autonomous computing environment.

2.9.3 Storage Virtualization for Green Data Centers

IT power consumption in the United States has more than doubled to 3 percent of the total energy consumed in the country. The large number of data centers in the country has contributed to this energy crisis to a great extent. More than half of the companies in the Fortune 500 are actively implementing new corporate energy policies. Recent surveys from both IDC and Gartner confirm the fact that virtualization had a great impact on cost reduction from reduced power consumption in physical computing systems. This alarming situation has made the IT industry become more energy aware.

With little evolution of alternate energy resources, there is an imminent need to con-serve power in all computers. Virtualization and server consolidation have already proven handy in this aspect. Green data centers and benefits of storage virtualization are considered to further strengthen the synergy of green computing.

2.9.4 Virtualization for IaaS

VM technology has increased in ubiquity. This has enabled users to create customized environments atop physical infrastructure for cloud computing.

Use of VMs in clouds has the following distinct benefits:

(1) System administrators consolidate workloads of underutilized servers in fewer servers;

(2) VMs have the ability to run legacy code without interfering with other APIs;

(3) VMs can be used to improve security through creation of sandboxes for running applications with questionable reliability;

(4) Virtualized cloud platforms can apply performance isolation, letting providers offer some guarantees and better QoS to customer applications.

2.9.5.5 VM Cloning for Disaster Recovery

VM technology requires an advanced disaster recovery scheme. One scheme is to recover one physical machine by another physical machine. The second scheme is to recover one VM by another VM. Traditional disaster recovery from one physical machine to another is rather slow, complex, and expensive. Total recovery time is attributed to the hardware configuration, installing and configuring the OS, installing the backup agents, and the long time to restart the physical machine. To recover a VM platform, the installation and configuration times for the OS and backup agents are eliminated.

Therefore, we end up with a much shorter disaster recovery time, about 40 percent of that to recover the physical machines. Virtualization aids in fast disaster recovery by VM encapsulation. The cloning of VMs offers an effective solution. The idea is to make a clone VM on a remote server for every running VM on a local server. Among all the clone VMs, only one needs to be active. The remote VM should be in a suspended mode.

A cloud control center should be able to activate this clone VM in case of failure of the original VM, taking a snapshot of the VM to enable live migration in a minimal amount of time. The migrated VM can run on a shared Internet connection.

Only updated data and modified states are sent to the suspended VM to update its state. The Recovery Property Objective (RPO) and Recovery Time Objective (RTO) are affected by the number of snapshots taken. Security of the VMs should be enforced during live migration of VMs.

UNIT III CLOUD ARCHITECTURE, SERVICES AND STORAGE

Layered Cloud Architecture Design – NIST Cloud Computing Reference Architecture – Public, Private and Hybrid Clouds - laaS – PaaS – SaaS – Architectural Design Challenges – Cloud Storage – Storage-as-a-Service – Advantages of Cloud Storage – Cloud Storage Providers – S3.

3.1 LAYERED ARCHITECTURE:

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 Goals
- ❖ Enabling Technologies for Clouds
- ❖ A Generic Cloud Architecture

Cloud Platform Design Goals

- Scalability
- Virtualization
- Efficiency
- Reliability
- Security

Cloud management receives the user request and finds the correct resources. Cloud calls the provisioning services which invoke the resources in the cloud. Cloud management software needs to support both physical and virtual machines

Enabling Technologies for Clouds

- Cloud users are able to demand more capacity at peak demand, reduce costs, experiment with new services, and remove unneeded capacity.
- Service providers can increase system utilization via multiplexing, virtualization and dynamic resource provisioning.
- Clouds are enabled by the progress in hardware, software and networking technologies
- Cloud users are able to demand more capacity at peak demand, reduce costs, experiment with new services, and remove unneeded capacity.
- Service providers can increase system utilization via multiplexing, virtualization and dynamic resource provisioning.

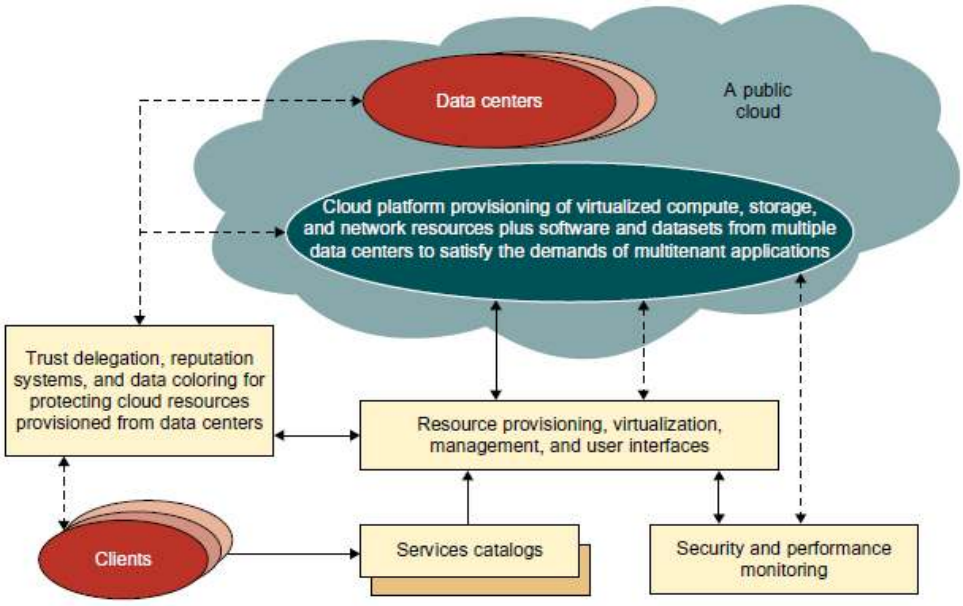
- Clouds are enabled by the progress in hardware, software and networking technologies

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

- The Internet cloud is envisioned as a massive cluster of servers.
- Servers are provisioned on demand to perform collective web services using data-center resources.
- The cloud platform is formed dynamically by provisioning or deprovisioning servers, software, and database resources.
- Servers in the cloud can be physical machines or VMs.
- User interfaces are applied to request services.

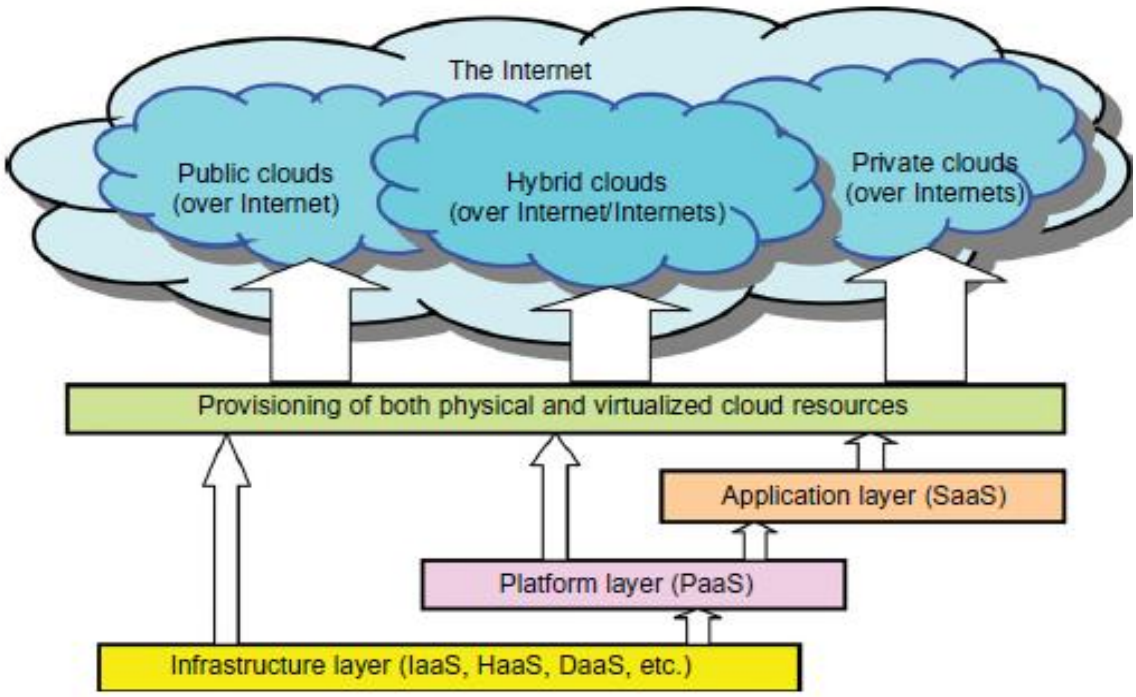


- The cloud computing resources are built into the data centers.
- Data centers are typically owned and operated by a third-party provider.

Consumers do not need to know the underlying technologies

- In a cloud, software becomes a service.
- Cloud demands a high degree of trust of massive amounts of data retrieved from large data centers.
- The software infrastructure of a cloud platform must handle all resource management and maintenance automatically.
- Software must detect the status of each node server joining and leaving.
- Cloud computing providers such as Google and Microsoft, have built a large number of data centers.
- Each data center may have thousands of servers.
- The location of the data center is chosen to reduce power and cooling costs.

Layered Cloud Architectural Development



- The architecture of a cloud is developed at three layers
  - Infrastructure
  - Platform
  - Application

- 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

Infrastructure Layer

- Foundation for building the platform layer.
- Built with virtualized compute, storage, and network resources.
- Provide the flexibility demanded by users.
- Virtualization realizes automated provisioning of resources and optimizes the infrastructure management process.

Platform Layer

- Foundation for implementing the application layer for SaaS applications.
- Used for general-purpose and repeated usage of the collection of software resources.
- Provides users with an environment to develop their applications, to test operation flows, and to monitor execution results and performance.

The platform should be able to assure users that they have scalability, dependability, and security protection

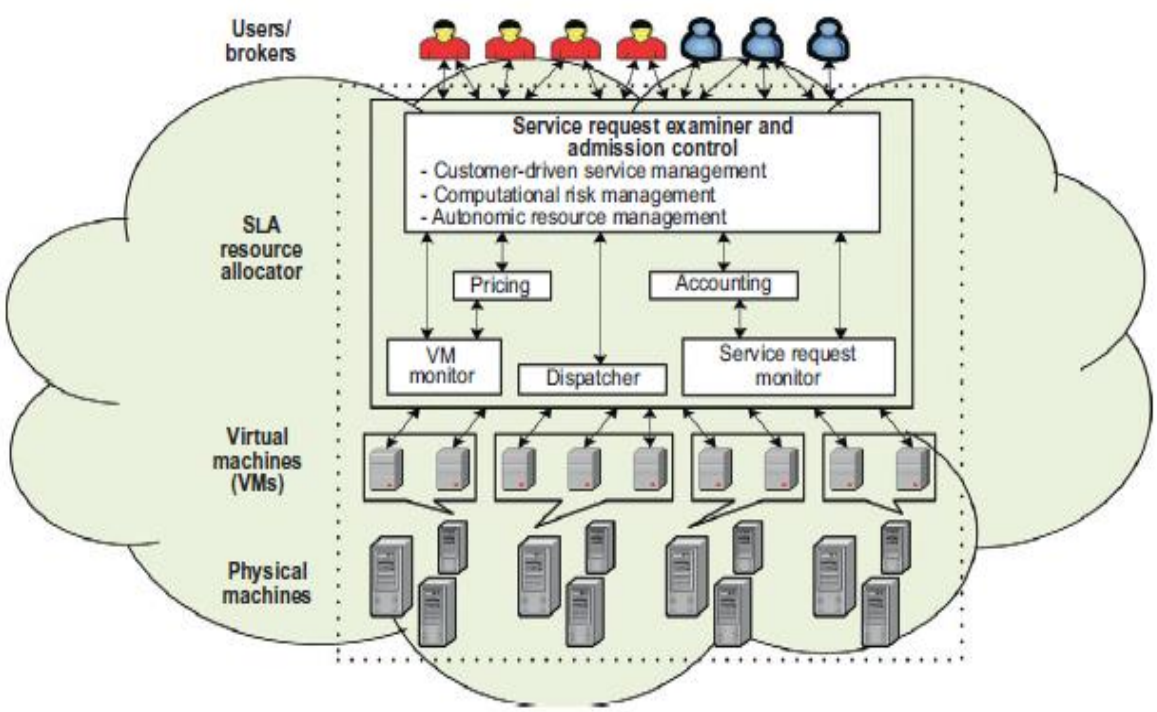
Application Layer

- 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 authentication services.
- The application layer is also heavily used by enterprises in business marketing and sales, consumer relationship management (CRM) and financial transactions.
- Not all cloud services are restricted to a single layer.
- Many applications may apply resources at mixed layers.
- Three layers are built from the bottom up with a dependence relationship.

Market-Oriented Cloud Architecture

- High-level architecture for supporting market-oriented resource allocation in a cloud computing environment.
- Users or brokers acting on user's behalf submit service requests to the data center.
- When a service request is first submitted, the service request examiner interprets the submitted request for QoS requirements.

Accept or Reject the request.



- **VM Monitor**: Latest status information regarding resource availability.
- **Service Request Monitor**: Latest status information workload processing
- **Pricing mechanism:**Decides how service requests are charged.
- **Accounting mechanism:**Maintains the actual usage of resources by requests to compute the final cost.
- VM Monitor mechanism keeps track of the availability of VMs and their resource entitlements.
- Dispatcher starts the execution of accepted service requests on allocated VMs.

  Service Request Monitor mechanism keeps track of the execution progress of service requests.

  Multiple VMs can be started and stopped on demand

Quality of Service Factors

QoS parameters

- Time
- Cost
- Reliability
- Trust/security

  QoS requirements cannot be static and may change over time.

3.1.1 CLOUD REFERENCE ARCHITECTURE

Definitions

- A model of computation and data storage based on “pay as you go” access to “unlimited” remote data center capabilities.
- A cloud infrastructure provides a framework to manage scalable, reliable, on-demand access to applications.
- Cloud services provide the “invisible” backend to many of our mobile applications.

High level of elasticity in consumption.

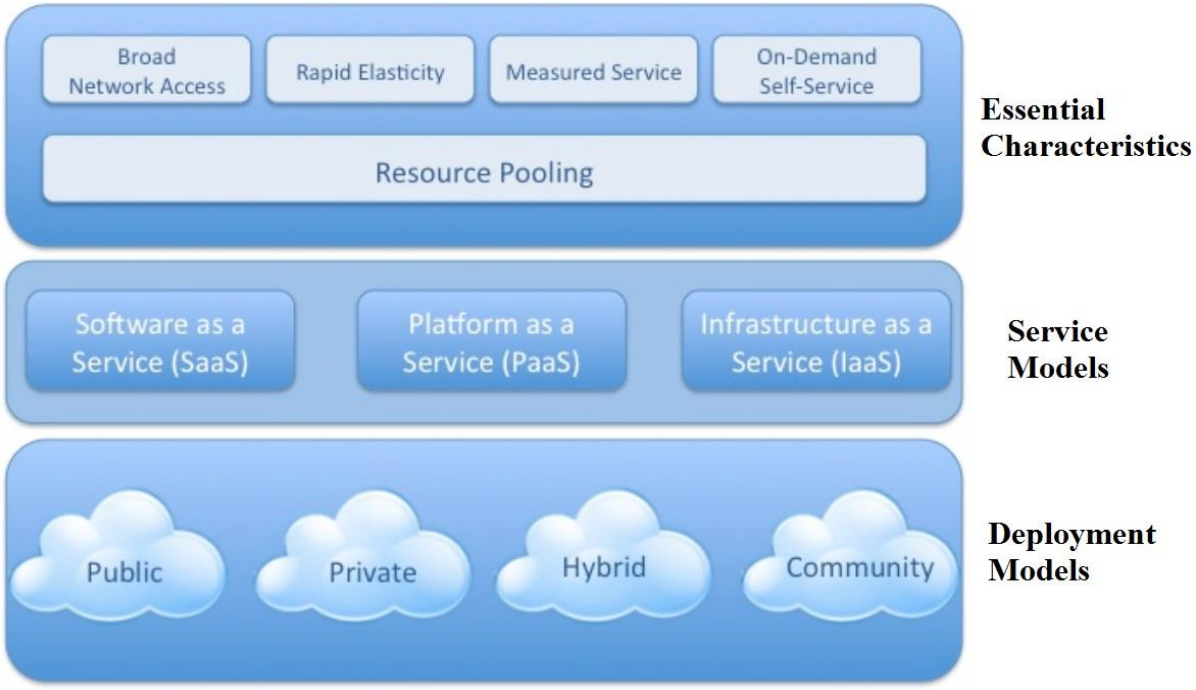
NIST Cloud Definition:

The National Institute of Standards and Technology (NIST) defines cloud computing as a

> **"pay-per-use model for enabling available, convenient and 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."**

Architecture

- Architecture consists of 3 tiers
  - Cloud Deployment Model
  - Cloud Service Model
  - Essential Characteristics of Cloud Computing .



Essential Characteristics 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 a service provider.

Essential Characteristics 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, laptops, and PDAs) as well as other traditional or cloudbased software services.

Essential Characteristics 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.

Essential Characteristics 4

- **Rapid elasticity.**
  - Capabilities can be rapidly and elastically provisioned - in some cases automatically - to quickly scale out; and rapidly released to quickly scale in.
  - To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

Essential Characteristics 5

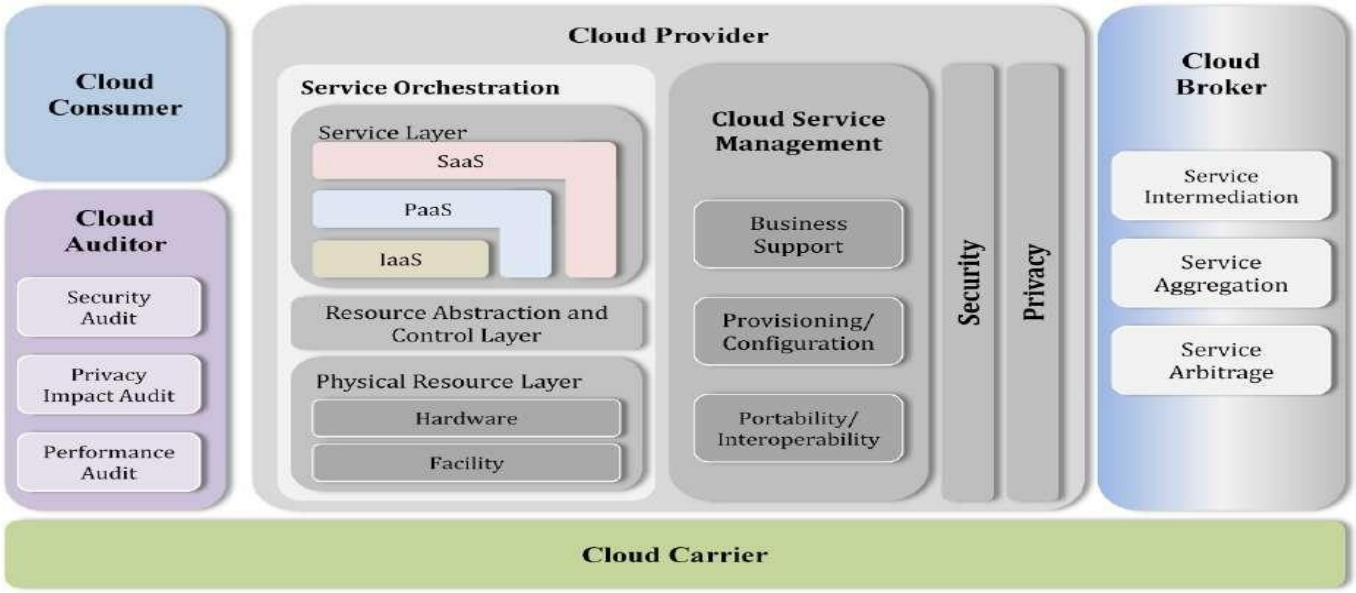
- **Measured service.**
  - Cloud systems automatically control and optimize resource usage by leveraging a metering capability at some level of abstraction appropriate to the type of service.

Resource usage can be monitored, controlled, and reported - providing transparency for both the provider and consumer of the service.

3.2 NIST (National Institute of Standards and Technology Background)

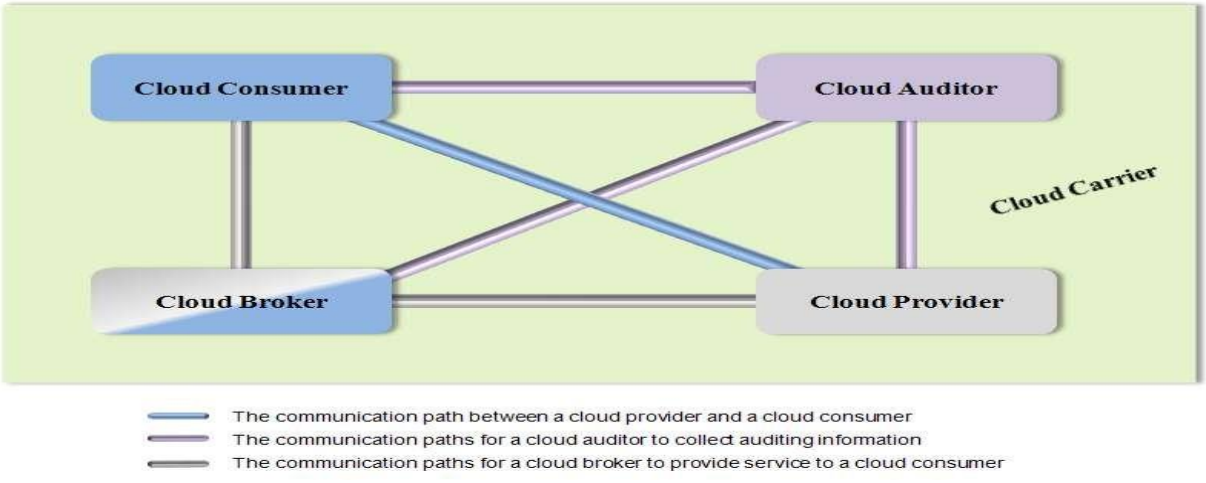
The goal is to accelerate the federal government's adoption of secure and effective cloud computing to reduce costs and improve services.

Cloud Computing Reference Architecture:



Actor	Definition
Cloud Consumer	A person or organization that maintains a business relationship with, and uses service from, *Cloud Providers*.
Cloud Provider	A person, organization, or entity responsible for making a service available to interested parties.
Cloud Auditor	A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation.
Cloud Broker	An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between *Cloud Providers* and *Cloud Consumers*.
Cloud Carrier	An intermediary that provides connectivity and transport of cloud services from *Cloud Providers* to *Cloud Consumers*.

Interactions between the Actors in Cloud Computing

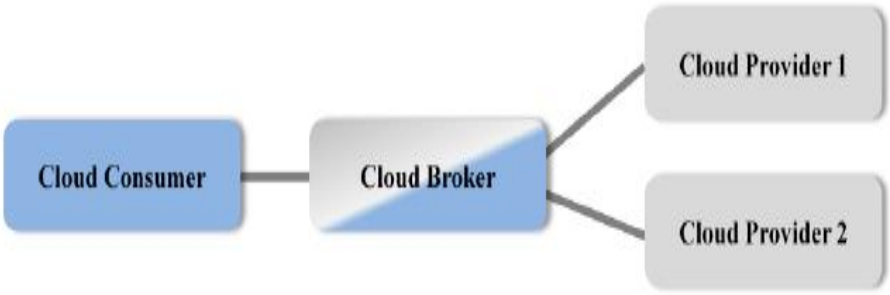


Example Usage Scenario 1:

- A cloud consumer may request service from a cloud broker instead of contacting a cloud provider directly.
- The cloud broker may create a new service by combining multiple services or by enhancing an existing service.

Usage Scenario- Cloud Brokers

- In this example, the actual cloud providers are invisible to the cloud consumer.
- The cloud consumer interacts directly with the cloud broker.

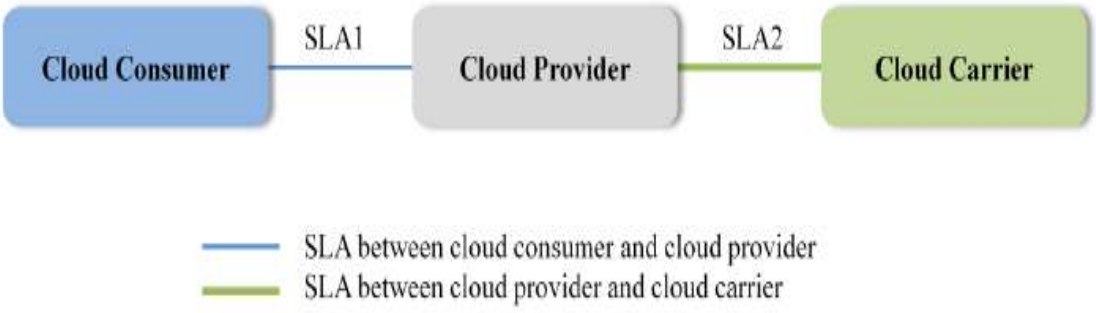


Example Usage Scenario 2

- Cloud carriers provide the connectivity and transport of cloud services from cloud providers to cloud consumers.
- A cloud provider participates in and arranges for two unique service level agreements (SLAs), one with a cloud carrier (e.g. SLA2) and one with a cloud consumer (e.g. SLA1).

Usage Scenario for Cloud Carriers

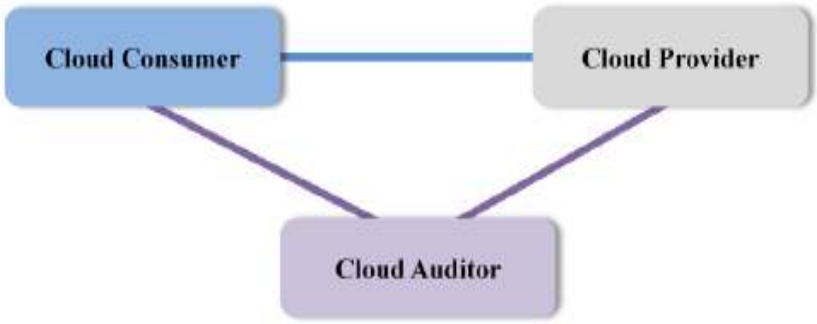
- A cloud provider arranges service level agreements (SLAs) with a cloud carrier.
- Request dedicated and encrypted connections to ensure the cloud services.



Example Usage Scenario 3

- For a cloud service, a cloud auditor conducts independent assessments of the operation and security of the cloud service implementation.

- The audit may involve interactions with both the Cloud Consumer and the Cloud Provider.

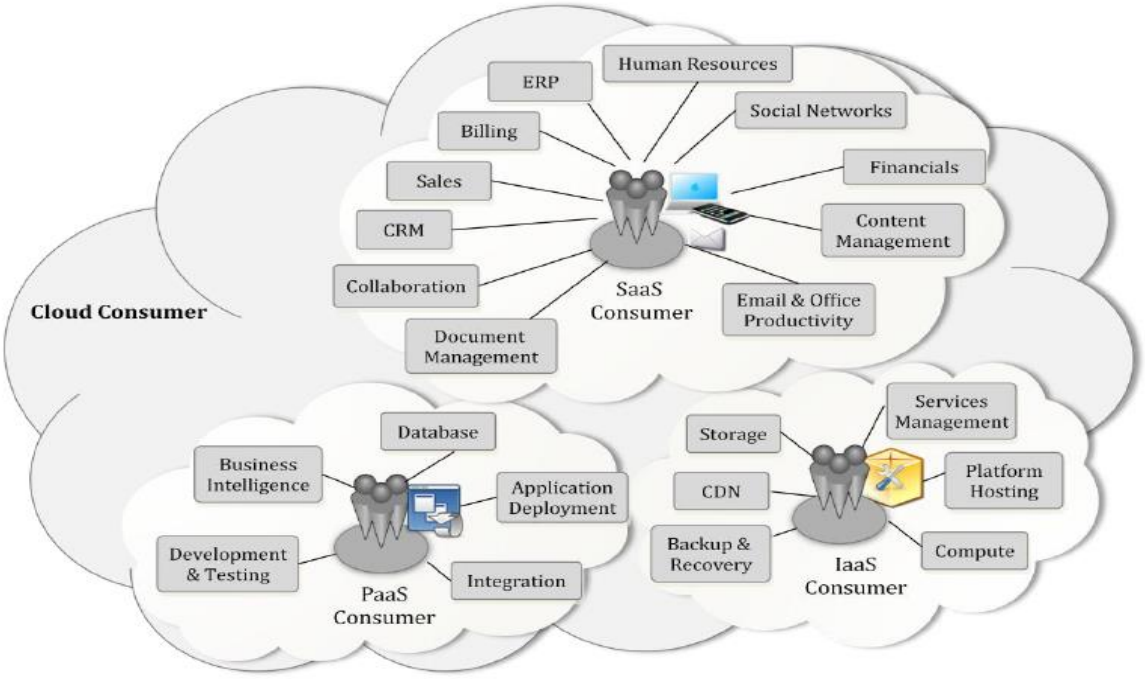


Cloud Consumer

- The cloud consumer is the principal stakeholder for the cloud computing service.
- A cloud consumer represents a person or organization that maintains a business relationship with, and uses the service from a cloud provider.

The cloud consumer may be billed for the service provisioned, and needs to arrange payments accordingly.

Example Services Available to a Cloud Consumer



- The consumers of SaaS can be organizations that provide their members with access to software applications, end users or software application administrators.
- SaaS consumers can be billed based on the number of end users, the time of use, the network bandwidth consumed, the amount of data stored or duration of stored data.

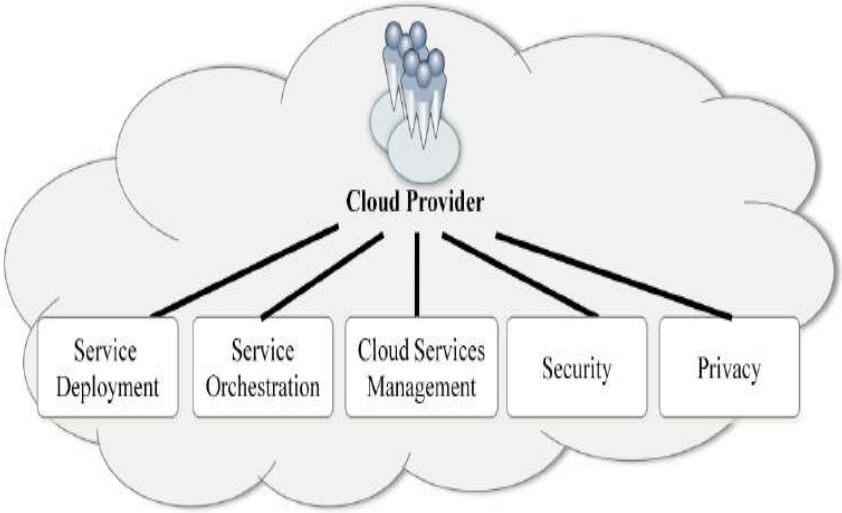
- Cloud consumers of PaaScan employ the tools and execution resources provided by cloud providers to develop, test, deploy and manage the applications.
- PaaS consumers can be application developers or application testers who run and test applications in cloud-based environments,.
- PaaS consumers can be billed according to, processing, database storage and network resources consumed.
- Consumers of IaaS have access to virtual computers, network-accessible storage & network infrastructure components.
- The consumers of IaaS can be system developers, system administrators and IT managers.
- IaaS consumers are billed according to the amount or duration of the resources consumed, such as CPU hours used by virtual computers, volume and duration of data stored.

Cloud Provider

- A cloud provider is a person, an organization;
- It is the entity responsible for making a service available to interested parties.
- A Cloud Provider acquires and manages the computing infrastructure required for providing the services.
- Runs the cloud software that provides the services.

Makes arrangement to deliver the cloud services to the Cloud Consumers through network access.

Cloud Provider - Major Activities



Cloud Auditor

- A cloud auditor is a party that can perform an independent examination of cloud service controls.
- Audits are performed to verify conformance to standards through review of objective evidence.
- A cloud auditor can evaluate the services provided by a cloud provider in terms of security controls, privacy impact, performance, etc.

Cloud Broker

- Integration of cloud services can be too complex for cloud consumers to manage.
- A cloud consumer may request cloud services from a cloud broker, instead of contacting a cloud provider directly.
- A cloud broker is an entity that manages the use, performance and delivery of cloud services. Negotiates relationships between cloud providers and cloud consumers.

Services of cloud broker

Service Intermediation:

- A cloud broker enhances a given service by improving some specific capability and providing value-added services to cloud consumers.

Service Aggregation:

- A cloud broker combines and integrates multiple services into one or more new services.
- The broker provides data integration and ensures the secure data movement between the cloud consumer and multiple cloud providers.

Services of cloud broker

Service Arbitrage:

- Service arbitrage is similar to service aggregation except that the services being aggregated are not fixed.
- Service arbitrage means a broker has the flexibility to choose services from multiple agencies.

Eg: The cloud broker can use a credit-scoring service to measure and select an agency with the best score.

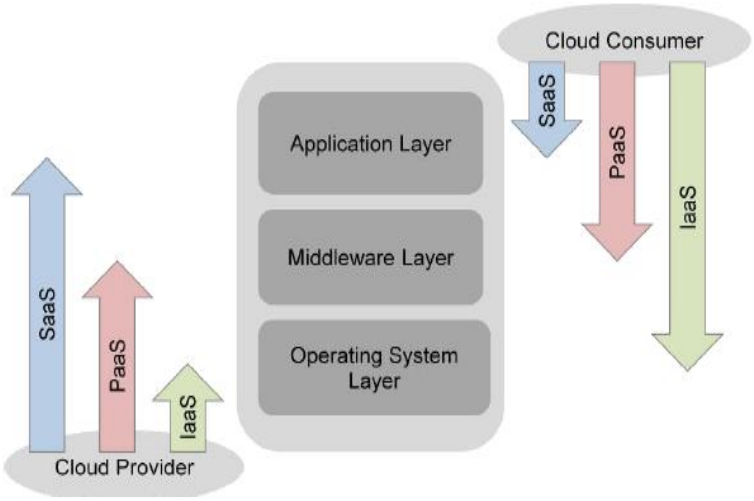
Cloud Carrier

- 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.
- The distribution of cloud services is normally provided by network and telecommunication carriers or a *transport agent*
- A transport agent refers to a business organization that provides physical transport of storage media such as high-capacity hard drives and other access devices.

Scope of Control between Provider and Consumer

The Cloud Provider and Cloud Consumer share the control of resources in a cloud system



- The application layer includes software applications targeted at end users or programs.

The applications are used by SaaS consumers, or installed/managed/maintained by PaaS consumers, IaaS consumers and SaaS providers.

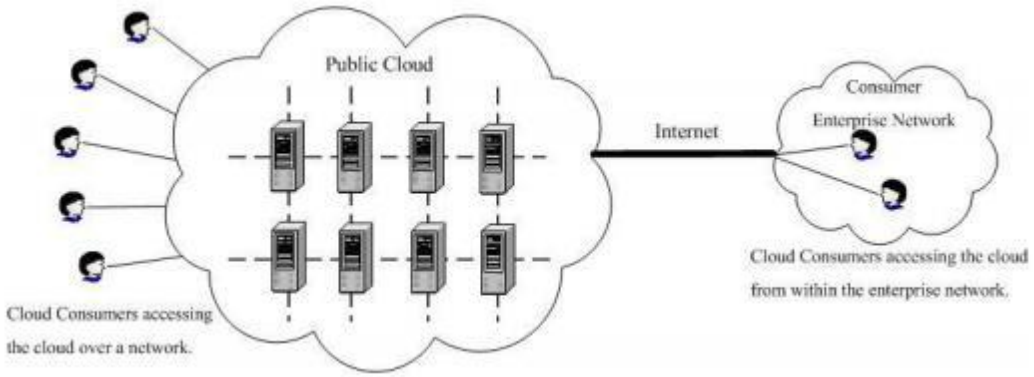
- The middleware layer provides software building blocks (e.g., libraries, database, and Java virtual machine) for developing application software in the cloud.
- Used by PaaS consumers, installed/ managed/ maintained by IaaS consumers or PaaS providers, and hidden from SaaS consumers.
- The OS layer includes operating system and drivers, and is hidden from SaaS consumers and PaaS consumers.
- An IaaS cloud allows one or multiple guest OS to run virtualized on a single physical host.

The IaaS consumers should assume full responsibility for the guest OS, while the IaaS provider controls the host OS,

3.3 Cloud Deployment Model

- Public Cloud
- Private Cloud
- Hybrid Cloud
- Community Cloud

3.3.1Public 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 meant to serve a multitude(huge number) of users, not a single customer.
- A fundamental characteristic of public clouds is multitenancy.
- Multitenancy allows multiple users to work in a software environment at the same time, each with their own resources.
- Built over the Internet (i.e., service provider offers resources, applications storage to the customers over the internet) and can be accessed by any user.
- Owned by service providers and are accessible through a subscription.
- Best Option for small enterprises, which are able to start their businesses without large up-front(initial) investment.
- By renting the services, customers were able to dynamically upsize or downsize their IT according to the demands of their business.
- Services are offered on a price-per-use basis.
- Promotes standardization, preserve capital investment
- Public clouds have geographically dispersed datacenters to share the load of users and better serve them according to their locations
- Provider is in control of the infrastructure

Examples:

o Amazon EC2 is a public cloud that provides Infrastructure as a Service

o Google AppEngine is a public cloud that provides Platform as a Service

o SalesForce.com is a public cloud that provides software as a service.

Advantage

- **Offers unlimited scalability** – on demand resources are available to meet your business needs.
- **Lower costs**—no need to purchase hardware or software and you pay only for the service you use.
- **No maintenance -** Service provider provides the maintenance.
- **Offers reliability:** Vast number of resources are available so failure of a system will not interrupt service.
- Services like SaaS, PaaS, IaaS are easily available on Public Cloud platform as it can be accessed from anywhere through any Internet enabled devices.
- **Location independent** – the services can be accessed from any location

Disadvantage

- No control over privacy or security
- Cannot be used for use of sensitive applications(Government and Military agencies will not consider Public cloud)
- Lacks complete flexibility(since dependent on provider)
- No stringent (strict) protocols regarding data management

3.3.2Private Cloud

- Cloud services are used by a single organization, which are not exposed to the public
- Services are always maintained on a private network and the hardware and software are dedicated only to single organization
- Private cloud is physically located at
    - Organization's premises [On-site private clouds] **(or)**
    - Outsourced(Given) to a third party[Outsource private Clouds]
- It may be managed either by
- Cloud Consumer organization (or)
    - By a third party
- Private clouds are used by

- government agencies
- financial institutions
- Mid size to large-size organisations.

- On-site private clouds

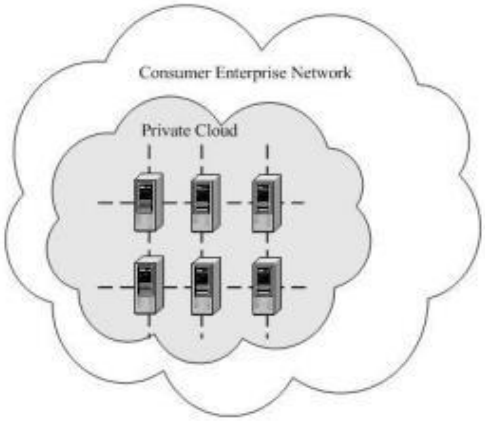
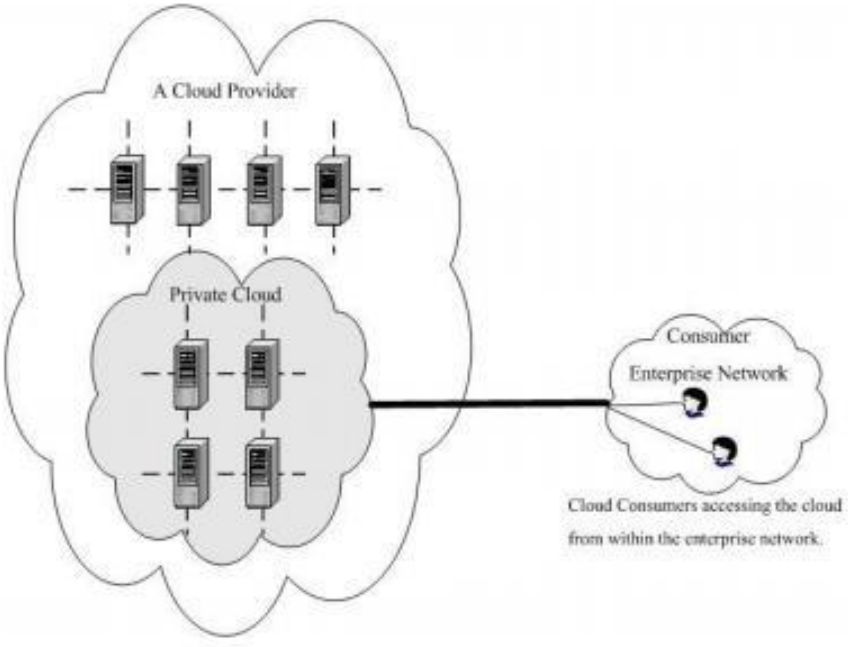


Fig: On-site private clouds

Out-sourced Private Cloud

- Supposed to deliver more efficient and convenient cloud



- Offers higher efficiency, resiliency(to recover quickly), security, and privacy
- **Customer information protection**: In-house security is easier to maintain and rely on.

- Follows its own(private organization) standard procedures and operations(where as in public cloud standard procedures and operations of service providers are followed)

Advantage

- Offers greater Security and Privacy
- Organization has control over resources
- Highly reliable
- Saves money by virtualizing the resources

Disadvantage

- Expensive when compared to public cloud
- Requires IT Expertise to maintain resources.

3.3.3Hybrid Cloud

- Built with both public and private clouds
- It is a heterogeneous cloud resulting from a private and public clouds.
- Private cloud are used for
  - sensitive applications are kept inside the organization's network
  - business-critical operations like financial reporting
- Public Cloud are used when
  - Other services are kept outside the organization's network
  - high-volume of data
  - Lower-security needs such as web-based email(gmail,yahoomail etc)
- The resources or services are temporarily leased for the time required and then released. This practice is also known as **cloud bursting.**

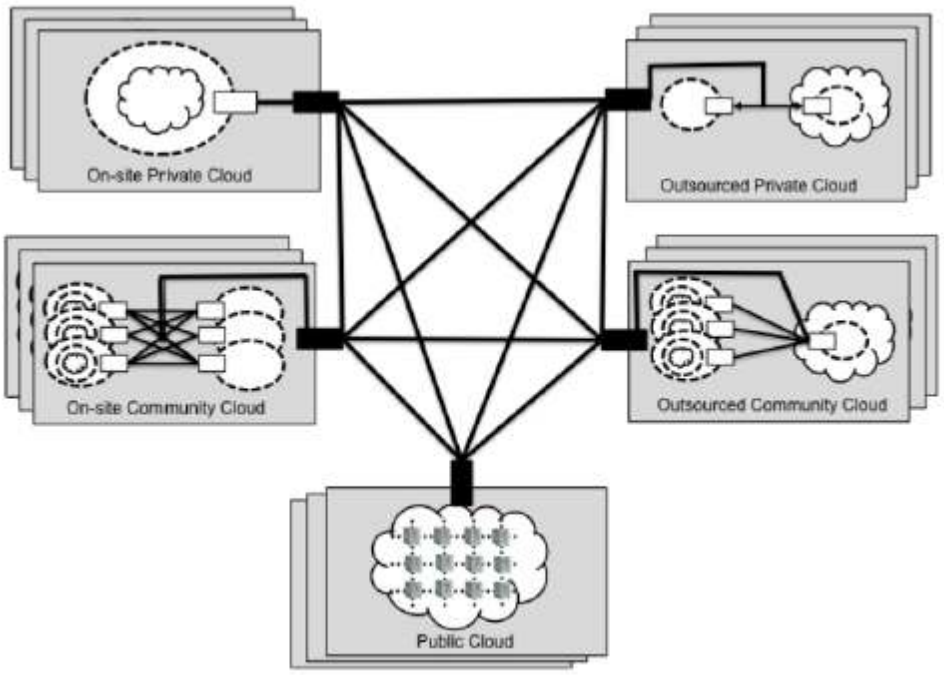


Fig:Hybrid Cloud

Advantage

- It is scalable
- Offers better security
- Flexible-Additional resources are availed in public cloud when needed
- Cost-effectiveness—we have to pay for extra resources only when needed.
- Control - Organisation can maintain a private infrastructure for sensitive application

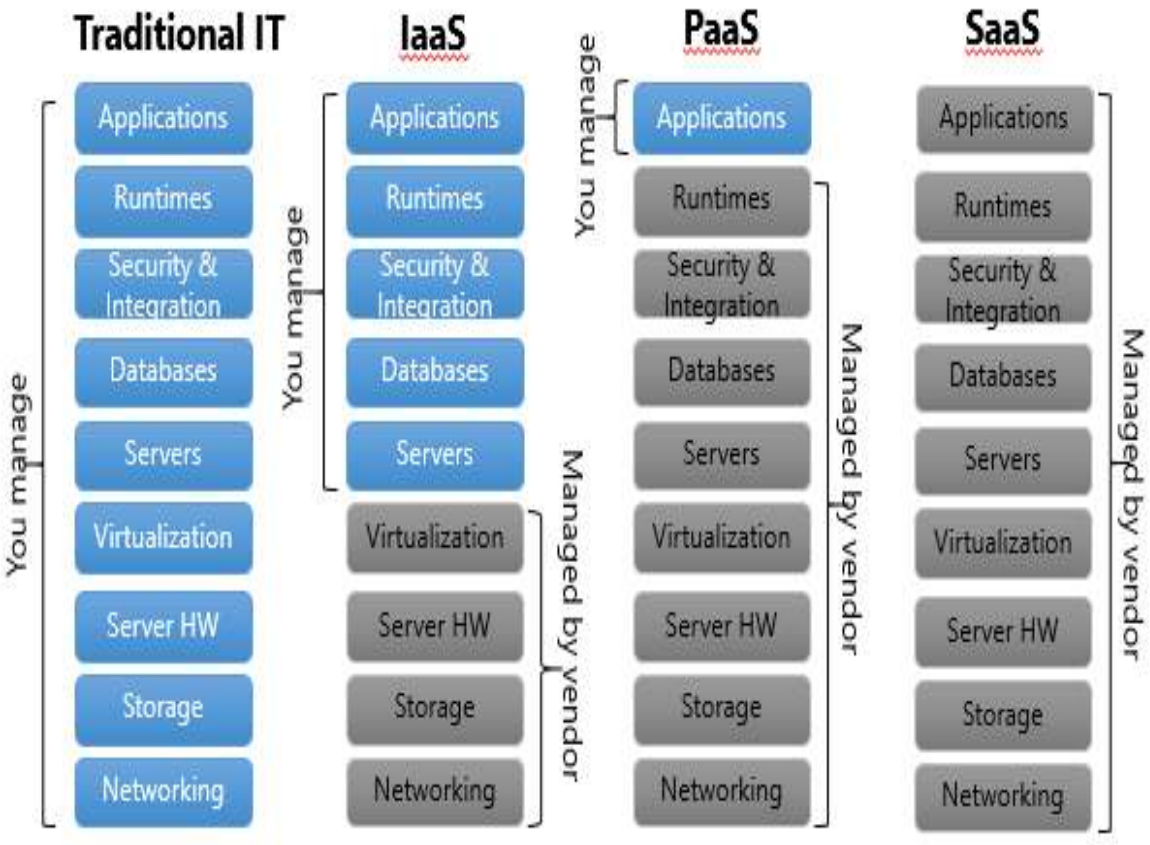
Disadvantage

- Infrastructure Dependency
- Possibility of security breach(violate) through public cloud

Difference	Public	Private	Hybrid
Tenancy	**Multi-tenancy:** the data of multiple organizations in stored in a shared environment.	**Single tenancy:** Single organizations data is stored in the cloud.	□ Data stored in the public cloud is multi-tenant. □ Data stored in private cloud is Single Tenancy.
Exposed to the Public	Yes: anyone can use the public cloud services.	No: Only the organization itself can use the private cloud services.	□ Services on private cloud can be accessed only by the organization's users □ Services on public cloud can be Accessed by anyone.
Data Center Location	Anywhere on the Internet	Inside the organization's network.	□ Private Cloud- Present in organization's network. □ Public Cloud - anywhere on the Internet.
Cloud Service Management	Cloud service provider manages the services.	Organization has their own administrators managing services	□ Organization manages the private cloud. □ Cloud Service Provider(CSP) manages the public cloud.
Hardware Components	CSP provides all the hardware.	Organization provides hardware.	□ Private Cloud – organization provides resources. □ Public Cloud – Cloud service Provider provides.
Expenses	Less Cost	Expensive when compared to public cloud	Cost required for setting up private cloud.

3.4 Cloud Service Models

- **S**oftware as a Service (**S**aaS)
- **P**latform as a Service (**P**aaS)
- **I**nfrastructure as a Service (**I**aaS)



These models are offered based on various SLAs between providers and users

 SLA of cloud computing covers

o service availability

o performance

- data protection

o Security

3.4.1 Software as a Service(SaaS)(Complete software offering on the cloud)

- SaaS is a licensed software offering on the cloud and pay per use
- SaaS is a software delivery methodology that provides licensed multi-tenant access to software and its functions remotely as a Web-based service. Usually billed based on usage
    - Usually multi tenant environment

- Highly scalable architecture

- Customers do not invest on software application programs.
- 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 a thin client interface such as a web browser (e.g., web-based email).
- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, data or even individual application capabilities, with the possible exception of limited user specific application configuration settings.
- On the customer side, there is no upfront investment in servers or software licensing.
- It is a "one-to-many" software delivery model, whereby an application is shared across multiple users
- Characteristic of Application Service Provider(ASP)

  o Product sold to customer is application access.

  o Application is centrally managed by Service Provider.

  o Service delivered is one-to-many customers

  o Services are delivered on the contract

  □ E.g. Gmail and docs, Microsoft SharePoint, and the CRM software(Customer Relationship management)
- **SaaS providers**
- Google's Gmail, Docs, Talk etc
- Microsoft's Hotmail, Sharepoint
- SalesForce,
- Yahoo
- Facebook

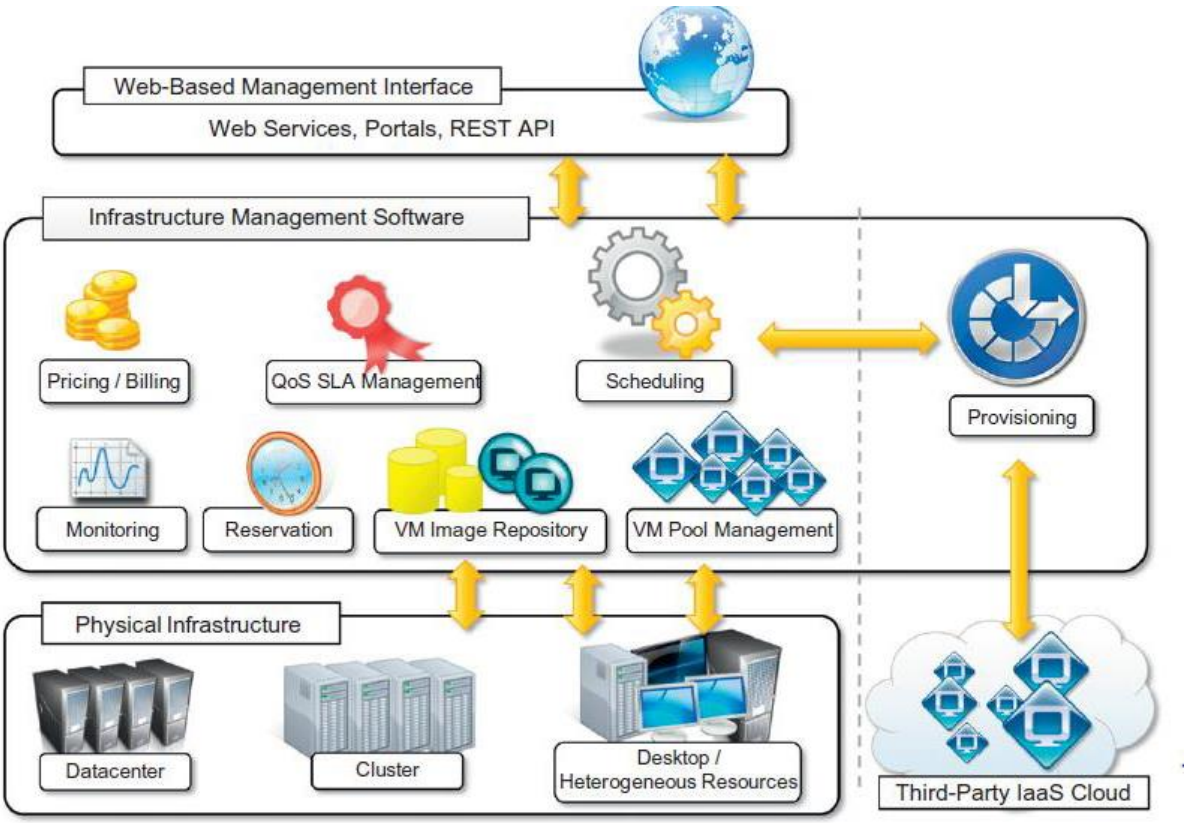
3.4.2Infrastructure as a Service (IaaS) (Hardware offerings on the cloud)

IaaS is the delivery of technology infrastructure (mostly hardware) as an on demand, scalable service .

- Usually billed based on usage
- Usually multi tenant virtualized environment
- Can be coupled with Managed Services for OS and application support
- User can choose his OS, storage, deployed app, networking components

- The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources.
- Consumer is able to deploy and run arbitrary software, which may 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.

- IaaS/HaaS solutions bring all the benefits of hardware virtualization: workload partitioning, application isolation, sandboxing, and hardware tuning
- **Sandboxing:** A program is set aside from other programs in a separate environment so that if errors or security issues occur, those issues will not spread to other areas on the computer.
- **Hardware tuning:** To improve the performance of system
- The user works on multiple VMs running guest OSes
- the service is performed by rented cloud infrastructure
- The user does not manage or control the cloud infrastructure, but can specify when to request and release the needed resources.



IaaS providers

- Amazon Elastic Compute Cloud (EC2)
  - Each instance provides 1-20 processors, upto 16 GB RAM, 1.69TB storage
- RackSpace Hosting
  - Each instance provides 4 core CPU, upto 8 GB RAM, 480 GB storage
- Joyent Cloud
  - Each instance provides 8 CPUs, upto 32 GB RAM, 48 GB storage
- Go Grid
  - Each instance provides 1-6 processors, upto 15 GB RAM, 1.69TB storage

3.4.3 Platform as a Service (PaaS) (Development platform)

- PaaS provides all of the facilities required to support the complete life cycle of building, delivering and deploying web applications and services entirely from the Internet.
- Typically applications must be developed with a particular platform in mind
  - Multi tenant environments
  - Highly scalable multi tier architecture
- The capability provided to the consumer is to deploy onto the cloud infrastructure consumer created or acquired applications created using programming languages and tools supported by the provider.
- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage.

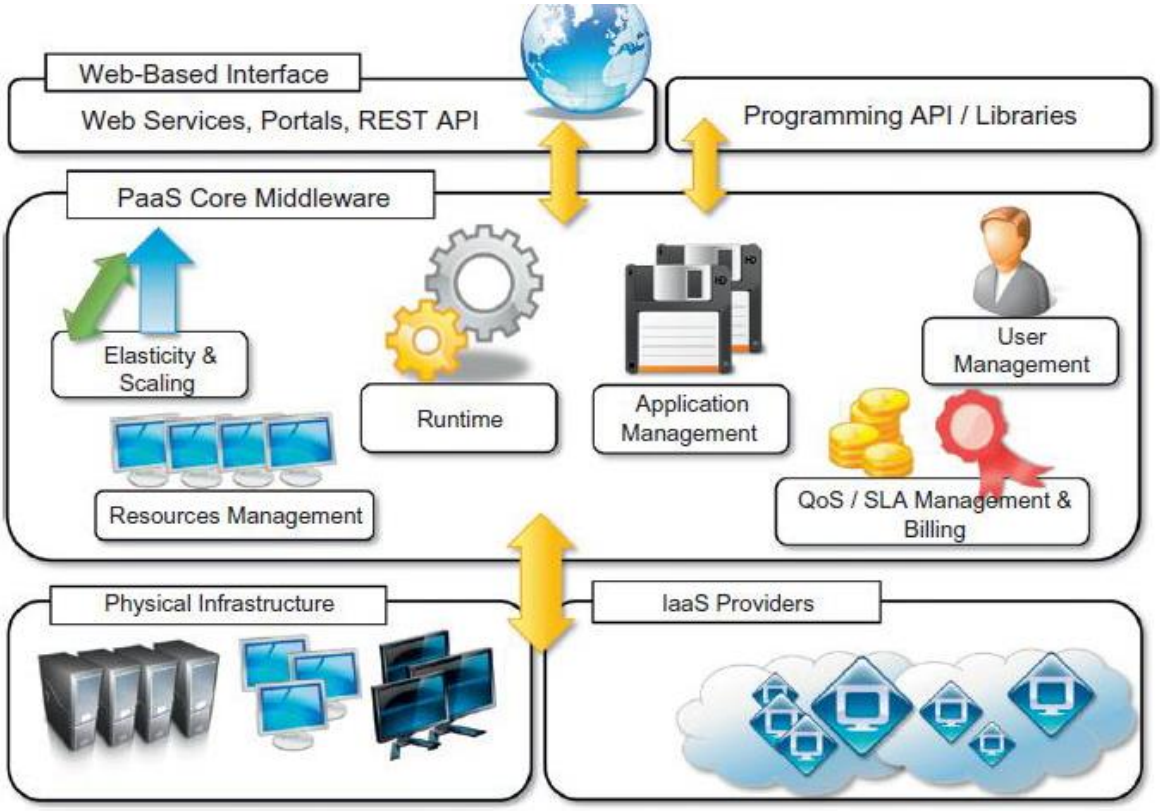
Have control over the deployed applications and possibly application hosting environment configurations.

Customers are provided with execution platform for developing applications.

 Execution platform includes operating system, programming language execution environment, database, web server, hardware etc.

 This acts as **middleware** on top of which applications are built

 The user is freed from managing the cloud infrastructure



Application management is the core functionality of the middleware

 Provides runtime(execution) environment

 Developers design their applications in the execution environment.

 Developers need not concern about hardware (physical or virtual), operating systems, and other resources.

 PaaS core middleware manages the resources and scaling of applications on demand.

 PaaS offers

o Execution environment and hardware resources (infrastructure) **(or)**

o software is installed on the user premises

 PaaS: Service Provider provides Execution environment and hardware resources (infrastructure)

Characteristics of PaaS

 Runtime framework: Executes end-user code according to the policies set by the user and the provider.

 Abstraction: PaaS helps to deploy(install) and manage applications on the cloud.

 Automation: Automates the process of deploying applications to the infrastructure, additional resources are provided when needed.

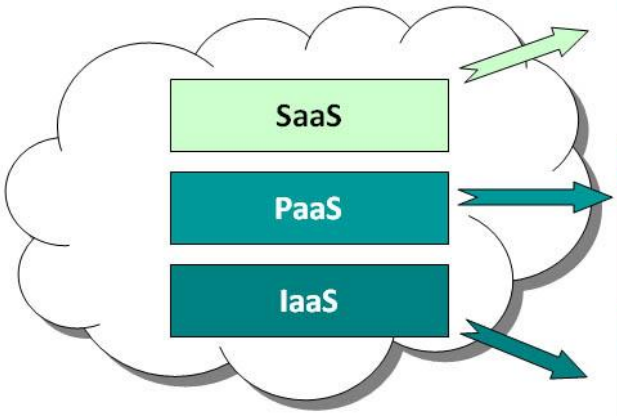
 Cloud services: helps the developers to simplify the creation and delivery cloud applications.

PaaS providers

- Google App Engine
  - Python, Java, Eclipse
- Microsoft Azure
  - .Net, Visual Studio
- Sales Force
  - Apex, Web wizard
- TIBCO,
- VMware,
- Zoho

Cloud Computing – Services

- Software as a Service - SaaS
- Platform as a Service - PaaS
- Infrastructure as a Service - IaaS



Who Uses It	What Services are available	Why use it?
Business Users	EMail, Office Automation, CRM, Website Testing, Wiki, Blog, Virtual Desktop ...	To complete business tasks
Developers and Deployers	Service and application test, development, integration and deployment	Create or deploy applications and services for users
System Managers	Virtual machines, operating systems, message queues, networks, storage, CPU, memory, backup services	Create platforms for service and application test, development, integration and deployment

Category	Description	Product Type	Vendors and Products
PaaS-I	Execution platform is provided along with hardware resources (infrastructure)	Middleware + Infrastructure	Force.com, Longjump
PaaS -II	Execution platform is provided with additional components	Middleware + Infrastructure, Middleware	Google App Engine
PaaS- III	Runtime environment for developing any kind of application development	Middleware + Infrastructure, Middleware	Microsoft Azure

3.5 Architectural Design Challenges

Challenge 1 : Service Availability and Data Lock-in Problem

Service Availability

- Service Availability in Cloud might be affected because of
- Single Point Failure
- Distributed Denial of Service
- Single Point Failure
  - o Depending on single service provider might result in failure.
  - o In case of single service providers, even if company has multiple data centres located in different geographic regions, it may have **common software infrastructure and accounting systems**.

Solution:

o Multiple cloud providers may provide more protection from failures and they provide High Availability(HA)

o Multiple cloud Providers will rescue the loss of all data.

Distributed Denial of service (DDoS) attacks.

o Cyber criminals, attack target websites and online services and makes services unavailable to users.

o DDoS tries to overwhelm (disturb) the services unavailable to user by having more traffic than the server or network can accommodate.

Solution:

o Some SaaS providers provide the opportunity to defend against DDoS attacks by using quick scale-ups.

Customers cannot easily extract their data and programs from one site to run on another.

Solution:

o Have standardization among service providers so that customers can deploy (install) services and data across multiple cloud providers.

Data Lock-in

 It is a situation in which a customer using service of a provider cannot be moved to another service provider because technologies used by a provider will be incompatible with other providers.

 This makes a customer dependent on a vendor for services and makes customer unable to use service of another vendor.

Solution:

o Have standardization (in technologies) among service providers so that customers can easily move from a service provider to another.

Challenge 2: Data Privacy and Security Concerns

 Cloud services are prone to attacks because they are accessed through internet.

Security is given by

o Storing the encrypted data in to cloud.

o Firewalls, filters.

 Cloud environment attacks include

o Guest hopping

o Hijacking

o VM rootkits.

 Guest Hopping: Virtual machine hyper jumping (VM jumping) is an attack method that exploits(make use of) hypervisor's weakness that allows a virtual machine (VM) to be accessed from another.

 Hijacking: Hijacking is a type of network security attack in which the attacker takes control of a communication

 VM Rootkit: is a collection of malicious (harmful) computer software, designed to enable access to a computer that is not otherwise allowed.

 A **man-in-the-middle (MITM)** attack is a form of eavesdroppping(Spy) where communication between two users is monitored and modified by an unauthorized party.

o Man-in-the-middle attack may take place **during VM migrations** [virtual machine (VM) migration - VM is moved from one physical host to another host].

 Passive attacks steal sensitive data or passwords.

 Active attacks may manipulate (control) kernel data structures which will cause major damage to cloud servers.

Challenge 3: Unpredictable Performance and Bottlenecks

 Multiple VMs can share CPUs and main memory in cloud computing, but I/O sharing is problematic.

 Internet applications continue to become more data-intensive (handles huge amount of data).

 Handling huge amount of data (data intensive) is a bottleneck in cloud environment.

 Weak Servers that does not provide data transfers properly must be removed from cloud environment

Challenge 4: Distributed Storage and Widespread Software Bugs

 The database is always growing in cloud applications.

 There is a need to create a storage system that meets this growth.

 This demands the design of efficient distributed SANs (Storage Area Network of Storage devices).

 Data centres must meet

o Scalability

o Data durability

o HA(High Availability)

o Data consistence

 Bug refers to errors in software.

 Debugging must be done in data centres.

Challenge 5: Cloud Scalability, Interoperability and Standardization

Cloud Scalability

 Cloud resources are scalable. Cost increases when storage and network bandwidth scaled(increased)

Interoperability

 Open Virtualization Format (OVF) describes an open, secure, portable, efficient, and extensible format for the packaging and distribution of VMs.

 OVF defines a transport mechanism for VM, that can be applied to different virtualization platforms

Standardization

 Cloud standardization, should have ability for virtual machine to run on any virtual platform.

Challenge 6: Software Licensing and Reputation Sharing

 Cloud providers can use both pay-for-use and bulk-use licensing schemes to widen the business coverage.

 Cloud providers must create reputation-guarding services similar to the "trusted e-mail" services

 Cloud providers want legal liability to remain with the customer, and vice versa.

3.6. Cloud Storage

 Storing your data on the storage of a cloud service provider rather than on a local system.

 Data stored on the cloud are accessed through Internet.

 Cloud Service Provider provides Storage as a Service

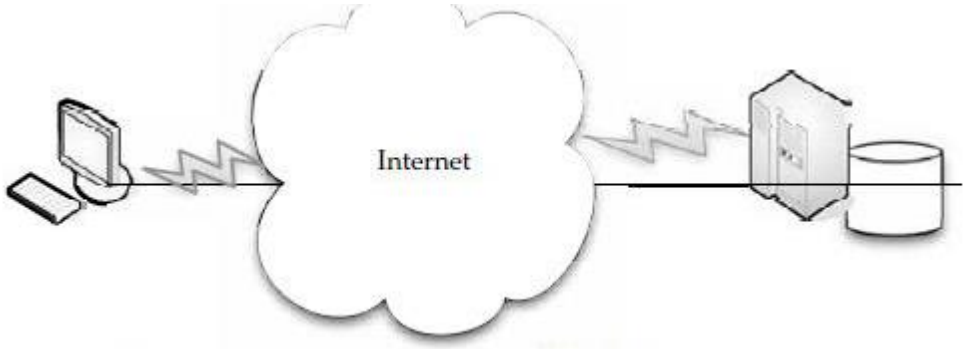
3.6.1 Storage as a Service

- Third-party provider rents space on their storage to cloud users.
- Customers move to cloud storage when they lack in budget for having their own storage.
- Storage service providers takes the responsibility of taking current backup, replication, and disaster recovery needs.
- Small and medium-sized businesses can make use of Cloud Storage
- Storage is rented from the provider using a

o cost-per-gigabyte-stored **(or)**

o cost-per-data-transferred

- The end user doesn't have to pay for infrastructure (resources), they have to pay only for how much they transfer and save on the provider's storage.



Clients rent storage capacity from cloud storage vendors.

5.2 Providers

- Google Docs allows users to upload documents, spreadsheets, and presentations to Google's data servers.
- Those files can then be edited using a Google application.
- Web email providers like Gmail, Hotmail, and Yahoo! Mail, store email messages on their own servers.
- Users can access their email from computers and other devices connected to the Internet.
- Flicker and Picasa host millions of digital photographs, Users can create their own online photo albums.
- YouTube hosts millions of user-uploaded video files.
- Hostmonster and GoDaddy store files and data for many client web sites.
- Facebook and MySpace are social networking sites and allow members to post pictures and other content. That content is stored on the company's servers.
- MediaMax and Strongspace offer storage space for any kind of digital data.

3.6.2 Data Security

- To secure data, most systems use a combination of techniques:

o Encryption

o Authentication

o Authorization

Encryption

o Algorithms are used to encode information. To decode the information keys are required.

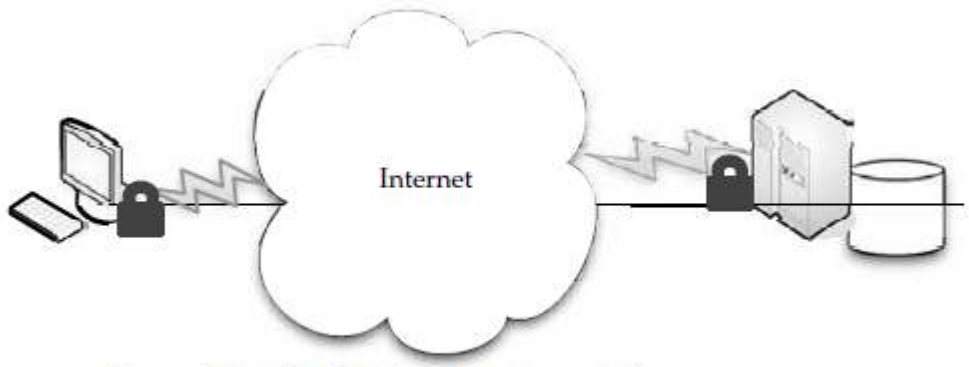
Authentication processes

o This requires a user to create a name and password.

Authorization practices

o The client lists the people who are authorized to access information stored on the cloud system.

If information stored on the cloud, the head of the IT department might have complete and free access to everything.



Encryption and authentication are two security measures you can use to keep your data safe on a cloud storage provider.

Reliability

- Service Providers gives reliability for data through redundancy (maintaining multiple copies of data).

Reputation is important to cloud storage providers. If there is a perception that the provider is unreliable, they won't have many clients.

Advantages

- Cloud storage providers balance server loads.
- Move data among various datacenters, ensuring that information is stored close and thereby available quickly to where it is used.
- It allows to protect the data in case there's a disaster.
- Some products are agent-based and the application automatically transfers information to the cloud via FTP

Cautions

- Don't commit everything to the cloud, but use it for a few, noncritical purposes.
- Large enterprises might have difficulty with vendors like Google or Amazon.
- Forced to rewrite solutions for their applications.

- Lack of portability.

Theft (Disadvantage)

- User data could be stolen or viewed by those who are not authorized to see it.
- Whenever user data is let out of their own datacenter, risk trouble occurs from a security point of view.
- If user store data on the cloud, make sure user encrypts data and secures data transit with technologies like SSL.

3.7 Cloud Storage Providers

Amazon Simple Storage Service (S3)

- The best-known cloud storage service is Amazon's Simple Storage Service (S3), launched in 2006.
- Amazon S3 is designed to make computing easier for developers.
- Amazon S3 provides an interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the Web.
- Amazon S3 is intentionally built with a minimal feature set that includes the following functionality:

  • Write, read, and delete objects containing from 1 byte to 5 gigabytes of data each.

  The number of objects that can be stored is unlimited.

  • Each object is stored and retrieved via a unique developer-assigned key.

  • Objects can be made private or public, and rights can be assigned to specific users.

  • Uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.

Design Requirements

Amazon built S3 to fulfill the following design requirements:

• **Scalable** Amazon S3 can scale in terms of storage, request rate, and users to support an unlimited number of web-scale applications.

- **Reliable** Store data durably, with 99.99 percent availability. Amazon says it does not allow any downtime.

• **Fast** Amazon S3 was designed to be fast enough to support high-performance applications. Server-side latency must be insignificant relative to Internet latency. Any performance bottlenecks can be fixed by simply adding nodes to the system.

• **Inexpensive** Amazon S3 is built from inexpensive commodity hardware components. As a result, frequent node failure is the norm and must not affect the overall system. It must be hardware-agnostic, so that savings can be captured as Amazon continues to drive down infrastructure costs.

• **Simple** Building highly scalable, reliable, fast, and inexpensive storage is difficult. Doing so in a way that makes it easy to use for any application anywhere is more difficult. Amazon S3 must do both.

Design Principles

Amazon used the following principles of distributed system design to meet Amazon S3 requirements:

• **Decentralization** It uses fully decentralized techniques to remove scaling bottlenecks and single points of failure.

• **Autonomy** The system is designed such that individual components can make decisions based on local information.

• **Local responsibility** Each individual component is responsible for achieving its consistency; this is never the burden of its peers.

• **Controlled concurrency** Operations are designed such that no or limited concurrency control is required.

• **Failure toleration** The system considers the failure of components to be a normal mode of operation and continues operation with no or minimal interruption.

• **Controlled parallelism** Abstractions used in the system are of such granularity that parallelism can be used to improve performance and robustness of recovery or the introduction of new nodes.

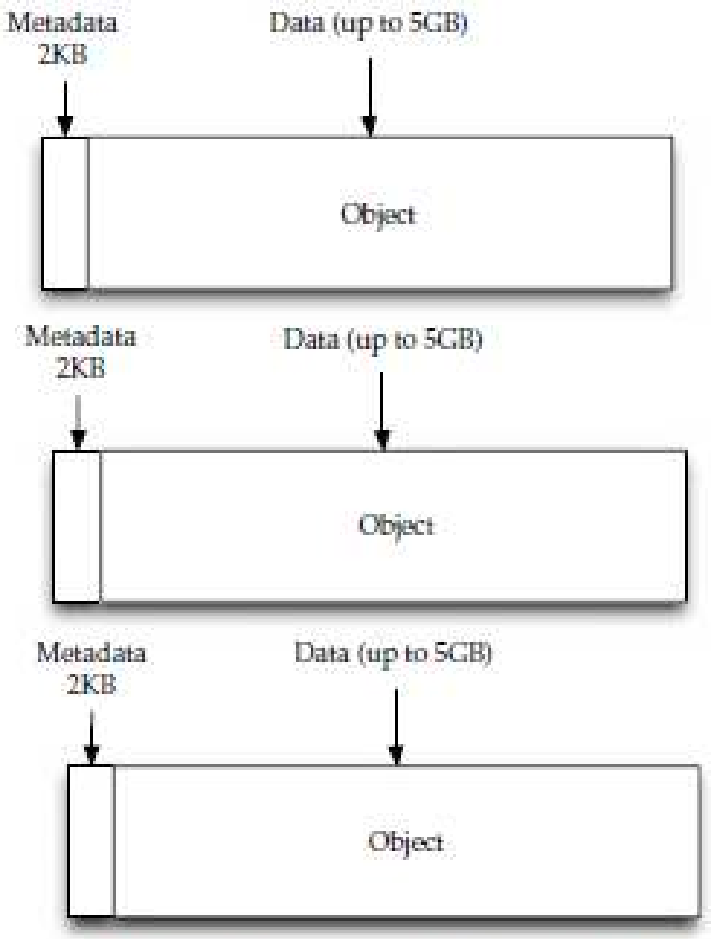
• **Small, well-understood building blocks** Do not try to provide a single service that does everything for everyone, but instead build small components that can be used as building blocks for other services.

• **Symmetry** Nodes in the system are identical in terms of functionality, and require no or minimal node-specific configuration to function.

• **Simplicity** The system should be made as simple as possible, but no simpler.

How S3 Works

Amazon keeps its lips pretty tight about how S3 works, but according to Amazon, S3's design aims to provide scalability, high availability, and low latency at commodity costs. S3 stores arbitrary objects at up to 5GB in size, and each is accompanied by up to 2KB of metadata. Objects are organized by *buckets*. Each bucket is owned by an AWS account and the buckets are identified by a unique, user-assigned key.



Multiple objects are stored in buckets in Amazon S3.

Buckets and objects are created, listed, and retrieved using either a REST-style or SOAP interface.

Objects can also be retrieved using the HTTP GET interface or via BitTorrent. An access control list restricts who can access the data in each bucket. Bucket names and keys are formulated so that they can be accessed using HTTP. Requests are authorized using an access control list associated with each bucket and object, for instance:

http://s3.amazonaws.com/examplebucket/examplekey

http://examplebucket.s3.amazonaws.com/examplekey

The Amazon AWS Authentication tools allow the bucket owner to create an authenticated URL with a set amount of time that the URL will be valid.

UNIT IV RESOURCE MANAGEMENT AND SECURITY IN CLOUD 10

Inter Cloud Resource Management – Resource Provisioning and Resource Provisioning Methods – Global Exchange of Cloud Resources – Security Overview – Cloud Security Challenges – Software-as-a-Service Security – Security Governance – Virtual Machine Security – IAM – Security Standards.

4.1 INTER-CLOUD RESOURCE MANAGEMENT

Cloud of Clouds (Inter cloud)

- Inter cloud or 'cloud of clouds'-refer to a theoretical model for cloud computing services.
- Combining many different individual clouds into one seamless mass in terms of on-demand operations.
- The inter cloud would simply make sure that a cloud could use resources beyond its reach.
- Taking advantage of pre-existing contracts with other cloud providers.
- Each single cloud does not have infinite physical resources or ubiquitous geographic footprint.
- A cloud may be saturated to the computational and storage resources of its infrastructure.
- It would still be able satisfy such requests for service allocations sent from its clients.
- A single cloud cannot always fulfill the requests or provide required services.
- When two or more clouds have to communicate with each other, or another intermediary comes into play and federates the resources of two or more clouds.
- In inter cloud, intermediary is known as "cloud broker" or simply "broker."
- Broker is the entity which introduces the cloud service customer (CSC) to the cloud service provider (CSP)

Inter-Cloud Resource Management Consists of

- Extended Cloud Computing Services
- Resource Provisioning and Platform Management
- Virtual Machine Creation and Management
- Global Exchange of Cloud Resources

4.1.1 Extended Cloud Computing Services

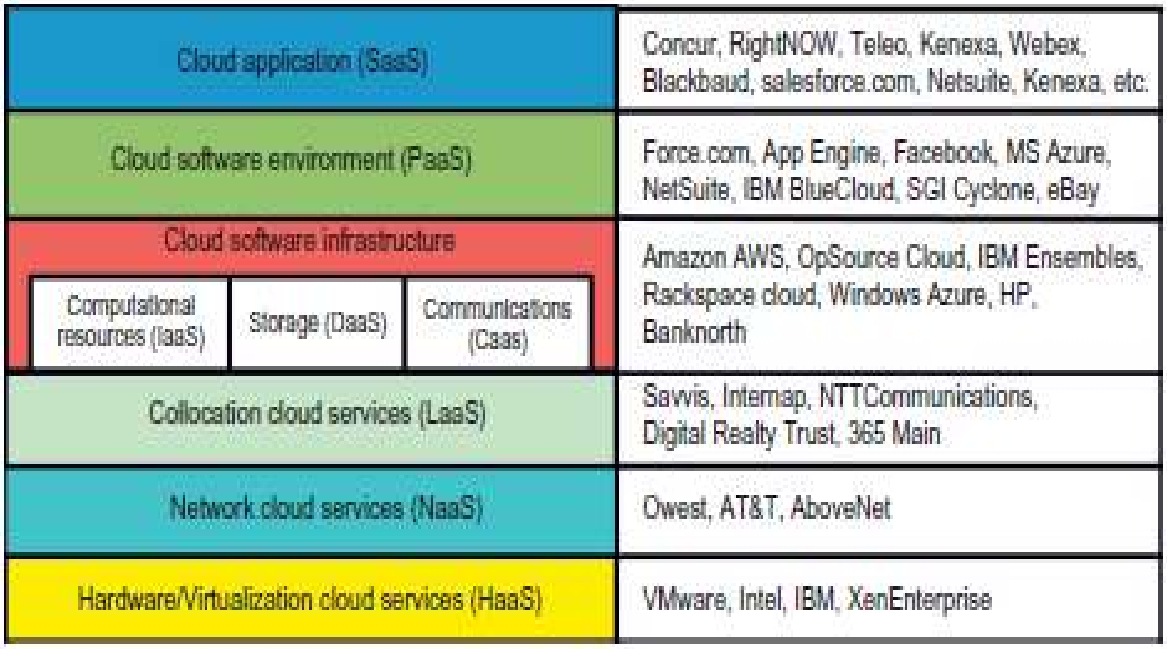


Fig: Six layers of cloud services and their providers

Six layers of cloud services

- Software as a Service(SaaS)
- Platform as a Service(PaaS)
- Infrastructure as a Service(IaaS)
- Hardware / Virtualization Cloud Services(HaaS)
- Network Cloud Services (NaaS)
- Collocation Cloud Services(LaaS)

- The top layer offers SaaS which provides cloud application.
- PaaS sits on top of IaaS infrastructure.
- The bottom three layers are more related to physical requirements.
- The bottommost layer provides Hardware as a Service (HaaS).
- NaaS is used for interconnecting all the hardware components.

- Location as a Service (LaaS), provides security to all the physical hardware and network resources. This service is also called as Security as a Service.
- The cloud infrastructure layer can be further subdivided as
  - Data as a Service (DaaS)
  - Communication as a Service (CaaS)
  - Infrastructure as a Service(IaaS)
- Cloud players are divided into three classes:
  - Cloud service providers and IT administrators
  - Software developers or vendors
  - End users or business 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	Develop and deploy software
End users or business users	To deploy and store data	To develop and test software	Use business software

Table: Cloud Differences in Perspective of Providers, Vendors, and Users

4.1.1 Cloud Service Tasks and Trends

- SaaS is mostly used for Business Applications
- Eg: CRM (Customer Relationship Management) used for business promotion, direct sales, and marketing services
- PaaS is provided by Google, Salesforce.com, and Facebook etc.
- IaaS is provided by Amazon, Windows Azure, and RackRack etc.
- Collocation services Provides security to lower layers.
- Network cloud services provide communications.

4.1.2 Software Stack for Cloud Computing

- The software stack structure of cloud computing software can be viewed as layers.
- Each layer has its own purpose and provides the interface for the upper layers.
- The lower layers are not completely transparent to the upper layers.

4.1.3 Runtime Support Services

- Runtime support refers to software needed in applications.
- The SaaS provides the software applications as a service, rather than allowing users purchase the software.
- On the customer side, there is no upfront investment in servers.

4.1.2 Resource Provisioning (Providing) and Platform Deployment

There are techniques to provision computer resources or VMs. Parallelism is exploited at the cluster node level.

4.1.2.1 Provisioning of Compute Resources (VMs)

- Providers supply cloud services by signing SLAs with end users.
- The SLAs must specify resources such as
  - CPU
  - Memory
  - Bandwidth

Users can use these for a preset (fixed) 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.
- Provisioning of resources to users is a challenging problem. The difficulty comes from the following
  - Unpredictability of consumer demand
  - Software and hardware failures
  - Heterogeneity of services

- Power management
- Conflict in signed SLAs between consumers and service providers.

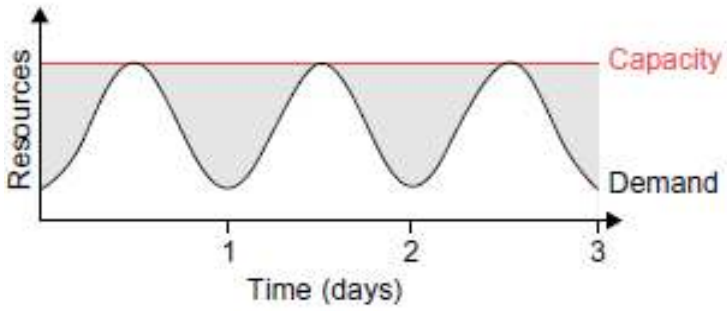
4.1.2.2 Provisioning Methods

Three cases of static cloud resource provisioning policies are considered.

Static cloud resource provisioning

case (a)

- over provisioning(Providing) with the peak load causes heavy resource waste (shaded area).

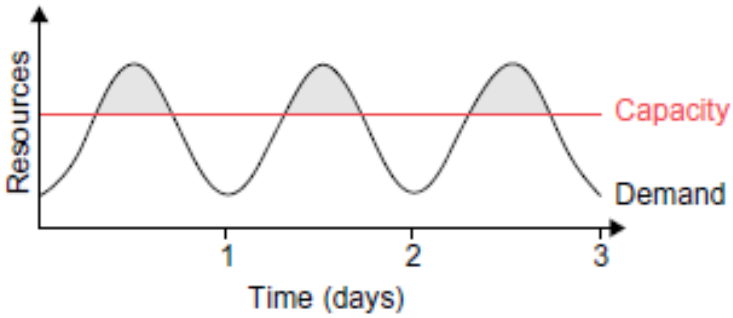


(a) Provisioning for peak load

..

case (b)

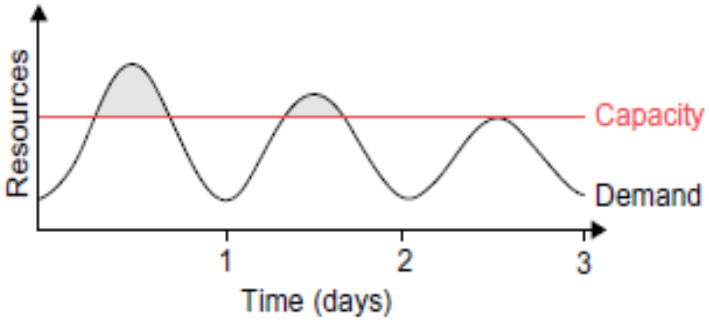
Under provisioning of resources results in losses by both user and provider. Users have paid for the demand (the shaded area above the capacity) is not used by users.



(b) Underprovisioning 1

case (c)

Declining in user demand results in worse resource waste.



(c) Underprovisioning 2

Constant provisioning

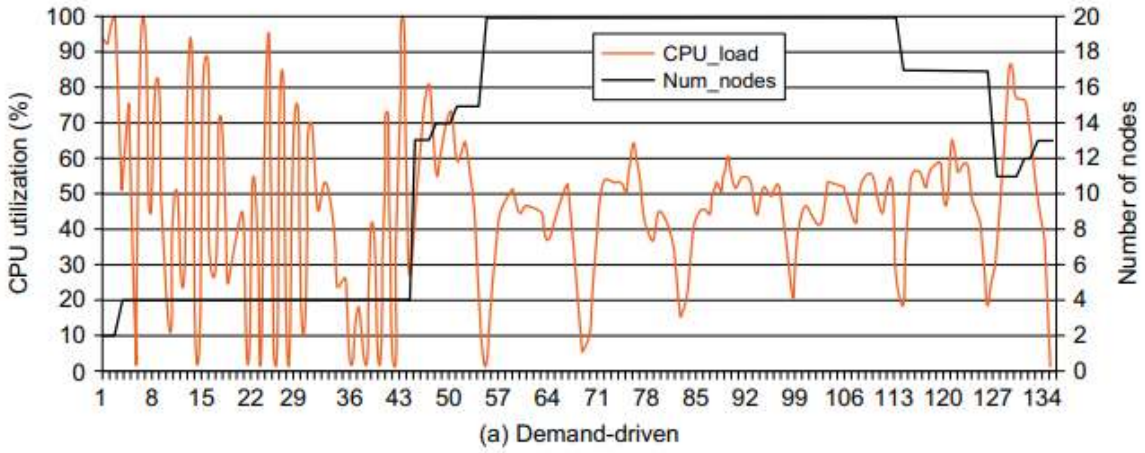
- Fixed capacity to a declining user demand could result in even worse resource waste.
- The user may give up the service by canceling the demand, resulting in reduced revenue for the provider.
- Both the user and provider may be losers in resource provisioning without elasticity.

Resource-provisioning methods are

- Demand-driven method - Provides static resources and has been used in grid computing
- Event-driven method - Based on predicted workload by time.
- Popularity-Driven Resource Provisioning – Based on Internet traffic monitored

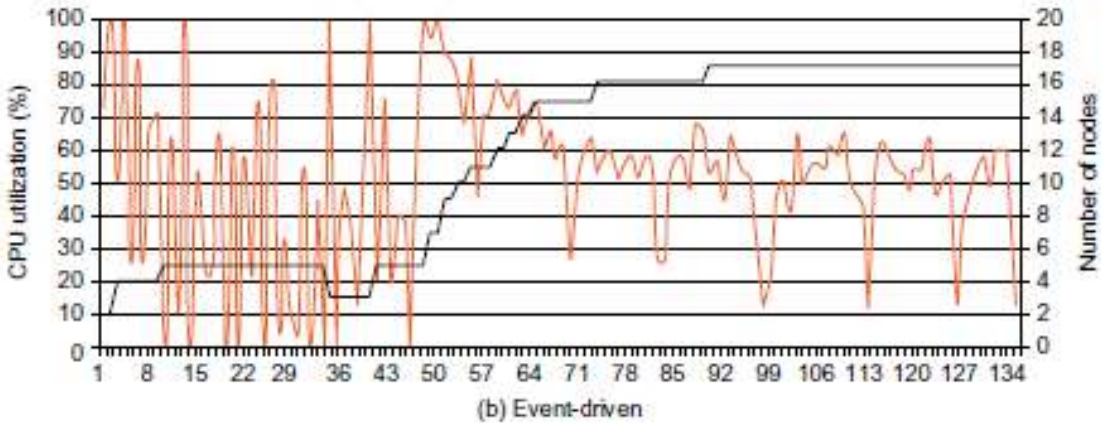
4.1.2.3 Demand Driven Methods

- Provides Static resources
- This method adds or removes nodes (VM) based on the current utilization(Use) level of the allocated resources.
- When a resource has surpassed (exceeded) a threshold (Upperlimit) for a certain amount of time, the scheme increases the resource (nodes) based on demand.
- When a resource is below a threshold for a certain amount of time, then resources could be decreased accordingly.
- This method is easy to implement.
- The scheme does not work out properly if the workload changes abruptly.



4.1.2.4 Event-Driven Resource Provisioning

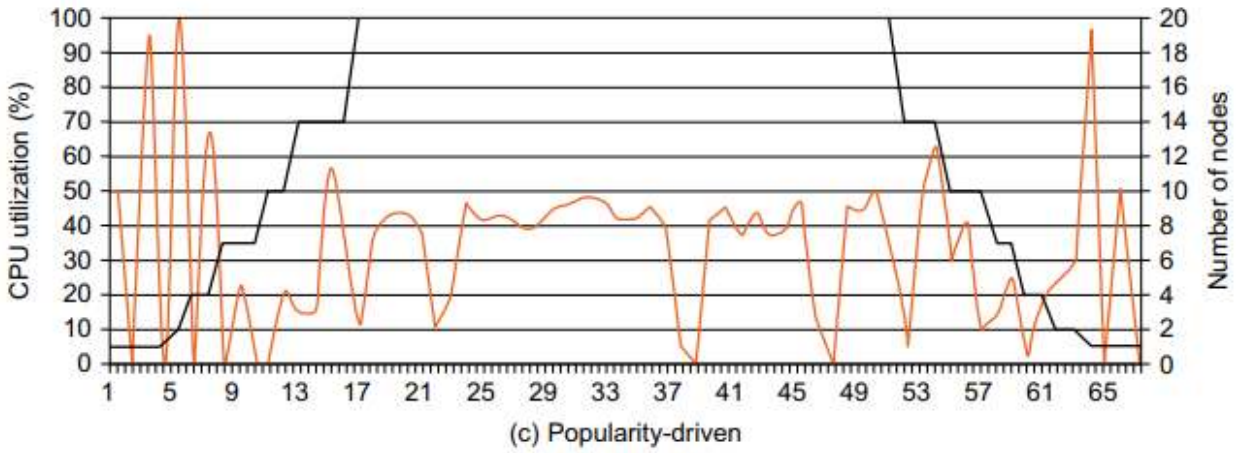
- This scheme adds or removes machine instances based on a specific time event.
- The scheme works better for seasonal or predicted events such as Christmastime in the West and the Lunar New Year in the East.
- During these events, the number of users grows before the event period and then decreases during the event period. This scheme anticipates peak traffic before it happens.
- The method results in a minimal loss of QoS, if the event is predicted correctly



4.1.2.5 Popularity-Driven Resource Provisioning

- Internet searches for popularity of certain applications and allocates resources by popularity demand.

- This scheme has a minimal loss of QoS, if the predicted popularity is correct.
- Resources may be wasted if traffic does not occur as expected.
- 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.



4.1.2.6 Dynamic Resource Deployment

- The cloud uses VMs as building blocks to create an execution environment across multiple resource sites.
- Dynamic resource deployment can be implemented to achieve scalability in performance.
- Peering arrangements established between gateways enable the allocation of resources from multiple grids to establish the execution environment.
- Dynamic resource deployment can be implemented to achieve scalability in performance.
- InterGrid is used for interconnecting distributed computing infrastructures.
- InterGrid provides an execution environment on top of the interconnected infrastructures.
- IGG(InterGridGateway) allocates resources from an
  - Organization's local cluster (Or)
  - Cloud provider.

- Under peak demands, IGG interacts with another IGG that can allocate resources from a cloud computing provider.
- Component called the DVE manager performs resource allocation and management.
- Intergrid gateway (IGG) allocates resources from a local cluster three steps:

(1) Requesting the VMs(Resources)

(2) Enacting (Validate) the leases

(3) Deploying (install) the VMs as requested.

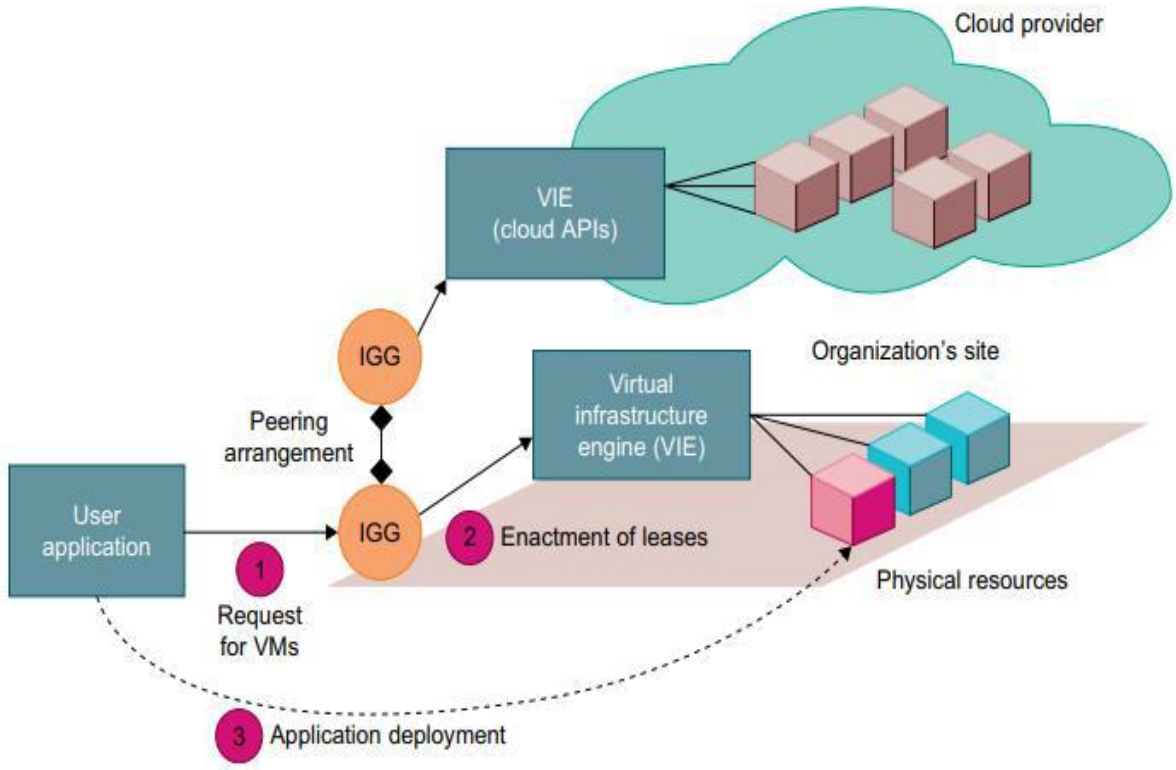


Fig: Cloud resource deployment using an IGG (intergrid gateway) to allocate the VMs from a Local cluster to interact with the IGG of a public cloud provider.

- Under peak demand, this IGG interacts with another IGG that can allocate resources from a cloud computing provider.
- A grid has predefined peering arrangements with other grids, which the IGG manages.
- Through multiple IGGs, the system coordinates the use of InterGrid resources.
- An IGG is aware of the peering terms with other grids, selects suitable grids that can provide the required resources, and replies to requests from other IGGs.
- Request redirection policies determine which peering grid InterGrid selects to process a request and a price for which that grid will perform the task.
- An IGG can also allocate resources from a cloud provider.
- The InterGrid allocates and provides a distributed virtual environment (DVE).

- This is a virtual cluster of VMs that runs isolated from other virtual clusters.
- A component called the DVE manager performs resource allocation and management on behalf of specific user applications.
- The core component of the IGG is a scheduler for implementing provisioning policies and peering with other gateways.
- The communication component provides an asynchronous message-passing mechanism.

4.1.2.7 Provisioning of Storage Resources

- Storage layer is built on top of the physical or virtual servers.
- Data is stored in the clusters of the cloud provider.
- The service can be accessed anywhere in the world.

- Eg:
  - E-mail system might have millions of users and each user can have thousands of e-mails and consume multiple gigabytes of disk space.
  - Web searching application.
  - To store huge amount of information solid-state drives are used instead of hard disk drives

In storage technologies, hard disk drives may be augmented (increased) with solid-state drives in the future.

Table 4.8 Storage Services in Three Cloud Computing Systems

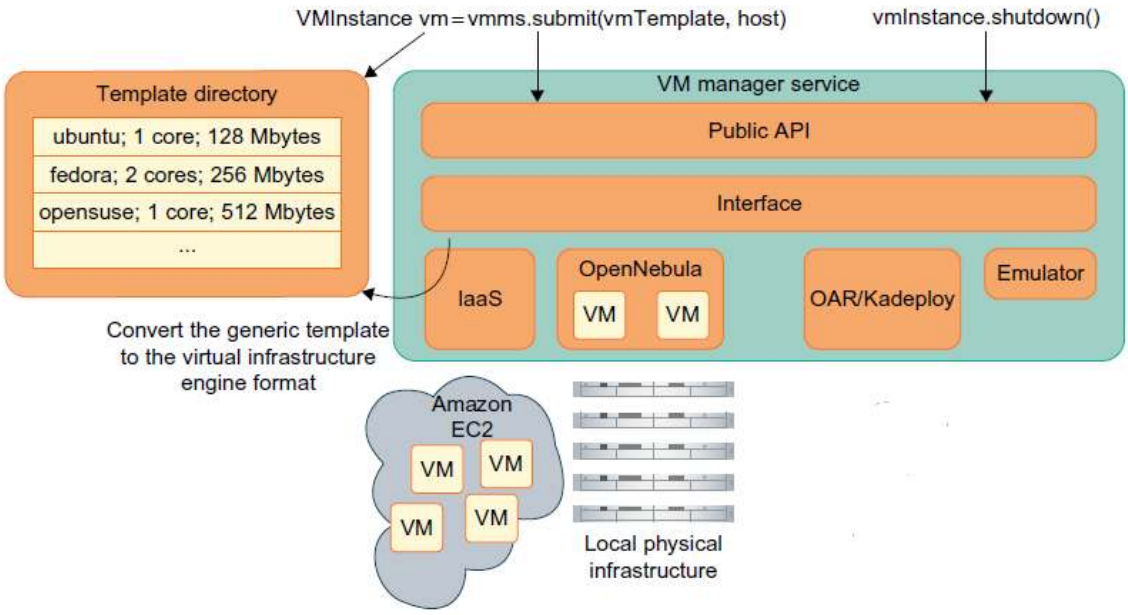
Storage System	Features
GFS: Google File System	Very large sustainable reading and writing bandwidth, mostly continuous accessing instead of random accessing. The programming interface is similar to that of the POSIX file system accessing interface.
HDFS: Hadoop Distributed File System	The open source 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.

4.5.3 Virtual Machine Creation and Management

The managers provide a public API for users to submit and control the VMs

.

Fig. Virtual Machine Creation and Management



Independent Service Management:

- Independent services request facilities to execute many unrelated tasks.
- Commonly, the APIs provided are some web services that the developer can use conveniently.

Running Third-Party Applications

- Cloud platforms have to provide support for building applications that are constructed by third-party
  application providers or programmers.

- The APIs are often in the form of services.
- Web service application engines are often used by programmers for building applications.
- The web browsers are the user interface for end users.

Virtual Machine Manager

The manager manage VMs deployed on a set of physical resources

- VIEs(Virtual Infrastructure Engine) can create and stop VMs on a physical cluster
- Users submit VMs on physical machines using different kinds of hypervisors

- To deploy a VM, the manager needs to use its template.
- Virtual Machine Templates contains a description for a VM with the following static information:
    - The number of cores or processors to be assigned to the VM
    - The amount of memory the VM requires
    - The kernel used to boot the VM's operating system.
    - The price per hour of using a VM
- OAR/Kadeploy is a deployment tool
- API(Application Programming Interface) - An API is a software intermediary that makes it possible for application programs to interact with each other and share data

Virtual Machine Templates

- A VM template is analogous to a computer's configuration and contains a description for a VM with the following static information:
- The number of cores or processors to be assigned to the VM
- The amount of memory the VM requires
- The kernel used to boot the VM's operating system
- The disk image containing the VM's file system
- The price per hour of using a VM

Distributed VM Management

- A distributed VM manager makes requests for VMs and queries their status.
- This manager requests VMs from the gateway on behalf of the user application.
- The manager obtains the list of requested VMs from the gateway.
- This list contains a tuple of public IP/private IP addresses for each VM with Secure Shell (SSH) tunnels.

4.1.4 Global Exchange of Cloud Resources

- 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.
- 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.
- The figure the high-level components of the Melbourne group's proposed InterCloud architecture

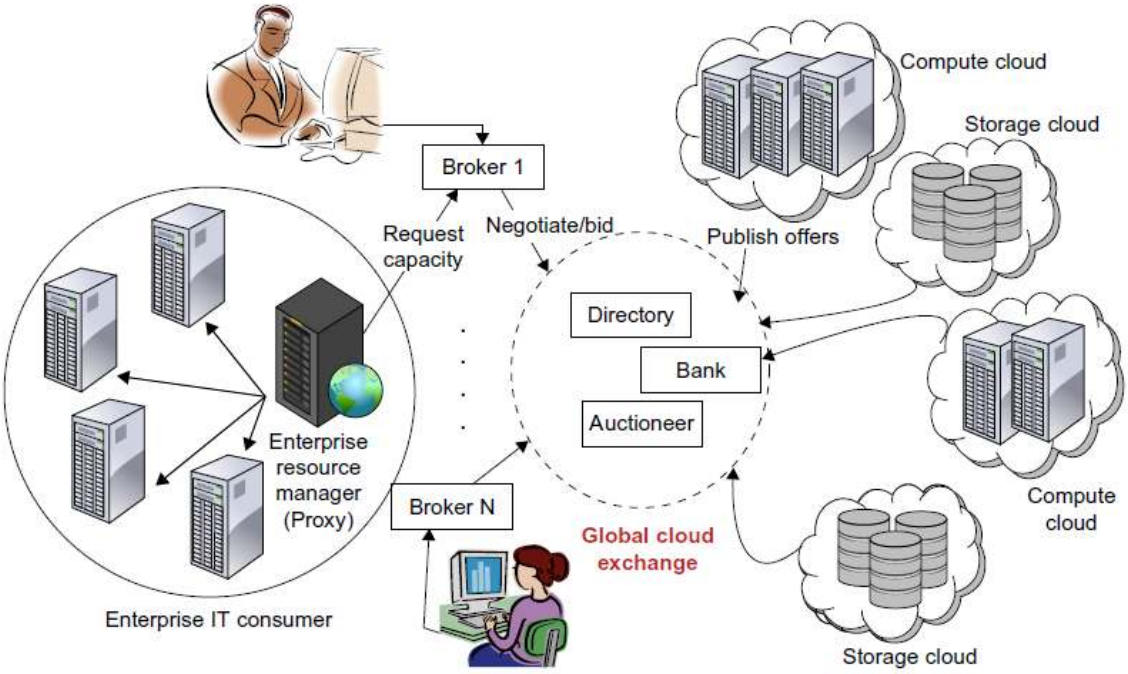


Fig: Inter-cloud exchange of cloud resources through brokering

- It is **not possible** for a cloud infrastructure provider to establish its **data centers at all possible locations** throughout the world.
- This results in **difficulty** in meeting the **QOS expectations** of their customers.

- Hence, services of **multiple cloud infrastructure** service providers are used.
- **Cloud coordinator** evaluates the available resources.
- The availability of a banking system ensures that financial transactions related to SLAs are carried out in a securely.
- By realizing InterCloud architectural principles in mechanisms in their offering, 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.
- They consist of client brokering and coordinator services that support utility-driven federation of clouds:
    - application scheduling
    - resource allocation
    - 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 crossdomain 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.
- It supports trading of cloud services based on competitive economic models such as commodity markets and auctions.
- CEx allows participants to locate providers and consumers with fitting offers.

4.2 Security

- Virtual machines from multiple organizations have to be co-located on the same physical server in order to maximize the efficiencies of virtualization.

- Cloud service providers must learn from the managed service provider (MSP) model and ensure that their customers' applications and data are secure if they hope to retain their customer base and competitiveness.
- Cloud environment should be free from abuses, cheating, hacking, viruses, rumors, and privacy and copyright violations.

4.2.1 Cloud Security Challenges

- In cloud model users lose control over physical security.
- In a public cloud, users are sharing computing resources with other companies.
- When users share the environment in the cloud, it results in data at risk of seizure (attack).
- Storage services provided by one cloud vendor may be incompatible with another vendor's services; this results in unable to move from one to the other.
- Vendors create "sticky services".
- Sticky services are the services which makes end user, in difficulty while transporting from one cloud vendor to another.

Example: Amazon's "Simple Storage Service" [S3] is incompatible with IBM's Blue Cloud, or Google, or Dell).

- Customers want their data encrypted while **data is at rest** (data stored) in the cloud vendor's storage pool.
- Data integrity means ensuring that data is identically maintained during any operation (such as transfer, storage, or retrieval).
- Data integrity is assurance that the data is consistent and correct.
- One of the key challenges in cloud computing is data-level security.
- It is difficult for a customer to find where its data resides on a network controlled by its provider.
- Some countries have strict limits on what data about its citizens can be stored and for how long.
- Banking regulators require that customers' financial data remain in their home country.

- Security managers will need to pay particular attention to systems that contain critical data such as corporate financial information.
- Outsourcing (giving rights to third party) loses control over data and not a good idea from a security perspective.
- Security managers have to interact with company's legal staff to ensure that appropriate contract terms are in place to protect corporate data.
- Cloud-based services will result in many mobile IT users accessing business data and services without traversing the corporate network.
- This will increase the need for enterprises to place security controls between mobile users and cloud-based services.
- Placing large amounts of sensitive data in a globally accessible cloud leaves organizations open to large distributed threats—attackers no longer have to come onto the premises to steal data, and they can find it all in the one "virtual" location.
- Virtualization efficiencies in the cloud require virtual machines from multiple organizations to be collocated on the same physical resources.
- Although traditional data center security still applies in the cloud environment, physical segregation and hardware-based security cannot protect against attacks between virtual machines on the same server.
- The dynamic and fluid nature of virtual machines will make it difficult to maintain the consistency of security and ensure the auditability of records.
- The ease of cloning and distribution between physical servers could result in the propagation of configuration errors and other vulnerabilities.
- Localized virtual machines and physical servers use the same operating systems as well as enterprise and web applications in a cloud server environment, increasing the threat of an attacker or malware exploiting vulnerabilities in these systems and applications remotely.
- Virtual machines are vulnerable as they move between the private cloud and the public cloud.
- Operating system and application files are on a shared physical infrastructure in a virtualized cloud environment and require system, file, and activity monitoring to provide

confidence and auditable proof to enterprise customers that their resources have not been compromised or tampered with.

- The **Intrusion Detection System(IDS)** and **Intrusion Prevention Systems(IPS)** detects malicious activity at virtual machine level.
- The co-location of multiple virtual machines increases the threat from attacker.
- If Virtual machines and physical machine use the same operating systems in a cloud environment, increases the threat from an attacker.
- A fully or partially shared cloud environment is expected to have a greater attack than own resources environment.
- Virtual machines must be self-defending.
- Cloud computing provider is incharge of customer data security and privacy.

4.2.2 Software as a Service Security (Or) Data Security (Or) Application Security (Or) Virtual Machine Security.

Cloud computing models of the future will likely combine the use of SaaS (and other XaaS's as appropriate), utility computing, and Web 2.0 collaboration technologies to leverage the Internet to satisfy their customers' needs. New business models being developed as a result of the move to cloudcomputing are creating not only new technologies and business operational processes but also newsecurity requirements and challenges

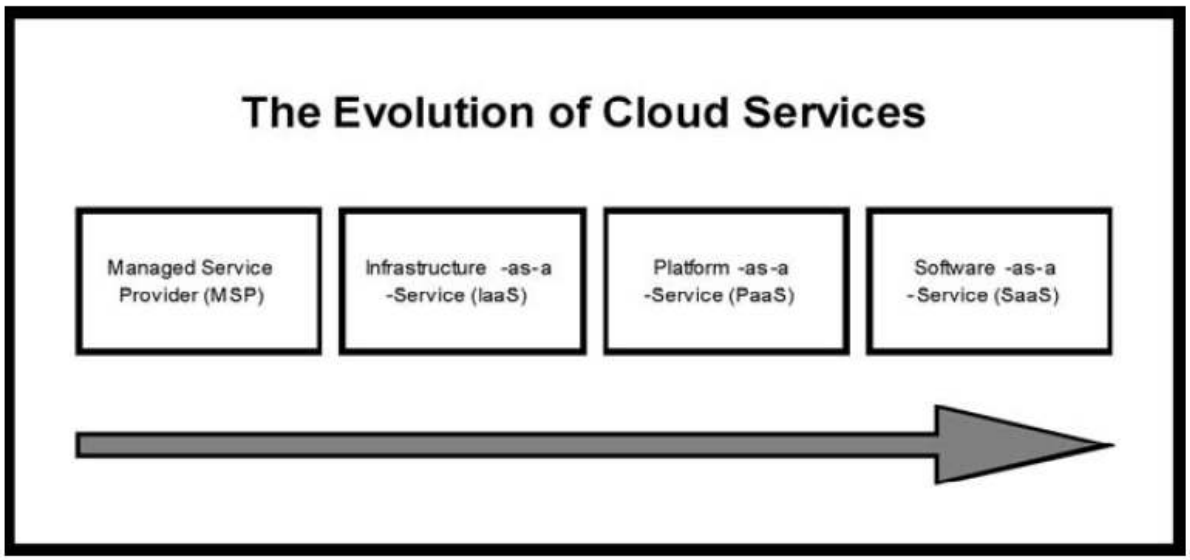


Fig: Evolution of Cloud Services

SaaS plays the dominant cloud service model and this is the area where the most critical need for security practices are required

 Security issues that are discussed with cloud-computing vendor:

1. **Privileged user access**—Inquire about who has specialized access to data, and about the hiring and management of such administrators.
2. **Regulatory compliance**—Make sure that the vendor is willing to undergo external audits and/or security certifications.
3. **Data location**—Does the provider allow for any control over the location of data?
4. **Data segregation**—Make sure that encryption is available at all stages, and that these encryption schemes were designed and tested by experienced professionals.
5. **Recovery**—Find out what will happen to data in the case of a disaster. Do they offer complete restoration? If so, how long would that take?
6. **Investigative support**—Does the vendor have the ability to investigate any inappropriate or illegal activity?
7. **Long-term viability**—What will happen to data if the company goes out of business? How will data be returned, and in what format?

The security practices for the SaaS environment are as follows:

Security Management (People)

- One of the most important actions for a security team is to develop a formal charter for the security organization and program.
- This will foster a shared vision among the team of what security leadership is driving toward and expects, and will also foster "ownership" in the success of the collective team.
- The charter should be aligned with the strategic plan of the organization or company the security team works for.

4.2.3 Security Governance

- A security committee should be developed whose objective is to focus on providing guidance about security initiatives with business and IT strategies.
- A charter for the security team is typically one of the first deliverables from the steering committee.

- This charter must clearly define the roles and responsibilities of the security team and other groups involved in performing information security functions.
- Lack of a formalized strategy can lead to an unsustainable operating model and
- security level as it evolves.
- In addition, lack of attention to security governance can result in key needs of the business not being met, including but not limited to, risk management, security monitoring, applicationsecurity, and sales support.
- Lack of proper governance and management of duties can also result in potential security risks being left unaddressed and opportunities to improve the business being missed.
- The security team is not focused on the key security functions and activities that are critical to the business.

Cloud security governance refers to the management model that facilitates effective and efficient security management and operations in the cloud environment so that an enterprise's business targets are achieved. This model incorporates a hierarchy of executive mandates, performance expectations, operational practices, structures, and metrics that, when implemented, result in the optimization of business value for an enterprise. Cloud security governance helps answer leadership questions such as:

- Are our security investments yielding the desired returns?
- Do we know our security risks and their business impact?
- Are we progressively reducing security risks to acceptable levels?
- Have we established a security-conscious culture within the enterprise?

Strategic alignment, value delivery, risk mitigation, effective use of resources, and performance measurement are key objectives of any IT-related governance model, security included. To successfully pursue and achieve these objectives, it is important to understand the operational culture and business and customer profiles of an enterprise, so that an effective security governance model can be customized for the enterprise.

Cloud Security Governance Challenges

Whether developing a governance model from the start or having to retrofit one on existing investments in cloud, these are some of the common challenges:

Lack of senior management participation and buy-in

The lack of a senior management influenced and endorsed security policy is one of the common challenges facing cloud customers. An enterprise security policy is intended to set the executive tone, principles and expectations for security management and operations in the cloud. However, many enterprises tend to author security policies that are often laden with tactical content, and lack executive input or influence. The result of this situation is the ineffective definition and communication of executive tone and expectations for security in the cloud.

Lack of embedded management operational controls

Another common cloud security governance challenge is lack of embedded management controls into cloud security operational processes and procedures. Controls are often interpreted as an auditor's checklist or repackaged as procedures, and as a result, are not effectively embedded into security operational processes and procedures as they should be, for purposes of optimizing value and reducing day-to-day operational risks. This lack of embedded controls may result in operational risks that may not be apparent to the enterprise. For example, the security configuration of a device may be modified (change event) by a staffer without proper analysis of the business impact (control) of the modification. The net result could be the introduction of exploitable security weaknesses that may not have been apparent with this modification.

Lack of operating model, roles, and responsibilities

Many enterprises moving into the cloud environment tend to lack a formal operating model for security, or do not have strategic and tactical roles and responsibilities properly defined and operationalized. This situation stifles the effectiveness of a security management and operational function/organization to support security in the cloud. Simply, establishing a hierarchy that includes designating an accountable official at the top, supported by a stakeholder committee, management team, operational staff, and third-party provider support (in that order) can help an enterprise to better manage and control security in the cloud, and protect associated investments in accordance with enterprise business goals.

Lack of metrics for measuring performance and risk

Another major challenge for cloud customers is the lack of defined metrics to measure security performance and risks – a problem that also stifles executive visibility into the real security risks in the cloud. This challenge is directly attributable to the combination of other challenges discussed above. For example, a metric that quantitatively measures the number of exploitable security vulnerabilities on host devices in the cloud over time can be leveraged as an indicator of risk in the host device environment. Similarly, a metric that measures the number of user-reported security incidents over a given period can be leveraged as a performance indicator

of staff awareness and training efforts. Metrics enable executive visibility into the extent to which security tone and expectations (per established policy) are being met within the enterprise and support prompt decision-making in reducing risks or rewarding performance as appropriate.

The challenges described above clearly highlight the need for cloud customers to establish a framework to effectively manage and support security in cloud management, so that the pursuit of business targets are not potentially compromised. Unless tone and expectations for cloud security are established (via an enterprise policy) to drive operational processes and procedures with embedded management controls, it is very difficult to determine or evaluate business value, performance, resource effectiveness, and risks regarding security operations in the cloud. Cloud security governance facilitates the institution of a model that helps enterprises explicitly address the challenges described above.

Key Objectives for Cloud Security Governance

Building a cloud security governance model for an enterprise requires strategic-level security management competencies in combination with the use of appropriate security standards and frameworks (e.g., NIST, ISO, CSA) and the adoption of a governance framework (e.g., COBIT). The first step is to visualize the overall governance structure, inherent components, and to direct its effective design and implementation. The use of appropriate security standards and frameworks allow for a minimum standard of security controls to be implemented in the cloud, while also meeting customer and regulatory compliance obligations where applicable. A governance framework provides referential guidance and best practices for establishing the governance model for security in the cloud. The following represents key objectives to pursue in establishing a governance model for security in the cloud. These objectives assume that appropriate security standards and a governance framework have been chosen based on the enterprise's business targets, customer profile, and obligations for protecting data and other information assets in the cloud environment.

1. **Strategic Alignment**

   Enterprises should mandate that security investments, services, and projects in the cloud are executed to achieve established business goals (e.g., market competitiveness, financial, or operationalperformance).

2. **Value Delivery**

   Enterprises should define, operationalize, and maintain an appropriate security function/organization with appropriate strategic and tactical representation, and charged with the

responsibility to maximize the business value (Key Goal Indicators, ROI) from the pursuit of security initiatives in the cloud.

3. **Risk Mitigation**

   Security initiatives in the cloud should be subject to measurements that gauge effectiveness in mitigating risk to the enterprise (Key Risk Indicators). These initiatives should also yield results that progressively demonstrate a reduction in these risks over time.

4. **Effective Use of Resources**

   It is important for enterprises to establish a practical operating model for managing and performing security operations in the cloud, including the proper definition and operationalization of due processes, the institution of appropriate roles and responsibilities, and use of relevant tools for overall efficiency and effectiveness.

5. **Sustained Performance**

   Security initiatives in the cloud should be measurable in terms of performance, value and risk to the enterprise (Key Performance Indicators, Key Risk Indicators), and yield results that demonstrate attainment of desired targets (Key Goal Indicators) over time.

Risk Management

- Effective risk management entails identification of technology assets; identification of data and its links to business processes, applications, and data stores; and assignment of ownership and custodial responsibilities.
- Actions should also include maintaining a repository of information assets
- A risk assessment process should be created that allocates security resources related to business continuity.

Risk Assessment

- Security risk assessment is critical to helping the information security organization make informed decisions when balancing the dueling priorities of business utility and protection of assets.
- Lack of attention to completing formalized risk assessments can contribute to an increase in information security audit findings, can jeopardize certification goals, and can lead to

inefficient and ineffective selection of security controls that may not adequately mitigate information security risks to an acceptable level.

Security Portfolio(selection) Management

- Security portfolio management ensures efficient and effective operation of any information.

Security Awareness

- Not providing proper awareness and training to the people who may need them can expose the company to a variety of security risks

Policies, Standards, and Guidelines

- Policies, standards, and guidelines are developed that can ensure consistency of performance.

Secure Software Development Life Cycle (SecSDLC)

- The SecSDLC involves identifying specific threats and the risks. The SDLC consists of six phases

Phase 1.Investigation:

-Define project goals, and document them.

Phase 2.Analysis:

-Analyze current threats and perform risk analysis.

Phase 3.Logical design:

-Develop a security blueprint(plan) and business responses to disaster.

Phase 4.Physical design:

-Select technologies to support the security blueprint(plan).

Phase 5.Implementation:

- Buy or develop security solutions.

Phase 6.Maintenance:

-Constantly monitor, test, modify, update, and repair to respond to changing threats.

Security Monitoring and Incident Response

- Centralized security management systems should be used to provide notification of security vulnerabilities and to monitor systems continuously.

Business Continuity Plan

Business continuity plan, ensures uninterrupted operations of business.

Forensics

Forensics includes recording and analyzing events to determine the nature and source of information abuse, security attacks, and other such incidents.

Security Architecture Design

A security architecture framework should be established with the following consideration

1. Authentication
2. Authorization
3. Availability
4. Confidentiality
5. Integrity
6. Privacy

Vulnerability Assessment

- Vulnerability assessment classifies network assets to more efficiently prioritize vulnerability-mitigation programs, such as patching and system upgrading.
- It measures the effectiveness of risk mitigation by setting goals of reduced vulnerability exposure and faster mitigation

Password Assurance Testing

- If the SaaS security team or its customers want to periodically test password strength by running
- password "crackers," they can use cloud computing to decrease crack time and pay only for what they use.
-

Security Images:

- Virtualization-based cloud computing provides the ability to create "Gold image" VM secure builds and to clone multiple copies.
- Gold image VMs also provide the ability to keep security up to date and reduce exposure by patching offline.

Data Privacy

- Depending on the size of the organization and the scale of operations, either an individual or a team should be assigned and given responsibility for maintaining privacy.
- A member of the security team who is responsible for privacy or security compliance team should collaborate with the company legal team to **address data privacy issues and concerns.**

- **Hiring a consultant** in privacy area, will ensure that your organization is prepared to meet the data privacy demands of its customers and regulators.

Data Governance

The data governance framework should include:

_ Data inventory

_ Data classification

_ Data analysis (business intelligence)

_ Data protection

_ Data privacy

_ Data retention/recovery/discovery

_ Data destruction

Data Security

The challenge in cloud computing is data-level security.

Security to data is given by

 Encrypting the data

 Permitting only specified users to access the data.

 Restricting the data not to cross the countries border.

For example, with data-level security, the enterprise can specify that this data is not allowed to go outside of the India.

Application Security

 This is collaborative effort between the security and product development team.

 Application security processes

o Secure coding guidelines

o Training

o Testing scripts

o Tools

 Penetration Testing is done to a System or application.

 Penetration Testing is defined as a type of Security Testing used to test the **insecure areas of the system or application**.

 The goal of this testing is to **find all the security vulnerabilities** that are present in the system being tested.

 SaaS providers should secure their web applications by following **Open Web Application Security Project (OWASP) guidelines** for secure application development, **by locking down ports** and unnecessary commands

5.3 Virtual Machine Security

In the cloud environment, physical servers are consolidated (combined) to multiple virtual machine instances.

Following are deployed on virtual machines to ensure security

 Firewalls

 Intrusion detection and prevention

 Integrity monitoring

 Log inspection

Virtual servers have security requirements identical to those of physical servers. The same applies to the applications and services they host. Virtualization provides security benefits: each virtual machine has a private security context, potentially with separate authentication and authorization rules, and with separate process, name and file system spaces. Deploying applications onto separate virtual machines provides better security control compared to running multiple applications on the same host operating system: penetrating one virtual machine's OS doesn't necessarily compromise workload and data residing in other virtual machines. Nonetheless, some practices should be kept in mind to prevent virtualization from introducing security vulnerabilities.

One aspect is physical security. Virtual infrastructure is not as 'visible' as physical infrastructure: there is no sticky label on a virtual machine to indicate its purpose and security classification. If a datacenter identifies servers with extremely high security requirements, and physically isolates them in a locked room or cage to prevent tampering or theft of data, then the physical machines hosting their virtualized workloads should be isolated in a similar way. Even without secured areas, many institutions keep workloads of different security classes on different servers. Those same isolation rules apply for virtual machines. Care should be taken to ensure

that the protected virtual machines are not migrated to a server in a less secure location. In the context of Oracle VM, this implies maintaining separate server pools, each with their own group of servers.

These rules of isolation should also be applied to networking: there are no color coded network cables to help staff identify and isolate different routes, segments and types network traffic to and from virtual machines or between them. There are no visual indicators that help ensure that application, management, and backup traffic are kept separate. Rather than plug network cables into different physical interfaces and switches, the Oracle VM administrator must ensure that the virtual network interfaces are connected to separate virtual networks. Specifically, use VLANs to isolate virtual machines from one another, and assign virtual networks for virtual machine traffic to different physical interfaces from those used for management, storage or backup. These can all be controled from the Oracle VM Manager user interface. Ensure that secure live migration is selected to guarantee that virtual machine memory data is not sent across the wire unencrypted.

Additional care must be given to virtual machine disk images. In most cases the virtual disks are made available over the network for migration and failover purposes. In many cases they are files, which could easily be copied and stolen if the security of network storage is compromised. Therefore it is essential to lock down the NAS or SAN environments and prevent unauthorized access. An intruder with root access to a workstation on the storage network could mount storage assets and copy or alter their contents. Use a separate network for transmission between the storage servers and the Oracle VM hosts to ensure its traffic is not made public and subject to being snooped. Make sure that unauthorized individuals are not permitted to log into the Oracle VM Servers, as that would give them access to the guests' virtual disk images, and potentially much more.

All of these steps require controlling access to the Oracle VM Manager and Oracle VM Server domain 0 instances. Network access to these hosts should be on a private network, and the user accounts able to log into any of the servers in the Oracle VM environment should be rigorously controlled, and limited to the smallest possible number of individuals.

4.4 Identity and access management architecture(IAM)

Basic concept and definitions of IAM functions for any service:

Authentication – is a process of verifying the identity of a user or a system.Authentication usually connotes a more roburst form of identification. In some use cases such as service – to- service interaction, authentication involves verifying the network service.

Authorization – is a process of determining the privileges the user or system is entitled to once the identity is established. Authorization usually follows theauthentication step and is used to determine whether the user or service has the necessary privileges to perform certain operations.

Auditing – Auditing entails the process of review and examination ofauthentication, authorization records and activities to determine the adequacy of IAMsystem controls, to verify complaints with established security policies and procedure,to detect breaches in security services and to recommend any changes that areindicated for counter measures

IAM Architecture and Practice

IAM is not a monolithic solution that can be easily deployed to gain capabilitiesimmediately. It is as much an aspect of architecture as it is acollection of technology components, processes, and standard practices. Standardenterprise IAM architecture encompasses several layers of technology, services, andprocesses. At the core of the deployment architecture is a directory service (such as LDAP or Active Directory) that acts as a repository for the identity, credential, and user attributes of the organization's user pool. The directory interacts with IAM technology components such as authentication, user management, provisioning, and federation services that support the standard IAM practice and processes within the organization.

The IAM processes to support the business can be broadly categorized as follows:

User management: Activities for the effective governance and management of identity life cycles

Authentication management: Activities for the effective governance and management of the process for determining that an entity is who or what it claims tobe.

Authorization management: Activities for the effective governance and management of the process for determining entitlement rights that decide what resources an entity is permitted to access in accordance with the organization's policies.

Access management: Enforcement of policies for access control in response to a request from an entity (user, services) wanting to access an IT resource within the organization.

Data management and provisioning: Propagation of identity and data for authorization to IT resources via automated or manual processes.

Monitoring and auditing: Monitoring, auditing, and reporting compliance by users regarding access to resources within the organization based on the defined policies.

IAM processes support the following operational activities:

Provisioning: Provisioning can be thought of as a combination of the duties of the human resources and IT departments, where users are given access to data repositories or systems, applications, and databases based on a unique user identity. Deprovisioning works in the opposite manner, resulting in the deletion or deactivation of an identity or of privileges assigned to the user identity.

Credential and attribute management: These processes are designed to manage the life cycle of credentials and user attributes— create, issue, manage, revoke—to inappropriate account use. Credentials are usually bound to an individual and are verified during the authentication process. The processes include provisioning of attributes, static (e.g., standard text password) and dynamic (e.g., one-time password) credentials that comply with a password standard (e.g., passwords resistant to dictionary attacks), handling password expiration, encryption management of credentials during transit and at rest, and access policies of user attributes (privacy and handling of attributes for various regulatory reasons).Minimize the business risk associated with

identityimpersonation

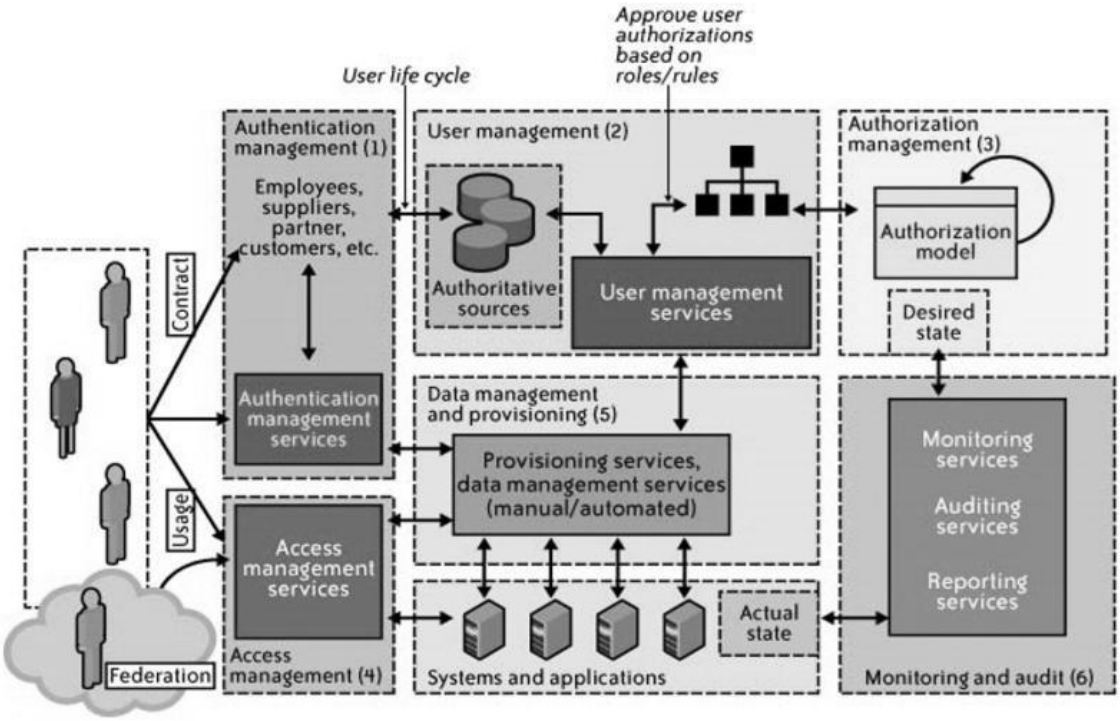


Figure 5.7 Enterprise IAM functional architecture

Entitlement management: Entitlements are also referred to as authorization policies. The processes in this domain address the provisioning and deprovisioning of privileges needed for the user to access resources including systems, applications, and databases. Proper entitlement management ensures that users are assigned only the required privileges.

Compliance management: This process implies that access rights and privileges are monitored and tracked to ensure the security of an enterprise's resources. The process also helps auditors verify compliance to various internal access control policies, and standards that include practices such as segregation of duties, access monitoring, periodic auditing, and reporting. An example is a user certification process that allows application owners to certify that only authorized users have the privileges necessary to access business-sensitive information.

Identity federation management: Federation is the process of managing the trust relationships established beyond the internal network boundaries or administrative domain boundaries among distinct organizations. A federation is an association of organizations that come together to exchange information about their users and resources to enable collaborations and transactions.

Centralization of authentication (authN) and authorization (authZ): A central authentication and authorization infrastructure alleviates the need for application developers to build custom authentication and authorization features into their applications. Furthermore, it promotes a loose coupling architecture where applications become agnostic to the authentication methods and policies. This approach is also called an ―externalization of authN and authZ from applications

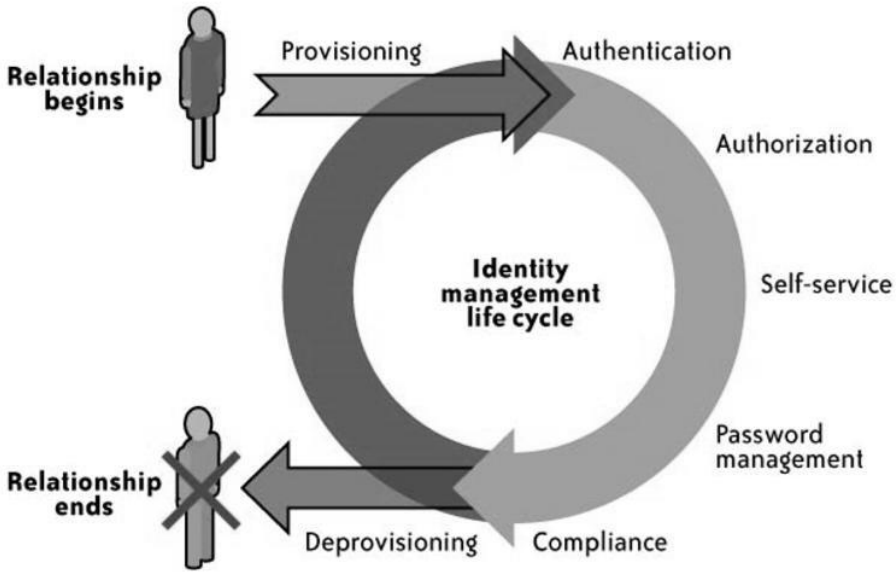


Figure 5.8 Identity Life cycle

IAM Standards and Specifications for Organisations

The following IAM standards and specifications will help organizations implement effective and efficient user access management practices and processes inthe cloud. These sections are ordered by four major challenges in user and access management faced by cloud users:

1. How can I avoid duplication of identity, attributes, and credentials and provide a single sign-on user experience for my users? SAML.
2. How can I automatically provision user accounts with cloud services and automate the process of provisoning and deprovisioning? SPML.

IAM Practices in the Cloud

When compared to the traditional applications deployment model within the enterprise, IAM practices in the cloud are still evolving. In the current state of IAM technology, standards support by CSPs (SaaS, PaaS, and IaaS) is not consistent across providers. Although large providers such as Google, Microsoft, and Salesforce.com seem to demonstrate basic IAM

capabilities, our assessment is that they still fall short of enterprise IAM requirements for managing regulatory, privacy, and data protection requirements. The maturity model takes into account the dynamic nature of IAM users, systems, and applications in the cloud and addresses the four key components of the IAM automation process:

• User Management, New Users
• User Management, User Modifications
• Authentication Management
• Authorization Management

IAM practices and processes are applicable to cloud services; they need to be adjusted to the cloud environment. Broadly speaking, user management functions in the cloud can be categorized as follows:

• Cloud identity administration, Federation or SSO
• Authorization management
• Compliance management

Cloud Identity Administration: Cloud identity administrative functions should focus on life cycle management of user identities in the cloud—provisioning, deprovisioning, identity federation, SSO, password or credentials management, profile management, and administrative management. Organizations that are not capable of supporting federation should explore cloud-based identity management services. This new breed of services usually synchronizes an organization's internal directories with its directory (usually multitenant) and acts as a proxy IdP for the organization.

Federated Identity (SSO): Organizations planning to implement identity federation that enables SSO for users can take one of the following two paths (architectures):

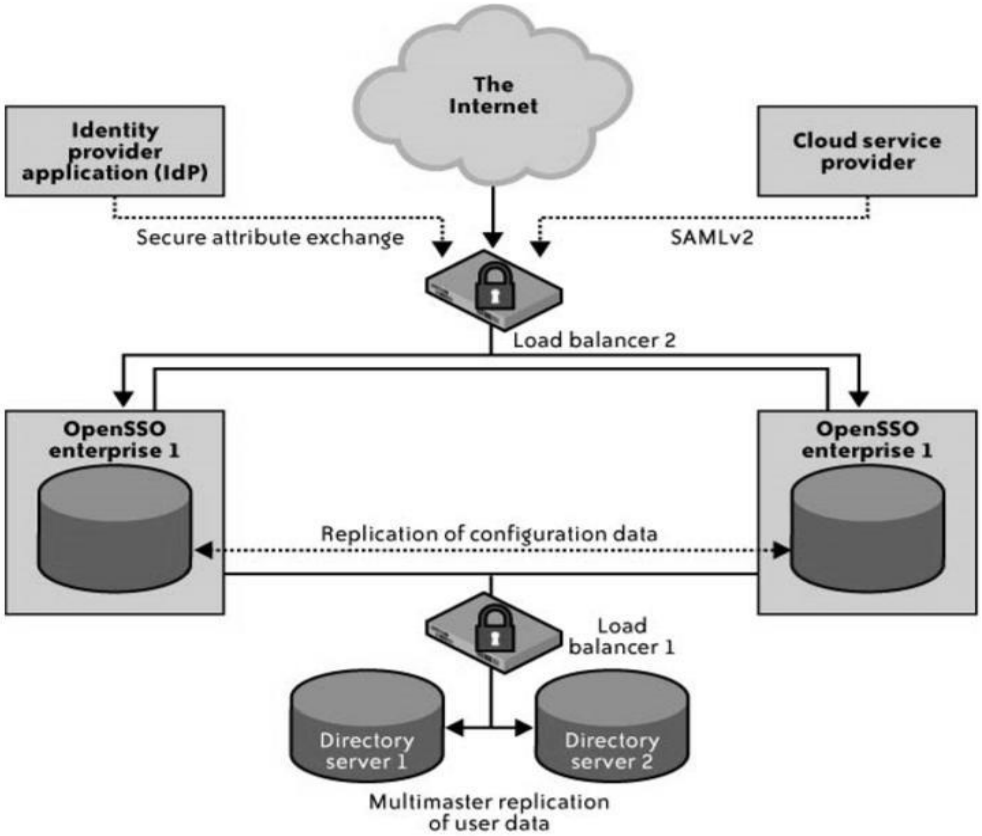
• Implement an enterprise IdP within an organization perimeter.
• Integrate with a trusted cloud-based identity management service provider.

Both architectures have pros and cons.

Enterprise identity provider: In this architecture, cloud services will delegate authentication to an organization's IdP. In this delegated authentication architecture, the organization federates identities within a trusted circle of CSP domains. A circle of trust can be created with all the domains that are authorized to delegate authentication to the IdP. In this deployment architecture,

where the organization will provide and support an IdP, greater control can be exercised over user identities, attributes, credentials, and policies for authenticating and authorizing users to a cloud service.

IdP deployment architecture.



4.5 Security standards

Security standards define the processes, procedures, and practices necessary for implementing a security program. These standards also apply to cloud-related IT activities and include specific steps that should be taken to ensure a secure environment is maintained that provides privacy and security of confidential information in a cloud environment. Security standards are based on a set of key principles intended to protect this type of trusted environment. Messaging standards, especially for security in the cloud, must also include nearly all the same considerations as any other IT security endeavor.

Security (SAML ,OAuth, OpenID, SSL/TLS)

A basic philosophy of security is to have layers of defense, a concept known as *defense in depth*. This means having overlapping systems designed to provide security even if one system fails. An example is a firewall working in conjunction with an intrusion-detection system (IDS). Defense in depth provides security because there is no single point of failure and no single-entry vector at which an attack can occur. No single security system is a solution by itself, so it is far better to secure all systems. This type of layered security is precisely what we are seeing develop in cloud computing. Traditionally, security was implemented at the endpoints, where the user controlled access. An organization had no choice except to put firewalls, IDSs, and antivirus software inside its own network. Today, with the advent of managed security services offered by cloud providers, additional security can be provided inside the cloud.

4.5.1 Security Assertion Markup Language (SAML)

SAML is an XML-based standard for communicating authentication, authorization, and attribute information among online partners. It allows businesses to securely send assertions between partner organizations regarding the identity and entitlements of a principal. The Organization for the Advancement of Structured Information Standards (OASIS) Security Services Technical Committee is in charge of defining, enhancing, and maintaining the SAML specifications.

SAML is built on a number of existing standards, namely, SOAP, HTTP, and XML. SAML relies on HTTP as its communications protocol and specifies the use of SOAP (currently, version 1.1). Most SAML transactions are expressed in a standardized form of XML. SAML assertions and protocols are specified using XML schema. Both SAML 1.1 and SAML 2.0 use digital signatures (based on the XML Signature standard) for authentication and message integrity. XML encryption is supported in SAML 2.0, though SAML 1.1 does not have encryption capabilities. SAML defines XML-based assertions and protocols, bindings, and profiles. The term SAML Core refers to the general syntax and semantics of SAML assertions as well as the protocol used to request and transmit those assertions from one system entity to another. SAML protocol refers to what is transmitted, not how it is transmitted. A SAML binding determines how SAML requests and responses map to standard messaging protocols. An important (synchronous) binding is the SAML SOAP binding.

SAML standardizes queries for, and responses that contain, user authentication, entitlements, and attribute information in an XML format. This format can then be used to request security information about a principal from a SAML authority. A SAML authority, sometimes called the asserting party, is a platform or application that can relay security information. The relying party (or assertion consumer or requesting party) is a partner site that receives the security information.

The exchanged information deals with a subject's authentication status, access authorization, and attribute information. A subject is an entity in a particular domain. A person identified by an email address is a subject, as might be a printer.

SAML assertions are usually transferred from identity providers to service providers. Assertions contain statements that service providers use to make access control decisions. Three types of statements are provided by SAML: authentication statements, attribute statements, and authorization decision statements. SAML assertions contain a packet of security information in this form:

<saml:Assertion A...>
<Authentication>
...
</Authentication>
<Attribute>
...
</Attribute>
<Authorization>
...
</Authorization>
</saml:Assertion A>

The assertion shown above is interpreted as follows:

Assertion A, issued at time T by issuer I, regarding subject
S, provided conditions C are valid.

Authentication statements assert to a service provider that the principal did indeed authenticate with an identity provider at a particular time using a particular method of authentication. Other information about the authenticated principal (called the authentication

context) may be disclosed in an authentication statement. An attribute statement asserts that a subject is associated with certain attributes. An attribute is simply a name-value pair. Relying parties use attributes to make access control decisions. An authorization decision statement asserts that a subject is permitted to perform action A on resource R given evidence E. The expressiveness of authorization decision statements in SAML is intentionally limited.

A SAML protocol describes how certain SAML elements (including assertions) are packaged within SAML request and response elements. It provides processing rules that SAML entities must adhere to when using these elements. Generally, a SAML protocol is a simple request-response protocol. The most important type of SAML protocol request is a query. A service provider makes a query directly to anidentity provider over a secure back channel. For this reason, query messages are typically bound to SOAP. Corresponding to the three types of statements, there are three types of SAML queries: the authentication query, the attribute query, and the authorization decision query. Of these, the attribute query is perhaps most important. The result of an attribute query is a SAML response containing anassertion, which itself contains an attribute statement.

4.5.2 Open Authentication (OAuth)

OAuth is an open protocol, initiated by Blaine Cook and Chris Messina, to allow secure API authorization in a simple, standardized method for various types of web applications. Cook and Messina had concluded that there were no open standards for API access delegation. The OAuth discussion group was created in April 2007, for the small group of implementers to write the draft proposal for an open protocol. DeWitt Clinton of Google learned of the OAuth project and expressed interest in supporting the effort. In July 2007 the team drafted an initial specification, and it was released in October of the same year. OAuth is a method for publishing and interacting with protected data. For developers, OAuth provides

users access to their data while protecting account credentials. OAuth allows users to grant access to their information, which is shared by the service provider and consumers without sharing all of their identity. The Core designation is used to stress that this is the baseline, and other extensions and protocols can build on it. By design, OAuth Core 1.0 does not provide many desired features (e.g., automated discovery of endpoints, language support, support for XML-RPC and SOAP, standard definition of resource access, OpenID integration, signing algorithms, etc.). This intentional lack of feature support is viewed by the authors as a significant

benefit. The Core deals with fundamental aspects of the protocol, namely, to establish a mechanism for exchanging a user name and password for a token with defined rights and to provide tools to protect the token. . In fact, OAuth by itself *provides no privacy at all* and depends on other protocols such as SSL to accomplish that.

4.5.3OpenID

OpenID is an open, decentralized standard for user authentication and access control that allows users to log onto many services using the same digital identity. It is a single-sign-on (SSO) method of access control. As such, it replaces the common log-in process (i.e., a log-in name and a password) by allowing users to log in once and gain access to resources across participating systems. The original OpenID authentication protocol was developed in May 2005 by Brad Fitzpatrick, creator of the popular community web site Live-Journal. In late June 2005, discussions began between OpenID developers and other developers from an enterprise software company named Net-Mesh. These discussions led to further collaboration on interoperability between OpenID and NetMesh's similar Light-Weight Identity (LID) protocol. The direct result of the collaboration was the Yadis discovery protocol, which was announced on October 24, 2005.

The Yadis specification provides a general-purpose identifier for a person and any other entity, which canbe used with a variety of services. It provides a syntax for a resource description document identifying services available using that identifier and an interpretation of the elements of that document. Yadis discovery protocol is used for obtaining a resource description document, given that identifier. Together these enable coexistence and interoperability of a rich variety of services using a single identifier. The identifier uses a standard syntax and a well-established namespace and requires no additional namespace administration infrastructure.

An OpenID is in the form of a unique URL and is authenticated by the entity hosting the OpenID URL.The OpenID protocol does not rely on a central authority to authenticate a user's identity. Neither the OpenID protocol nor any web sites requiring identification can mandate that a specific type of authentication be used; nonstandard forms of authentication such as smart cards, biometrics, or ordinary passwords are allowed. A typical scenario for using OpenID might be something like this: A user visits a web site that displays an OpenID log-in form somewhere on the page. Unlike a typical log-in form, which has fields for user name and password, the OpenID

log-in form has only one field for the OpenID identifier (which is an OpenID URL). This form is connected to an implementation of an OpenID client library.

A user will have previously registered an OpenID identifier with an OpenID identity provider. The user types this OpenID identifier into the OpenID log-in form. The relying party then requests the web page located at that URL and reads an HTML link tag to discover the identity provider service URL. With OpenID 2.0, the client discovers the identity provider service URL by requesting the XRDS document (also called the Yadis document) with the content type **application/xrds+xml,** which may be available at the target URL but is always available for a target XRI.

There are two modes by which the relying party can communicate with the identity provider: **checkid_immediate** and **checkid_setup.** In **checkid_immediate,** the relying party requests that the provider not interact with the user. All communication is relayed through the user's browser without explicitly notifying the user. In **checkid_setup,** the user communicates with the provider server directly using the same web browser as is used to access the relying party site. The second option is more popular on the web.

To start a session, the relying party and the identity provider establish a shared secret—referenced by an associate handle—which the relying party then stores. Using **checkid_setup,** the relying party redirects the user's web browser to the identity provider so that the user can authenticate with the provider. The method of authentication varies, but typically, an OpenID identity provider prompts the user for a password, then asks whether the user trusts the relying party web site to receive his or her credentials and identity details. If the user declines the identity provider's request to trust the relying party web site, the browser is redirected to the relying party with a message indicating that authentication was rejected.

The site in turn refuses to authenticate the user. If the user accepts the identity provider's request to trust the relying party web site, the browser is redirected to the designated return page on the relying party web site along with the user's credentials. That relying party must then confirm that the credentials really came from the identity provider. If they had previously established a shared secret, the relying party can validate the shared secret received with the credentials against the one previously stored. In this case, the relying party is considered to be stateful, because it stores the shared secret between sessions (a process sometimes referred to as

persistence). In comparison, a stateless relying party must make background requests using the **check_authentication** method to be sure that the data came from the identity provider.

4.5.4 SSL/TLS

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographically secure protocols designed to provide security and data integrity for communications over TCP/IP. TLS and SSL encrypt the segments of network connections at the transport layer. Several versions of the protocols are in general use in web browsers, email, instant messaging, and voice-over-IP. TLS is an IETF standard protocol which was last updated in RFC 5246.

The TLS protocol allows client/server applications to communicate across a network in a way specifically designed to prevent eavesdropping, tampering, and message forgery. TLS provides endpoint authentication and data confidentiality by using cryptography. TLS authentication is one-way—the server is authenticated, because the client already knows the server's identity. In this case, the client remains unauthenticated. At the browser level, this means that the browser has validated the server's certificate—more specifically, it has checked the digital signatures of the server certificate's issuing chain ofCertification Authorities (CAs).

Validation does not identify the server to the end user. For true identification, the end user must verify the identification information contained in the server's certificate (and, indeed, its whole issuing CA chain).This is the only way for the end user to know the "identity" of the server, and this is the only way identity can be securely established, verifying that the URL, name, or address that is being used is specified inthe server's certificate. Malicious web sites cannot use the valid certificate of another web site becausethey have no means to encrypt the transmission in a way that it can be decrypted with the valid certificate.

Since only a trusted CA can embed a URL in the certificate, this ensures that checking the apparent URL with the URL specified in the certificate is an acceptable way of identifying the site.TLS also supports a more secure bilateral connection mode whereby both ends of the connection can be assured that they are communicating with whom they believe they are connected. This is known asmutual (assured) authentication. Mutual authentication requires the TLS client-side to also maintain a certificate.

TLS involves three basic phases:

1. Peer negotiation for algorithm support
2. Key exchange and authentication
3. Symmetric cipher encryption and message authentication

During the first phase, the client and server negotiate cipher suites, which determine which ciphers are used; makes a decision on the key exchange and authentication algorithms to be used; and determines the message authentication codes. The key exchange and authentication algorithms are typically public key algorithms. The message authentication codes are made up from cryptographic hash functions. Once these decisions are made, data transfer may begin.

UNIT V CLOUD TECHNOLOGIES AND ADVANCEMENTS 8

Hadoop – MapReduce – Virtual Box -- Google App Engine – Programming Environment for Google App Engine — Open Stack – Federation in the Cloud – Four Levels of Federation – Federated Services and Applications – Future of Federation

5.1 Introduction to Hadoop Framework

- Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models.
- Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.
- Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes.

Hadoop = MapReduce + HDFS

(MapReduce→ Processing ; HDFS → Storage)

Users of Hadoop:

- Hadoop is running search on some of the Internet's largest sites:
  - Amazon Web Services: Elastic MapReduce
  - AOL: Variety of uses, e.g., behavioral analysis & targeting
  - Ebay: Search optimization (532-node cluster)
  - Facebook: Reporting/analytics, machine learning (1100 m.)
  - LinkedIn: People You May Know (2x50 machines)
  - Twitter: Store + process tweets, log files, other data Yahoo: >36,000 nodes; biggest cluster is 4,000 nodes

Hadoop Architecture

- Hadoop has a Master Slave Architecture for both Storage & Processing
- Hadoop framework includes following four modules:
- Hadoop Common: These are Java libraries and provide file system and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.

- ❖ Hadoop YARN: This is a framework for job scheduling and cluster resource management.
- ❖ Hadoop Distributed File System (HDFS): A distributed file system that provides high-throughput access to application data.
- ❖ HadoopMapReduce: This is system for parallel processing of large data sets.

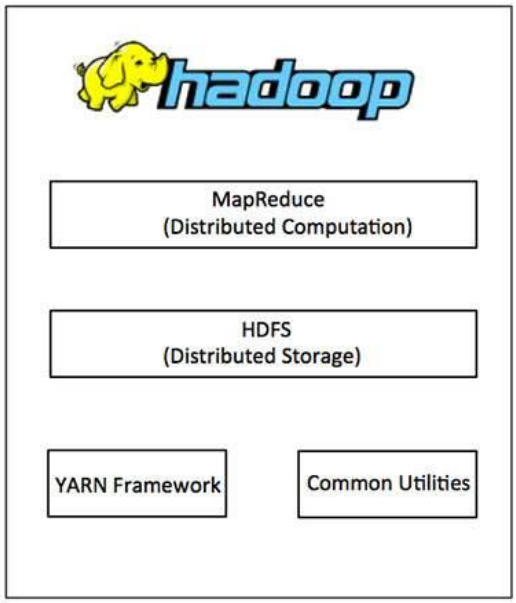


Figure 5.1 Hadoop Architecture

- ☐ The Hadoop core is divided into two fundamental layers:
  - ◘ MapReduce engine
  - ◘ HDFS
- ☐ The MapReduce engine is the computation engine running on top of HDFS as its data storage manager.
- ☐ 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 Name Node as the master and a number of Data Nodes as workers (slaves).

HDFS

- ☐ To store a file in this architecture,
- ☐ HDFS splits the file into fixed-size blocks (e.g., 64 MB) and stores them on workers (Data Nodes).

- The mapping of blocks to Data Nodes is determined by the Name Node.
- The NameNode (master) also manages the file system's metadata and namespace.
- 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.
- NameNode in the metadata stores all information regarding the location of input splits/blocks in all DataNodes.
- Each DataNode, usually one per node in a cluster, manages the storage attached to the node.
- Each DataNode is responsible for storing and retrieving its file blocks

HDFS- Features

Distributed file systems have special requirements

- Performance
- Scalability
- Concurrency Control
- Fault Tolerance
- Security Requirements

HDFS Fault Tolerance

Block replication:

- To reliably store data in HDFS, file blocks are replicated in this system.
- 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: The placement of replicas is another factor to fulfill the desired fault tolerance in HDFS.
- Storing replicas on different nodes (DataNodes) located in different racks across the whole cluster.
- HDFS stores one replica in the same node the original data is stored.
- One replica on a different node but in the same rack
- One replica on a different node in a different rack.
- 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.
- Each Blockreport contains a list of all blocks on a DataNode .
- The NameNode receives such messages because it is the sole decision maker of all replicas in the system.

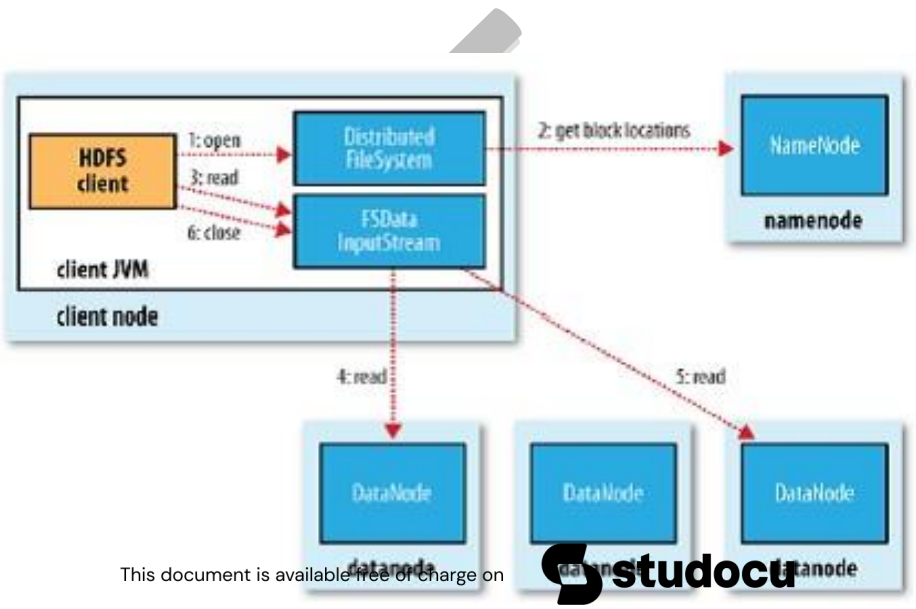
HDFS High Throughput

- Applications run on HDFS typically have large data sets.
- Individual files are broken into large blocks to allow HDFS to decrease the amount of metadata storage required per file.
- The list of blocks per file will shrink as the size of individual blocks increases.
- By keeping large amounts of data sequentially within a block, HDFS provides fast streaming reads of data.

HDFS- Read Operation

Reading a file :

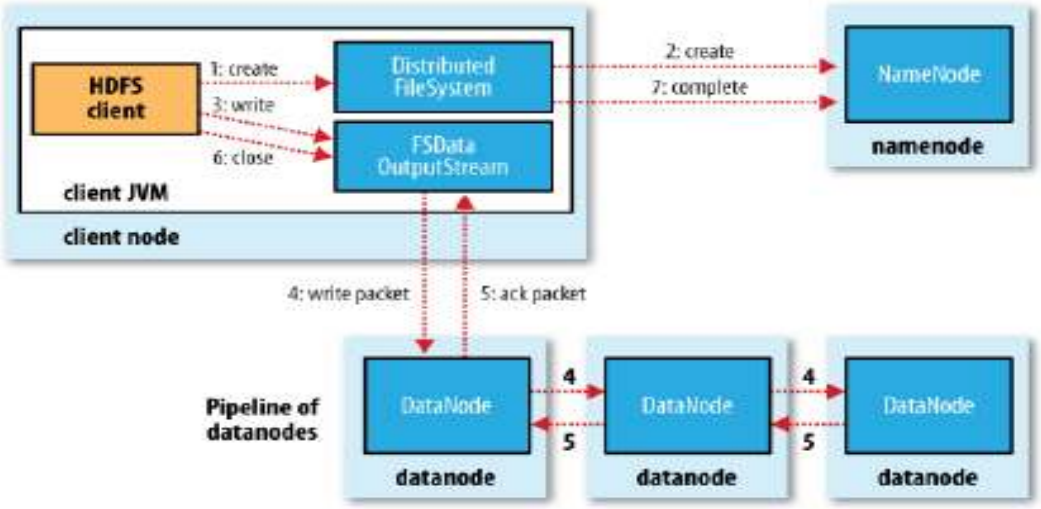
- To read a file in HDFS, a user sends an "open" request to the NameNode to get the location of file blocks.
- For each file block, the NameNode returns the address of a set of DataNodes containing replica information for the requested file.
- The number of addresses depends on the number of block replicas.
- The user calls the read function to connect to the closest DataNode containing the first block of the file.
- Then 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.



HDFS-Write Operation

Writing to a file:

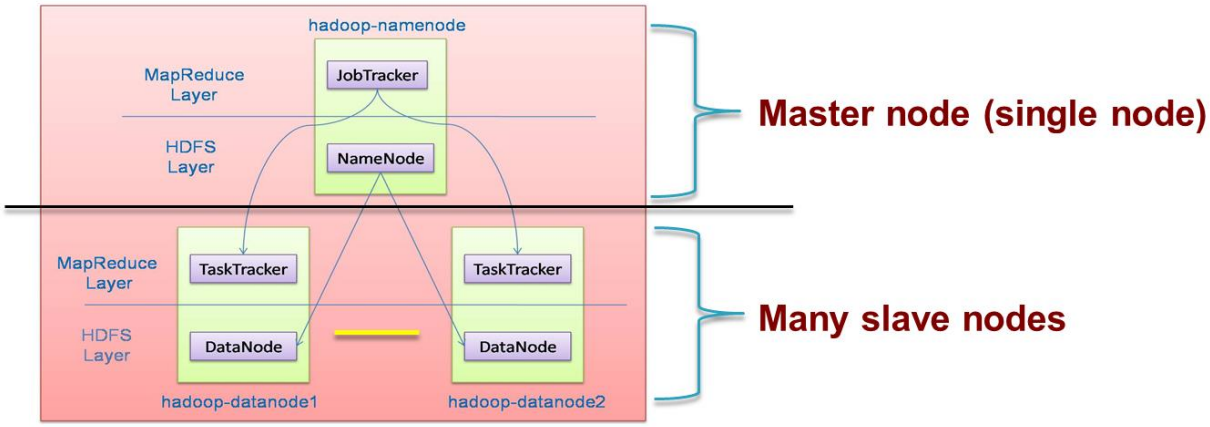
- To write a file in HDFS, a user sends a "create" request to the NameNode to create a new file in the file system namespace.
- If the file does not exist, the NameNode notifies the user and allows him to start writing data to the file by calling the write function.
- The first block of the file is written to an internal queue termed the data queue.
- A data streamer monitors its writing into a DataNode.
- Each file block needs to be replicated by a predefined factor.
- The data streamer first sends a request to the NameNode to get a list of suitable DataNodes to store replicas of the first block.
- The steamer then stores the block in the first allocated DataNode.
- Afterward, the block is forwarded to the second DataNode by the first DataNode.
- The process continues until all allocated DataNodes receive a replica of the first block from the previous DataNode.
- Once this replication process is finalized, the same process starts for the second block.



4

5.1.1 Architecture of Mapreduce in Hadoop

- Distributed file system (HDFS)
- Execution engine (MapReduce)

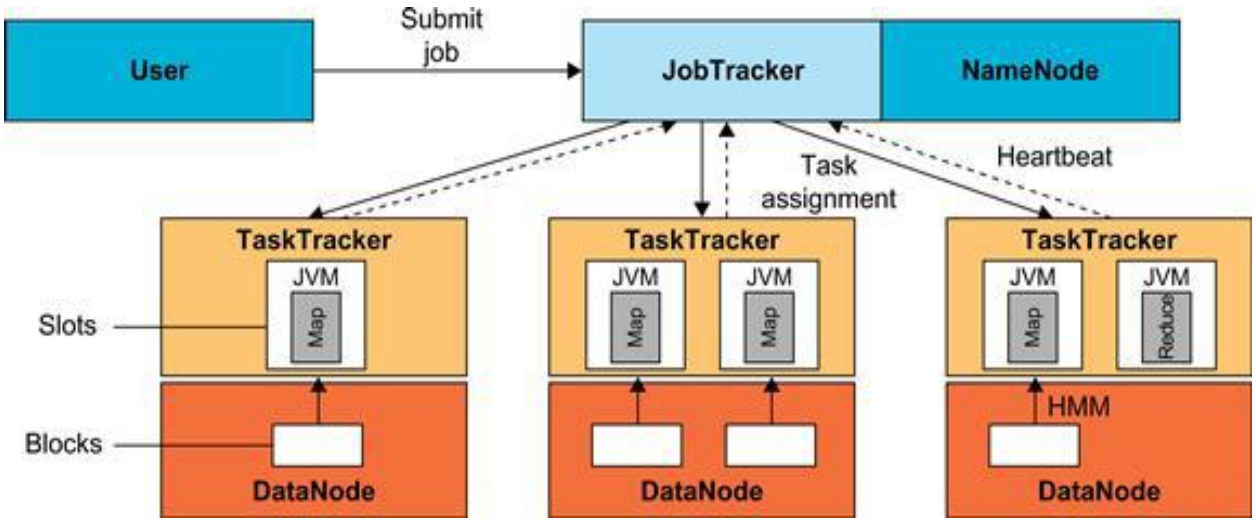


Properties of Hadoop Engine

- 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 NameNode (master) also manages the file system's metadata and namespace.
- Job Tracker is the master node (runs with the namenode)
  - Receives the user's job
  - Decides on how many tasks will run (number of mappers)
  - Decides on where to run each mapper (concept of locality)
- Task Tracker is the slave node (runs on each datanode)
  - Receives the task from Job Tracker
  - Runs the task until completion (either map or reduce task)
  - Always in communication with the Job Tracker reporting progress (heartbeats)

Running a Job in Hadoop



- Three components contribute in running a job in this system:
  - a user node,
  - a JobTracker, and
  - severalTaskTrackers.

- 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
- Job Submission: Each job is submitted from a user node to the JobTracker node.
- Task assignment : The JobTracker creates one map task for each computed input split
- 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.
- Task running check : A task running check is performed by receiving periodic heartbeat messages to the JobTracker from the TaskTrackers.
- Heartbeat:notifies the JobTracker that the sending TaskTracker is alive, and whether the sending TaskTracker is ready to run a new task.

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing, including:

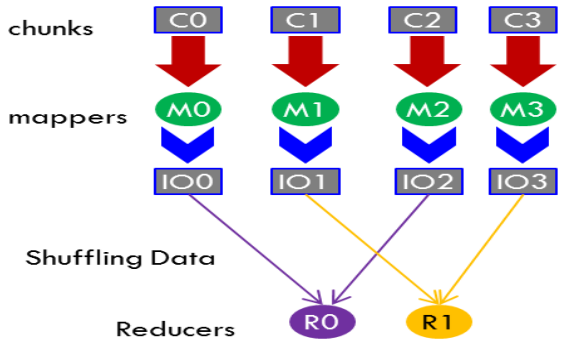
- HadoopCore, our flagship sub-project, provides a distributed filesystem (HDFS) and support for the MapReduce distributed computing metaphor.
- HBase builds on Hadoop Core to provide a scalable, distributed database.
- Pig is a high-level data-flow language and execution framework for parallel computation.It is built on top of Hadoop Core.
- ZooKeeperis a highly available and reliable coordination system. Distributed applications use ZooKeeper to store and mediate updates for critical shared state.
- Hive is a data warehouse infrastructure built on Hadoop Core that provides data summarization, adhoc querying and analysis of datasets.

MAP REDUCE

- MapReduce is a programming model for data processing.
- MapReduce is designed to efficiently process large volumes of data by connecting many commodity computers together to work in parallel
- Hadoop can run MapReduce programs written in various languages like Java, Ruby, and Python
- MapReduce works by breaking the processing into two phases:
    - The map phase and

- The reduce phase.

- Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer.
- The programmer also specifies two functions:
  - The map function and
  - The reduce function.
- In MapReduce, chunks are processed in isolation by tasks called Mappers
- The outputs from the mappers are denoted as intermediate outputs (IOs) and are brought into a second set of tasks called Reducers
- The process of bringing together IOs into a set of Reducers is known as shuffling process
- The Reducers produce the final outputs (FOs)



☐ Overall, MapReduce breaks the data flow into two phases, map phase and reduce phase

Mapreduce Workflow

Application writer specifies

- A pair of functions called Mapper and Reducer and a set of input files and submits the job
- Input phase generates a number of FileSplits from input files (one per Map task)
- The Map phase executes a user function to transform input key-pairs into a new set of key-pairs
- The framework Sorts & Shuffles the key-pairs to output nodes
- The Reduce phase combines all key-pairs with the same key into new keypairs
- The output phase writes the resulting pairs to files as "parts"

Characteristics of MapReduce is characterized by:

- Its simplified programming model which allows the user to quickly write and test distributed systems

- Its efficient and automatic distribution of data and workload across machines
- Its flat scalability curve. Specifically, after a Mapreduce program is written and functioning on 10 nodes, very little-if any- work is required for making that same program run on 1000 nodes

The core concept of MapReduce in Hadoop is that input may be split into logical chunks, and each chunk may be initially processed independently, by a map task. The results of these individual processing chunks can be physically partitioned into distinct sets, which are then sorted. Each sorted chunk is passed to a reduce task.

A map task may run on any compute node in the cluster, and multiple map tasks may berunning in parallel across the cluster. The map task is responsible for transforming the input records into key/value pairs. The output of all of the maps will be partitioned, and each partition will be sorted. There will be one partition for each reduce task. Each partition's sorted keys and the values associated with the keys are then processed by the reduce task. There may be multiple reduce tasks running in parallel on the cluster.

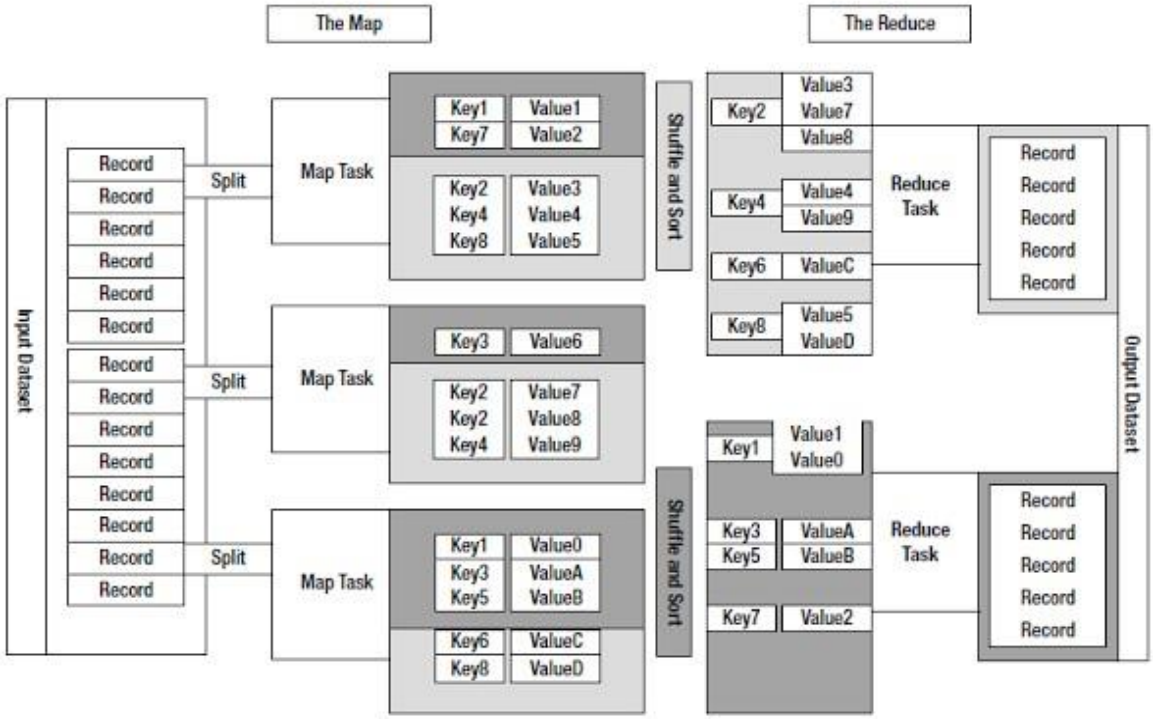
The application developer needs to provide only four items to the Hadoop framework: the class that will read the input records and transform them into one key/value pair per record, a map method, a reduce method, and a class that will transform the key/value pairs that the reduce method outputs into output records.

My first MapReduce application was a specialized web crawler. This crawler received as input large sets of media URLs that were to have their content fetched and processed. The media items were large, and fetching them had a significant cost in time and resources.

The job had several steps:

1. Ingest the URLs and their associated metadata.
2. Normalize the URLs.
3. Eliminate duplicate URLs.
4. Filter the URLs against a set of exclusion and inclusion filters.
5. Filter the URLs against a do not fetch list.
6. Filter the URLs against a recently seen set.
7. Fetch the URLs.
8. Fingerprint the content items.
9. Update the recently seen set.

10. Prepare the work list for the next application.

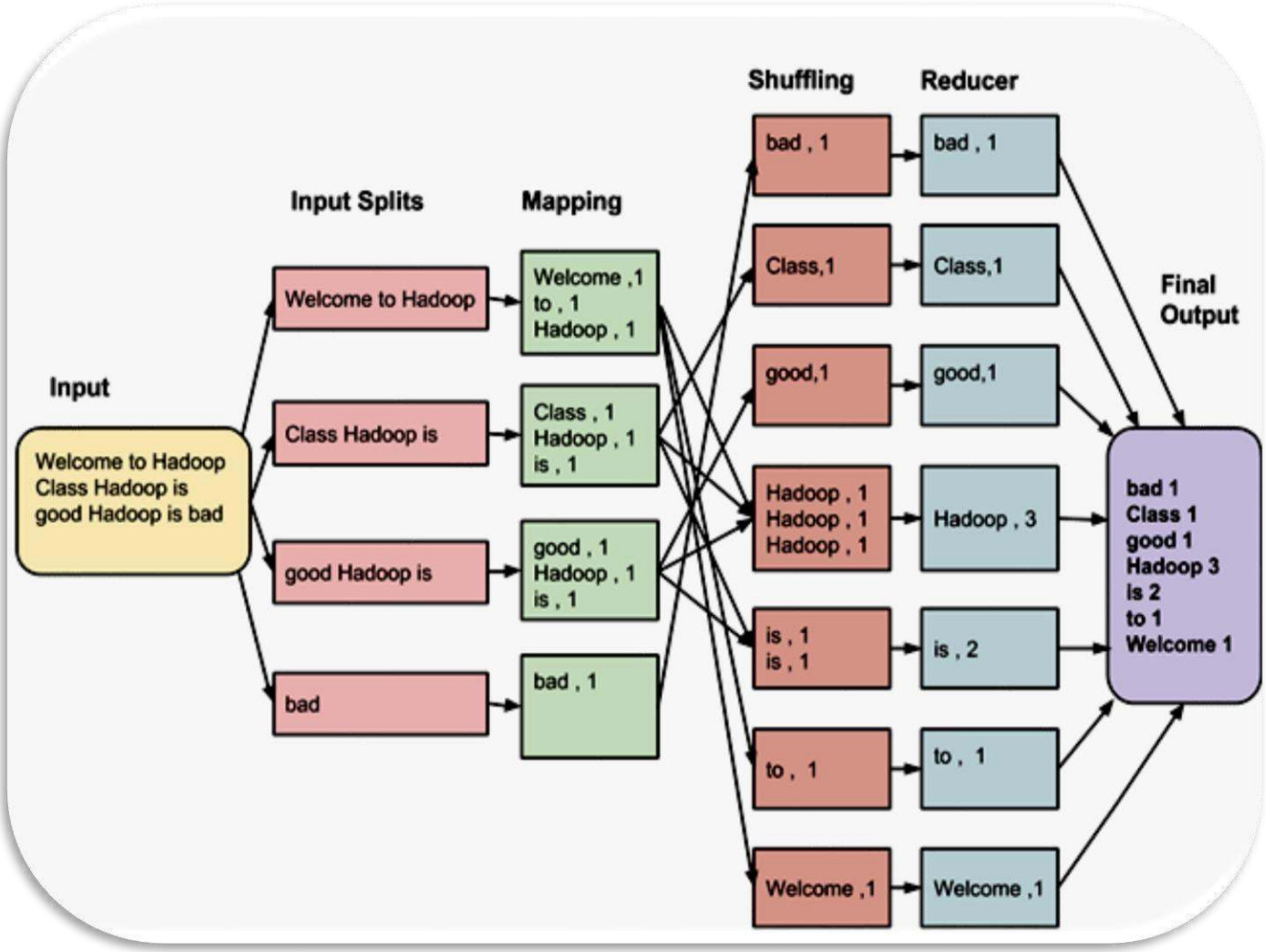


Input File:

Welcome to Hadoop Class

Hadoop is good

Hadoop is bad



5.2 VirtualBox

VirtualBox is a general-purpose full virtualizer for x86 hardware, targeted at server, desktop, and embedded use. Developed initially by Innotek GmbH, it was acquired by Sun Microsystems in 2008, which was, in turn, acquired by Oracle in 2010.

VirtualBox is an extremely feature rich, high-performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. It supports Windows, Linux, Macintosh, Sun Solaris, and FreeBSD. Virtual Box has supported Open Virtualization Format (OVF) since version 2.2.0 (April 2009)

Operating system virtualization allows your computer's hardware to run many operating system images simultaneously. One of the most used instances of this is to test software or applications in a different environment, rather than on a different computer. This can potentially save you quite a bit of money by running multiple servers virtually on one computer.

Pros and Cons of Virtual Box over VMWare

- Virtual Box is a general-purpose full virtualizer for x86 hardware, targeted at server, desktop, and embedded use.
- This product is a Type 2 hypervisor, so it's virtualization host software that runs on an already established operating system as an application.
- With VirtualBox, it's also possible to share your clipboard between the virtualized and host operating system.
- While VMWare functions on Windows and Linux, not Mac, Virtual Box works with Windows, Mac, and Linux computers.
- Virtual Box truly has a lot of support because it's open-source and free. Being open-source means that recent releases are sometimes a bit buggy, but also that they typically get fixed relatively quickly.
- With VMWare player, instead, you have to wait for the company to release an update to fix the bugs.
- Virtual Box offers you an unlimited number of snapshots.
- Virtual Box is easy to install, takes a smaller amount of resources, and is many people's first choice.
- VMWare often failed to detect my USB device Besides, VirtualBox can detect as well as identify USB devices after installing Virtual Box Extension Pack.
- With VirtualBox Guest Addition, files can be dragged and copied between VirtualBox and host.
- VMware outperforms VirtualBox in terms of CPU and memory utilization.
- VirtualBox has snapshots and VMware has rollback points to which you can revert back to in case you break your virtual machine.
- VMware calls it Unity mode and VirtualBox calls it the seamless mode and they both enable you to open application windows on the host machine, while the VM supporting that app is running in the background quietly.

- In the case of VirtualBox, the UI is simple and clean. Your settings are split into Machine Tools and Global Tools and the former is for creating, modifying, starting, stop and deleting virtual machines. VMware, on the other hand, has a much more complicated UI, menu items are named with technical terms which may seem like jargon to average users. This is primarily because the VMware folks cater to cloud providers and server-side virtualizations more.
- In case of VirtualBox, PCIe pass through can be accomplished, although you might have to jump through some hoops. VMware on the other offers excellent customer support and would help you out if you are in a fix.
- VirtualBox is basically a highly secure program that allows users to download and run OS as a virtual machine. With Virtual Box, users are able to abstract their hardware via complete virtualization thus guaranteeing a higher degree of protection from viruses running in the guest OS.
- Virtual Box offers limited support for 3D graphics. And VMWare has a high-level 3D graphics support with DX10 and OpenGL 3.3 support.
- The real advantage of VirtualBox over VMware server lies in its performance. VirtualBox apparently runs faster than VMware server. A timed experiment of an installation of Windows XP as the guest OS took 20 mins in VirtualBox and 35 mins on VMware server. A similar test on the booting time of the guest OS also shows favor to VirtualBox with timing of 45secs compared to 1min 39 secs on VMware server.
- In VirtualBox, the remote file sharing feature is built right in the package. Setting up remote file sharing is easy and you only need to do it once: point the file path to the directory that you want to share.

5.3 GOOGLE APPLICATION ENGINE (GAE)

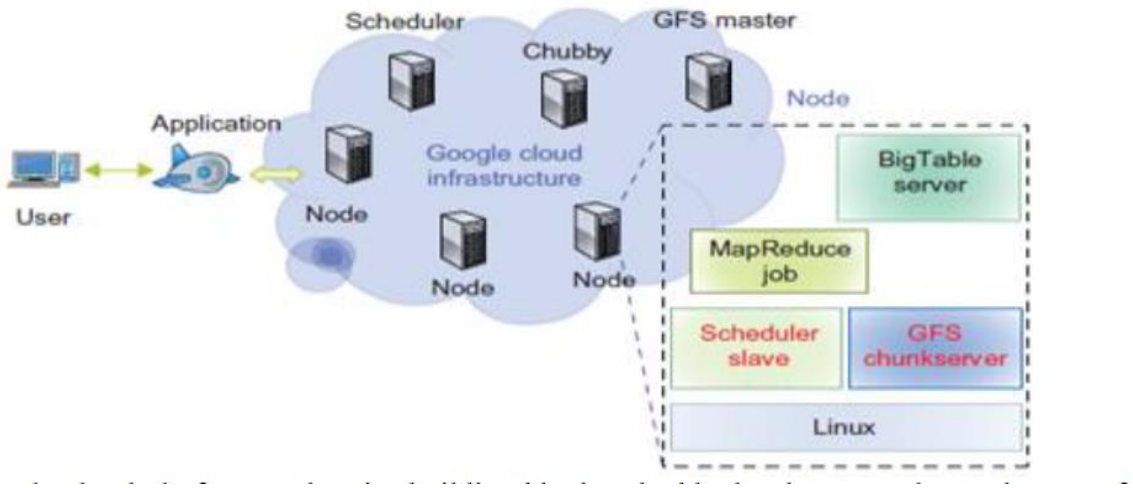
- Google App Engine is a PaaS cloud that provides a complete Web service environment(Platform)
- GAE provides Web application development platform for users.
- All required hardware, operating systems and software are provided to clients.
- Clients can develop their own applications, while App Engine runs the applications on Google's servers.

- GAE helps to easily develop an Web Application
- App Engine only supports the Java and Python programming languages.
- The Google App Engine (GAE) provides a powerful distributed data storage service.

GOOGLE CLOUD INFRASTRUCTURE

- Google has established cloud development by making use of large number of data centers.
- Eg: Google established cloud services in
    - Gmail
    - Google Docs
    - Google Earth etc.
- These applications can support a large number of users simultaneously with High Availability (HA).
- In 2008, Google announced the GAE web application platform.
- GAE enables users to run their applications on a large number of data centers.
- Google App Engine environment includes the following features :
    - Dynamic web serving
    - Persistent(constant) storage with queries, sorting, and transactions
    - Automatic scaling and load balancing
- Provides Application Programming Interface(API) for authenticating users.
- Send email using Google Accounts.
- Local development environment that simulates(create) Google App Engine on your computer.

GAE ARCHITECTURE



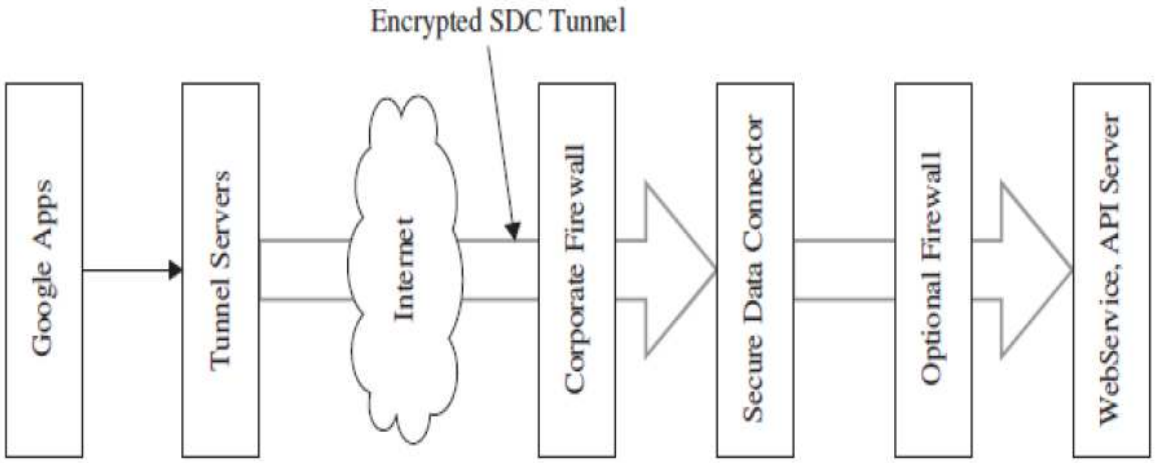
TECHNOLOGIES USED BY GOOGLE ARE

- Google File System(GFS) ->for storing large amounts of data.
- MapReduce->for application program development.
- Chubby-> for distributed application lock services.
- BigTable-> offers a storage service.
- Third-party application providers can use GAE to build cloud applications for providing services.
- Inside each data center, there are thousands of servers forming different clusters.
- GAE runs the user program on Google's infrastructure.
- Application developers now do not need to worry about the maintenance of servers.
- GAE can be thought of as the combination of several software components.
- GAE supports Python and Java programming environments.

FUNCTIONAL MODULES OF GAE

- The GAE platform comprises the following five major components.
- DataStore: offers data storage services based on BigTable techniques.
- The Google App Engine (GAE) provides a powerful distributed data storage service.
- This provides a secure data Storage.

GOOGLE SECURE DATA CONNECTOR (SDC)



FUNCTIONAL MODULES OF GAE

- When the user wants to get the data, he/she will first send an authorized data requests to Google Apps.
- It forwards the request to the tunnel server.
- The tunnel servers validate the request identity.
- If the identity is valid, the tunnel protocol allows the SDC to set up a connection, authenticate, and encrypt the data that flows across the Internet.
- SDC also validates whether a user is authorized to access a specified resource.
- Application runtime environment offers a platform for web programming and execution.
- It supports two development languages: Python and Java.
- Software Development Kit (SDK) is used for local application development.
- The SDK allows users to execute test runs of local applications and upload application code.
- Administration console is used for easy management of user application development cycles.
- GAE web service infrastructure provides special guarantee flexible use and management of storage and network resources by GAE.
- Google offers essentially free GAE services to all Gmail account owners.

- We can register for a GAE account or use your Gmail account name to sign up for the service.
- The service is free within a quota.
- If you exceed the quota, extra amount will be charged.
- Allows the user to deploy user-built applications on top of the cloud infrastructure.
- They are built using the programming languages and software tools supported by the provider (e.g., Java, Python)

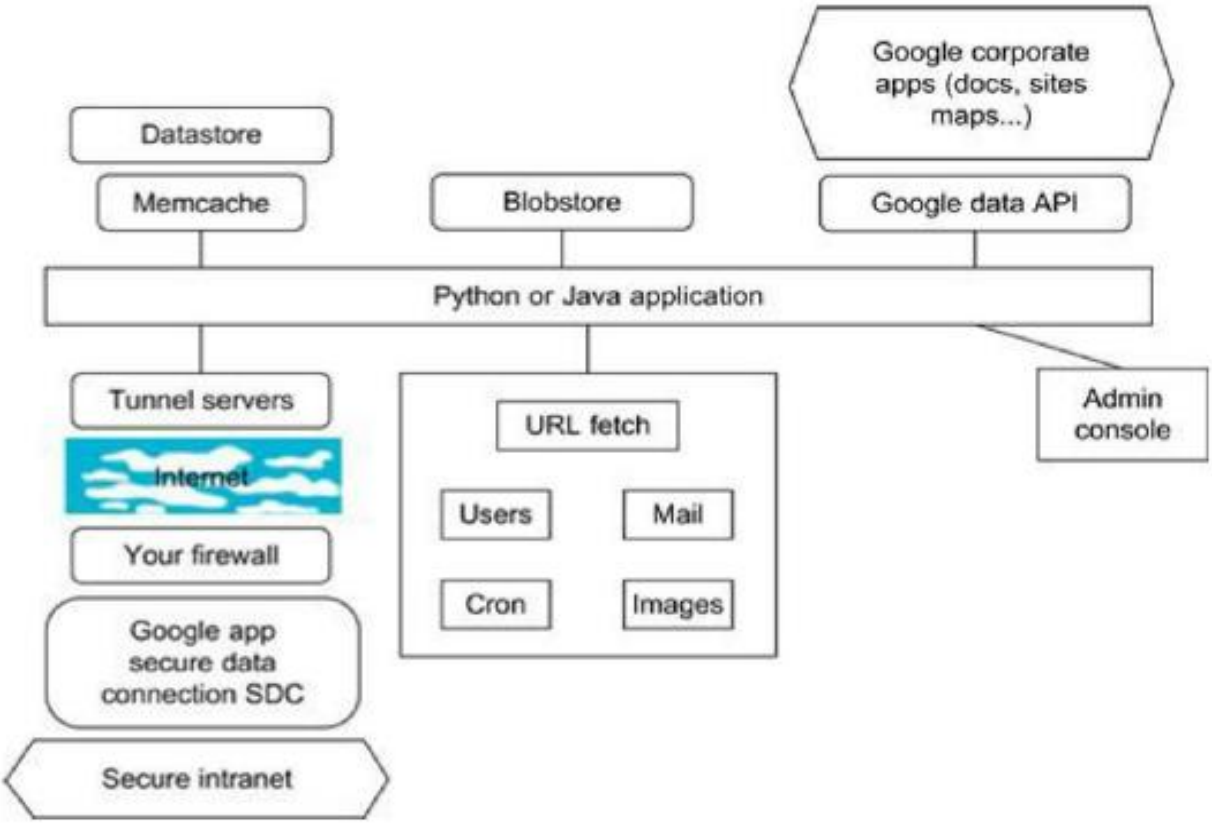
GAE APPLICATIONS

Well-known GAE applications

- Google Search Engine
- Google Docs
- Google Earth
- Gmail
- These applications can support large numbers of users simultaneously.
- Users can interact with Google applications via the web interface provided by each application.
- Applications run in the Google data centers.
- Inside each data center, there might be thousands of server nodes to form different clusters.
- Each cluster can run multipurpose servers.

5. 3.1 Programming Support of Google App Engine

GAE programming model for two supported languages: Java and Python. A client environment includes an Eclipse plug-in for Java allows you to debug your GAE on your local machine. Google Web Toolkit is available for Java web application developers. Python is used with frameworks such as Django and CherryPy, but Google also has webapp Python environment.



There are several powerful constructs for storing and accessing data. The data store is a NOSQL data management system for entities. Java offers Java Data Object (JDO) and Java Persistence API (JPA) interfaces implemented by the Data Nucleus Access platform, while Python has a SQL-like query language called GQL. 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.

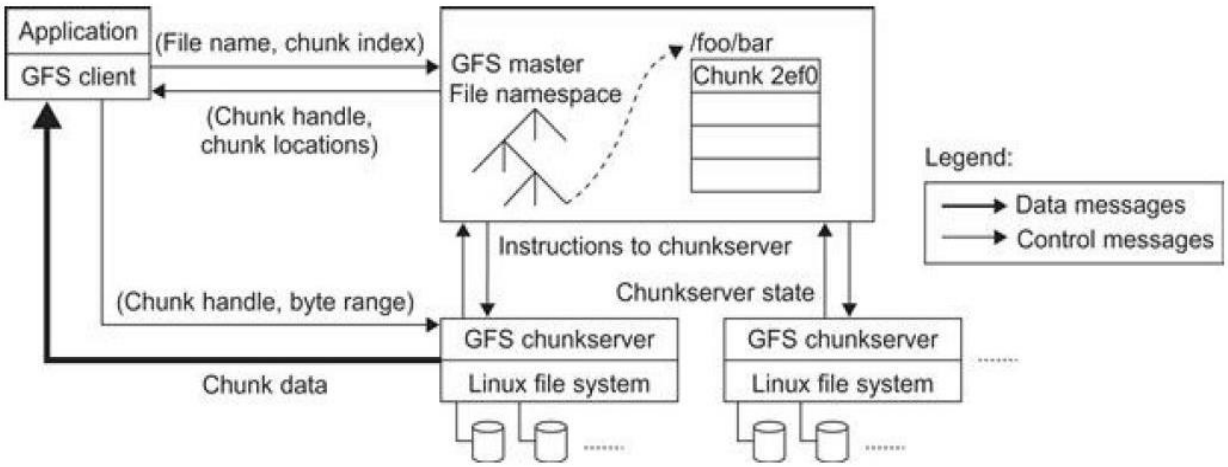
Recently, Google added the blobstore which is suitable for large files as its size limit is 2 GB. There are several mechanisms for incorporating external resources. The Google SDC Secure Data Connection can tunnel through the Internet and link your intranet to an external GAE application. The URL Fetch operation provides the ability for applications to fetch resources and communicate with other hosts over the Internet using HTTP and HTTPS requests.

An application can use Google Accounts for user authentication. Google Accounts handles user account creation and sign-in, and a user that already has a Google account (such as a Gmail account) can use that account with your app. GAE provides the ability to manipulate image data using a dedicated Images service which can resize, rotate, flip, crop, and enhance images. A GAE application is configured to consume resources up to certain limits or quotas. With quotas, GAE ensures that your application won't exceed your budget, and that other applications running on GAE won't impact the performance of your app. In particular, GAE use is free up to certain quotas.

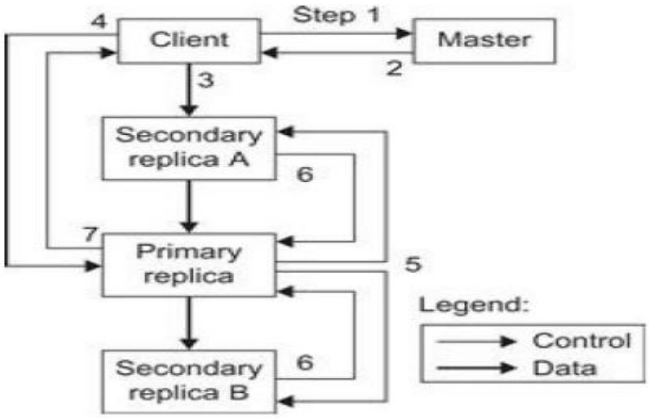
Google File System (GFS)

GFS is a fundamental storage service for Google's search engine. GFS was designed for Google applications, and Google applications were built for GFS. There are several concerns in GFS. rate). As servers are composed of inexpensive commodity components, it is the norm rather than the exception that concurrent failures will occur all the time. Other concerns the file size in GFS. GFS typically will hold a large number of huge files, each 100 MB or larger, with files that are multiple GB in size quite common. Thus, Google has chosen its file data block size to be 64 MB instead of the 4 KB in typical traditional file systems. The I/O pattern in the Google application is also special. Files are typically written once, and the write operations are often the appending data blocks to the end of files. Multiple appending operations might be concurrent. The customized API can simplify the problem and focus on Google applications.

Figure shows the GFS architecture. It is quite obvious that there is a single master in the whole cluster. 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.



The master has enough information to keep the whole cluster in a healthy state. Google uses a shadow master to replicate all the data on the master, and the design guarantees that all the data operations are performed directly between the client and the chunk server. The control messages are transferred between the master and the clients and they can be cached for future use. With the current quality of commodity servers, the single master can handle a cluster of more than 1,000 nodes.



The mutation takes the following steps:

1. The client asks the master which chunk server holds the current lease for the chunk and the locations of the other replicas. If no one has a lease, the master grants one to a replica it chooses (not shown).

2. The master replies with the identity of the primary and the locations of the other (secondary) replicas. The client caches this data for future mutations. It needs to contact the master again only when the primary becomes unreachable or replies that it no longer holds a lease.
3. The client pushes the data to all the replicas. Each chunk server will store the data in an internal LRU buffer cache until the data is used or aged out. By decoupling the data flow from the control flow, we can improve performance by scheduling the expensive data flow based on the network topology regardless of which chunk server is the primary.
4. Once all the replicas have acknowledged receiving the data, the client sends a write request to the primary. The request identifies the data pushed earlier to all the replicas. The primary assigns consecutive serial numbers to all the mutations it receives, possibly from multiple clients, which provides the necessary serialization. It applies the mutation to its own local state in serial order.
5. The primary forwards the write request to all secondary replicas. Each secondary replica applies mutations in the same serial number order assigned by the primary.
6. The secondaries all reply to the primary indicating that they have completed the operation.
7. The primary replies to the client. Any errors encountered at any replicas are reported to the client. In case of errors, the write corrects at the primary and an arbitrary subset of the secondary replicas. The client request is considered to have failed, and the modified region is left in an inconsistent state. Our client code handles such errors by retrying the failed mutation

Big Table

BigTable was designed to provide a service for storing and retrieving structured and semistructured data. BigTable applications include storage of web pages, per-user data, and geographic locations. The database needs to support very high read/write rates and the scale might be millions of operations per second. Also, the database needs to support efficient scans over all or interesting subsets of data, as well as efficient joins of large one-to-one and one-to-many data sets. The application may need to examine data changes over time.

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.

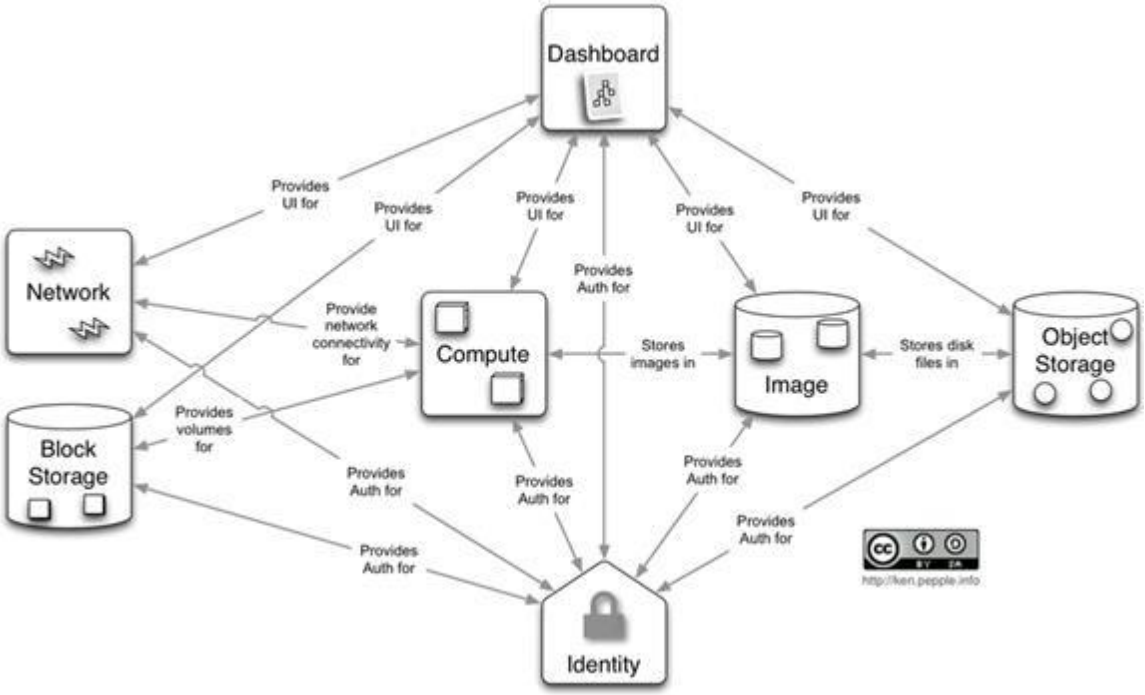
The BigTable system is built on top of an existing Google cloud infrastructure. BigTable uses the following building blocks:

1. GFS: stores persistent state
2. Scheduler: schedules jobs involved in BigTable serving
3. Lock service: master election, location bootstrapping
4. MapReduce: often used to read/write BigTable data.

5.4 Open Stack

- OpenStack is a free and open-source software platform for cloud computing.
- OpenStack is a virtualization tool to manage your virtual infrastructure.
- **OpenStack consists of multiple components.**
  - Compute (Nova)
  - Image Service (Glance)
  - Object Storage (Swift)
  - Dashboard (Horizon)
  - Identity Service (Keystone)
  - Networking (Neutron)
  - Block Storage (Cinder)
  - Telemetry (Ceilometer)
  - Orchestration (Heat)
  - Workflow (Mistral)
  - Database (Trove)
  - Elastic map reduce (Sahara)
  - Bare metal (Ironic)
  - Messaging (Zaqar)
  - Shared file system (Manila)
  - DNS (Designate)
  - Search (Searchlight)
  - Key manager (Barbican)
  - Root Cause Analysis (Vitrage)

- Rule-based alarm actions (Aodh)



Compute (Nova)

 OpenStack Compute is also known as OpenStack Nova.

 Nova is the primary compute engine of OpenStack, used for deploying and managing virtual machine.

 OpenStack Compute manages pools of computer resources and work with virtualization technologies.

 Nova can be deployed using hypervisor technologies such as KVM, VMware, LXC, XenServer, etc.

Image Service (Glance)

 OpenStack image service offers storing and retrieval of virtual machine disk images.

 OpenStack Compute makes use of this during VM provisioning.

 Glance has client-server architecture which allows querying of virtual machine image.

 While deploying new virtual machine instances, Glance uses the stored images as templates.

 OpenStack Glance supports VirtualBox, VMWare and KVM virtual machine images.

Object Storage (Swift)

 OpenStack Swift creates redundant (repetition), scalable data storage to store petabytes of accessible data.

 The data can be included,retrieved and updated.

 It has a distributed architecture, providing greater redundancy, scalability, and performance, with no central point of control.

 It helps organizations to store lots of data safely, cheaply and efficiently.

Dashboard (Horizon)

 OpenStack Horizon is a web-based graphical interface that cloud administrators and users can access tomanage OpenStack compute, storage and networking services.

 To service providers it provides services such as monitoring,billing, and other management tools.

Identity Service (Keystone)

 Provides an authentication and authorization service for other OpenStack services. Provides a catalog of OpenStack Services.

 Keystone provides a central list of users, mapped against all the OpenStack services, which they can access.

 Keystone supports various forms of authentication like standard username & password credentials.

Networking (Neutron)

 Neutron provides networking capability like managing networks and IP addresses for OpenStack.

 OpenStack networking allows users to create their own networks and connects devices and servers to one or more networks.

 Neutron also offers an extension framework, which supports deploying and managing of other network services such as virtual private networks (VPN), firewalls, load balancing, and intrusion detection system (IDS)

Block Storage (Cinder)

 Orchestrates multiple composite cloud applications by using templates.

 It creates and manages service that provides persistent data storage to cloud computing applications.

 Provides persistent block storage to running virtual machine.

 Cinder also provides a self-service application programming interface (API) to enable users to request and consume storage resources.

 A cloud user can manage their storage needs by integrating block storage volumes with Dashboard and Nova.

 It is appropriate for expandable file systems and database storage.

Telemetry (Ceilometer)

 It provides customer billing, resource tracking, and alarming capabilities across all OpenStack core components.

Orchestration (Heat)

 Heat is a service to orchestrate (coordinates) multiple composite cloud applications using templates.

Workflow (Mistral)

 Mistral is a service that manages workflows.

 User typically writes a workflow using workflow language and uploads the workflow definition.

- The user can start workflow manually.

Database (Trove)

 Trove is Database as a Service for OpenStack.

 Allows users to quickly and easily utilize the features of a database without the burden of handling complex administrative tasks.

Elastic map reduce (Sahara)

 Sahara is a component to easily and rapidly provision Hadoop clusters.

 Users will specify several parameters like the Hadoop version number, the cluster topology type, node flavor details (defining disk space, CPU and RAM settings), and others.

Bare metal (Ironic)

 Ironic provisions bare metal machines instead of virtual machines.

Messaging (Zaqar)

 Zaqar is a multi-tenant cloud messaging service for Web developers.

Shared file system (Manila)

 Manila is the OpenStack Shared Filesystems service for providing Shared Filesystems as a service.

 Allows to create, delete, and give/deny access to a file.

DNS (Designate)

 Designate is a multi-tenant API for managing DNS.

Search (Searchlight)

 Searchlight provides advanced and consistent search capabilities across various OpenStack cloud services.

Key manager (Barbican)

 It provides secure storage, provisioning and management of secret data.

 This includes keying material such as Symmetric Keys,Asymmetric Keys and Certificates.

Root Cause Analysis (Vitrage)

 Vitrage is the OpenStack RCA (Root Cause Analysis) service for organizing, analyzing and expanding OpenStack alarms & events, yielding insights regarding the root cause of problems and deducing their existence before they are directly detected.

Rule-based alarm actions (Aodh)

 This alarming service enables the ability to trigger actions based on defined rules against an event data collected by Ceilometer.

5.5 Federation in the cloud

Inter cloud:

- The Inter-Cloud is an interconnected global "cloud of clouds" and an extension of the Internet "network of networks" on which it is based.
- Inter-Cloud computing is interconnecting multiple cloud providers' infrastructures.
- The main focus is on direct interoperability between public cloud service providers.
- To provide cloud services as utility successfully, interconnected clouds are required.
- Interoperability and portability are important factors.
- The limitations of cloud are that they have limited physical resources.
- If a cloud has exhausted all the computational and storage resources, it cannot provide service to the clients.
- The Inter-Cloud environment provides benefits like diverse Geographical locations, better application resilience and avoiding vendor lock-in to the cloud client.
- Benefits for the cloud provider are expand-on-demand and better service level agreements (SLA) to the cloud client.

Types of Inter-Cloud

- ✓ Federation Clouds
- ✓ Multi-Cloud

Federation Clouds

- ☐ A Federation cloud is an Inter-Cloud where a set of cloud providers willingly interconnect their cloud infrastructures in order to share resources among each other.
- The cloud providers in the federation voluntarily collaborate to exchange resources.
- This type of Inter-Cloud is suitable for collaboration of governmental clouds.

- Types of federation clouds are Peer to Peer and Centralized clouds.

Multi-Cloud

- In a Multi-Cloud, a client or service uses multiple independent clouds.
- A multi-cloud environment has no volunteer interconnection and sharing of the cloud service providers' infrastructures.
- Managing resource provisioning and scheduling is the responsibility of client or their representatives.
- This approach is used to utilize resources from both governmental clouds and private cloud portfolios.
- Types of Multi-cloud are Services and Libraries

Cloud Federation

- Provides Federated cloud ecosystem by connecting multiple cloud computing providers using a common standard.
- The combination of disparate things, so that they can act as one.
- Cloud federation refers to the unionization of software infrastructure and platform services from disparate networks that can be accessed by a client.
- The federation of cloud resources is facilitated through network gate ways that connect public or external clouds like private or internal clouds
- It is owned by a single entity and/or community clouds owned by several co-operating entities.
- Creating a hybrid cloud computing environment.
- It is important to note that federated cloud computing services still relay on the existing of physical data centers.

Benefits of cloud federation:

- The federation of cloud resources allows client to optimize enterprise IT service delivery.
- The federation of cloud resources allows a client to choose best cloud service providers
- In terms of flexibility cost and availability of services to meet a particular business or technological need within their organization.
- Federation across different cloud resources pools allows applications to run in the

most appropriate infrastructure environments.

- The federation of cloud resources allows an enterprise to distribute workload around the globe and move data between desperate networks and implement innovative security models for user access to cloud resources

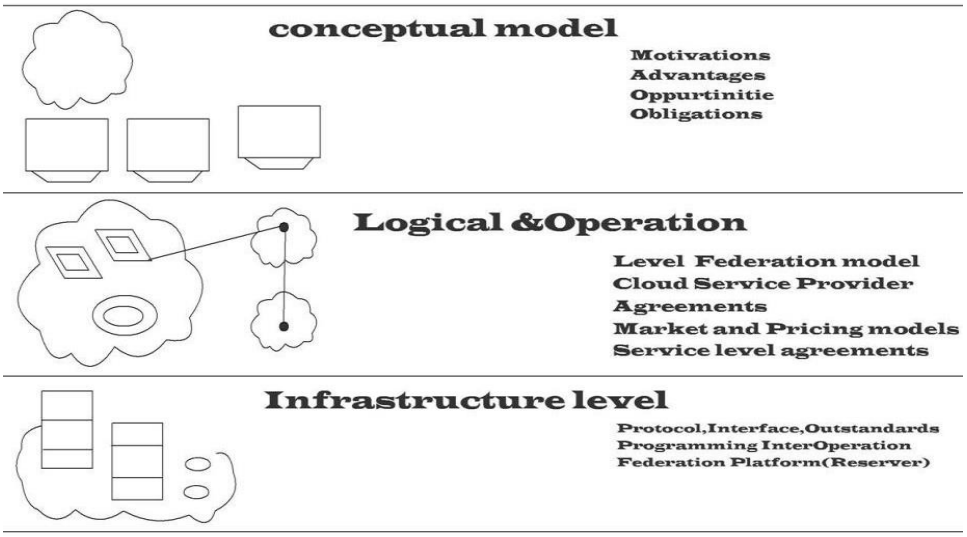
Cloud Federation and Implementation

- One weakness that exist in the federation of cloud resources is the difficulty in programming connectivity.
- Connection between a client and a given external cloud provider is difficult as they each possess their own unique network addressing scheme.
- Cloud providers must grant clients the permission to specify an addressing scheme for each server the cloud provider has external to the internet.
- This provides customers to with the ability to access cloud services without the need for reconfiguration when using resources from different service providers.
- Cloud federation can also be implemented behind a firewall which providing clients with the menu of cloud services provided by one or more trusted entities

5.5.2 Four Levels of Federation:

- Permissive
- Verified
- Encrypted
- Trusted

Permissive Federation:



- Permissive federation occurs when a server accepts a connection from a peer network server without verifying its identity using DNS lookups or certificate checking.
- The lack of verification or authentication may lead to domain spoofing.
- The unauthorized use of a third party domain name in an email message in order to pretend to be someone else), which opens the door to widespread spam and other abuses.

Verified Federation:

- This type of federation occurs when a server accepts a connection from a peer after the identity of the peer has been verified.
- It uses information obtained via DNS and by means of domain-specific keys exchanged beforehand.
- The connection is not encrypted, and the use of identity verification effectively prevents domain spoofing.
- Federation requires proper DNS setup, and that is still subject to DNS poisoning attacks.
- Verified federation has been the default service policy on the open XMPP since the release of the open-source jabberd 1.2 server.
- XMPP-real time communication protocol uses XML.
- Prevent Address spoofing

Encrypted federation:

- Server accepts a connection from a peer if and only if the peer supports Transport Layer Security (TLS) as defined for XMPP in Request for Comments (RFC) 3920.
- The peer must present a digital certificate.
- The certificate may be self-signed, but this prevents using mutual authentication.
- XEP-0220 defines the Server Dialback protocol, which is used between XMPP servers to provide identity verification.
- Server Dialback uses the DNS as the basis for verifying identity.
- The basic approach is that a receiving server receives a server-to- server connection request from an originating server.
- It does not accept the request until it has verified a key with an authoritative server for the domain asserted by the originating server.
- Server Dialback does not provide strong authentication or trusted federation

- Although it is subject to DNS poisoning attacks, it has effectively prevented most instances of address spoofing on the XMPP network

Trusted federation:

- A server accepts a connection from a peer only under the stipulation that the peer supports TLS and the peer can present a digital certificate issued by a root certification authority (CA) that is trusted by the authenticating server.
- The list of trusted root CAs may be determined by one or more factors, such as the operating system, XMPP server software, or local service policy.
- The use of digital certificates results not only in a channel encryption but also in strong authentication.
- The use of trusted domain certificates effectively prevents DNS poisoning attacks.
- But makes federation more difficult, since such certificates have traditionally not been easy to obtain.

5.5.3Federated Services and Applications:

- Clouds typically consist of all the users, devices, services, and applications connected to the network.
- In order to fully leverage the capabilities of this cloud structure, a participant needs the ability to find other entities of interest.
- Such entities might be end users, multiuser chat rooms, real-time content feeds, user directories, data relays, messaging gateways, etc.
- Finding these entities is a process called discovery.
- XMPP uses service discovery (as defined in XEP-0030) to find the aforementioned entities.
- The discovery protocol enables any network participant to query another entity regarding its identity, capabilities, and associated entities.
- When a participant connects to the network, it queries the authoritative server for its particular domain about the entities associated with that authoritative server.
- Then the authoritative server informs the inquirer about services hosted there and may also detail services that are available but hosted elsewhere.
- XMPP includes a method for maintaining personal lists of other entities, known as roster technology, which enables end users to keep track of various types of entities.

Future of Federation:

- The implementation of federated communications is a precursor to building a seamless cloud that can interact with people, devices, information feeds, documents, application interfaces, and other entities.
- It enables software developers and service providers to build and deploy such applications without asking permission from a large, centralized communications operator.
- Many big companies (e.g. banks, hosting companies, etc.) and also many large institutions maintain several distributed data-centers or server-farms, for example to serve to multiple geographically distributed offices, to implement HA, or to guarantee server proximity to the end user. Resources and networks in these distributed data-centers are usually configured as non-cooperative separate elements.
- Many educational and research centers often deploy their own computing infrastructures, that usually do not cooperate with other institutions, except in same punctual situations (e.g. in joint projects or initiatives). Many times, even different departments within the same institution maintain their own non-cooperative infrastructures.
- Cloud end-users are often tied to a unique cloud provider, because of the different APIs, image formats, and access methods exposed by different providers that make very difficult for an average user to move its applications from one cloud to another, so leading to a vendor lock-in problem.
- Many SMEs have their own on-premise private cloud infrastructures to support the internal computing necessities and workloads. These infrastructures are often over-sized to satisfy peak demand periods, and avoid performance slow-down. Hybrid cloud (or cloud bursting) model is a solution to reduce the on-premise infrastructure size, so that it can be dimensioned for an average load, and it is complemented with external resources from a public cloud provider to satisfy peak demands.
- The cloud consumer is often presented with "take-it-or-leave-it standard contracts that might be cost-saving for the provider but is often undesirable for the user". The commission aims to develop with "stakeholders model terms for cloud computing service level agreements for contracts".