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

---

1.1. Introduction to Cloud Computing

Computing is being transformed in to a model consisting of services that are commoditized and delivered a manner similar to utilities such as water, electricity, gas, and telephony. In such a model, users access services based on their requirements, regardless of where the services are hosted.

Several computing paradigms, such as grid computing, have promised to deliver this utility computing vision. Cloud computing is the most recent emerging paradigm promising to turn the vision of "computing utilities" in to a reality.

Cloud computing is a technological advancement that focuses on the way we design computing systems, develop applications, and leverage existing services for building software. It is based on the concept of dynamic provisioning, which is applied not only to services but also to compute capability, storage, networking, and information technology (IT) infrastructure in general.

Resources are made available through the Internet and offered on a pay-per-use basis from cloud Computing vendors. Today, anyone with a credit card can subscribe to cloud services and deploy and configure servers for an application in hours, growing and shrinking the infrastructure serving its application according to the demand, and paying only for the time these resources have been used.

1.2. Defining a cloud

Cloud computing has become a popular buzzword; it has been widely used to refer to different technologies, services, and concepts.

It is often associated with virtualized infrastructure or hard-ware on demand, utility computing, IT outsourcing, platform and software as a service, and many other things that now are the focus of the IT industry.

The term cloud has historically been used in the telecommunications industry as an abstraction of the network in system diagrams. It then became the symbol of the most popular computer network "the Internet".

Cloud computing refers to an Internet-centric way of computing. The Internet plays a fundamental role in cloud computing, since it represents either the medium or the platform through which many cloud computing services are delivered and made accessible.

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and system software in the datacenters that provide those services.

This definition describes cloud computing as a phenomenon touching on the entire stack: from the underlying hardware to the high-level software services and applications.

It introduces the concept of everything as a service, where the different components of a system—IT infrastructure, development platforms, databases, and soon—can be delivered, measured, and consequently priced as a service.

This new approach significantly influences not only the way that we build software but also the way we deploy it, make it accessible, and design our IT infrastructure, and even the way companies allocate the costs for IT needs.

The approach fostered by cloud computing is global: it covers both the needs of a single user hosting documents in the cloud and as well as the entire corporate IT infrastructure in the public cloud.

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

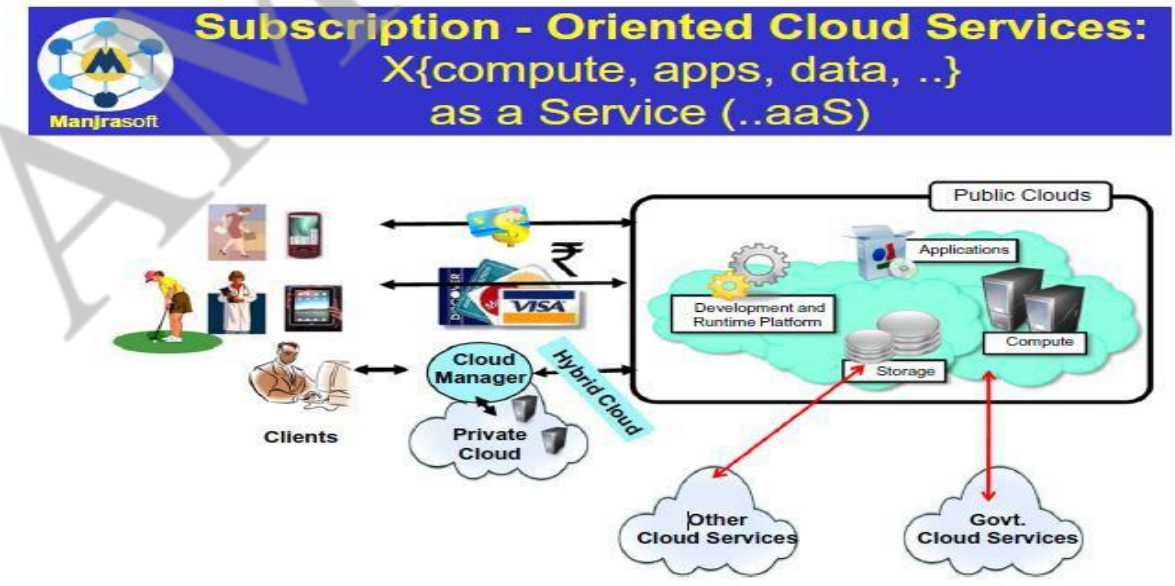


Fig: View of Cloud Computing

Another important aspect of cloud computing is its utility-oriented approach. More than any other trend in distributed computing, cloud computing focuses on delivering services with a given pricing model, in most cases a "pay-per-use" strategy. It makes it possible to access online storage, rent virtual hardware, or use development platforms and pay only for their effective usage, with no or minimal up-front costs.

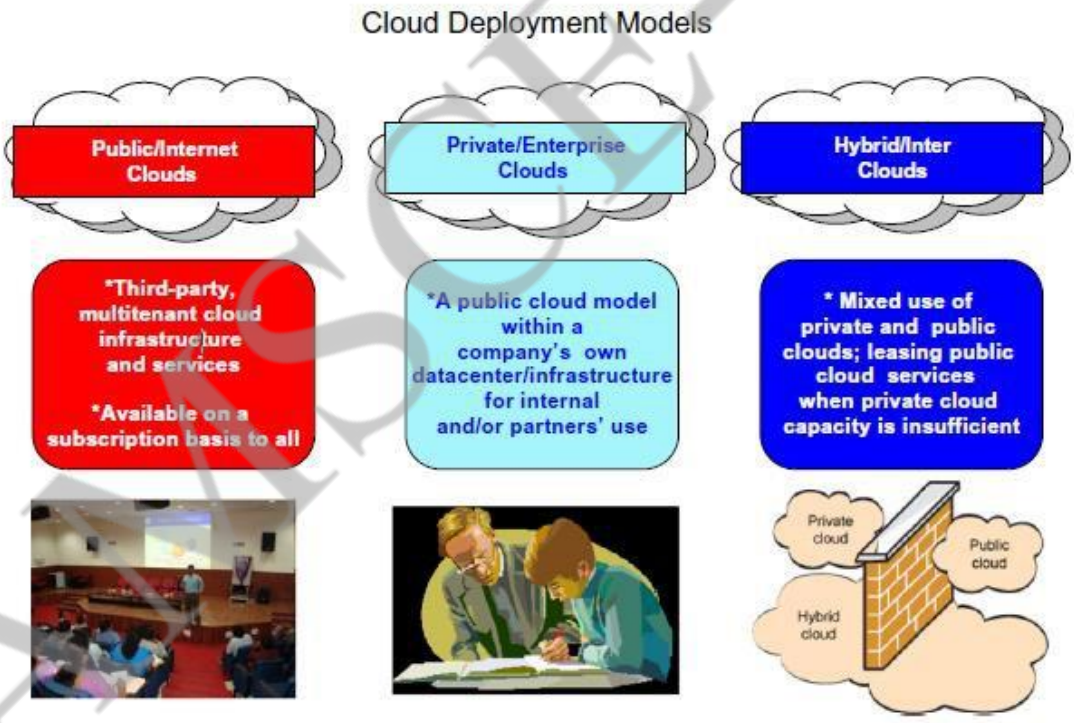
All these operations can be performed and billed simply by entering the credit card details and accessing the exposed services through a Web browser. This helps us provide a different and more practical characterization of cloud computing.

We can define three criteria to discriminate whether a service is delivered in the cloud computing style:

• The service is accessible via a Web browser (nonproprietary) or a Web services application programming interface (API).

• Zero capital expenditure is necessary to get started.

• You pay only for what you use as you use it.



There are three major models for deploying and accessing cloud computing environments. They are public clouds, private/enterprise clouds, and hybrid clouds.

Private Cloud: This is a form of cloud computing platform that is implemented within the corporate firewall, under the control of the IT department.

Public Cloud: it is a form of cloud computing in which a company relies on a third-party cloud service provider for services such as servers, data storage and applications, which are delivered to the company through the Internet.

Hybrid Cloud: It is composed of public cloud resources and privately owned infra- structures, are created to serve the organization's needs.

Even though many cloud computing services are freely available for single users, enterprise-class services are delivered according a specific pricing scheme. In this case users subscribe to the service and establish with the service provider a service-level agreement(SLA) defining the Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service(IaaS) and so on are quality-of-service parameters under which the service is delivered.

The utility-oriented nature of cloud computing is expressed as: A cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers.

1.3. Evolution of Cloud Computing:

The idea of renting computing services by leveraging large distributed computing facilities has been around for longtime. It dates back to the days of the mainframes in the early 1950s. From there on, technology has evolved and been refined. This process has created a series of favorable conditions for the realization of cloud computing.

In tracking the historical evolution, there are five core technologies that played an important role in the realization of cloud computing.

These technologies are distributed systems, virtualization, Web2.0, service orientation, and utility computing.

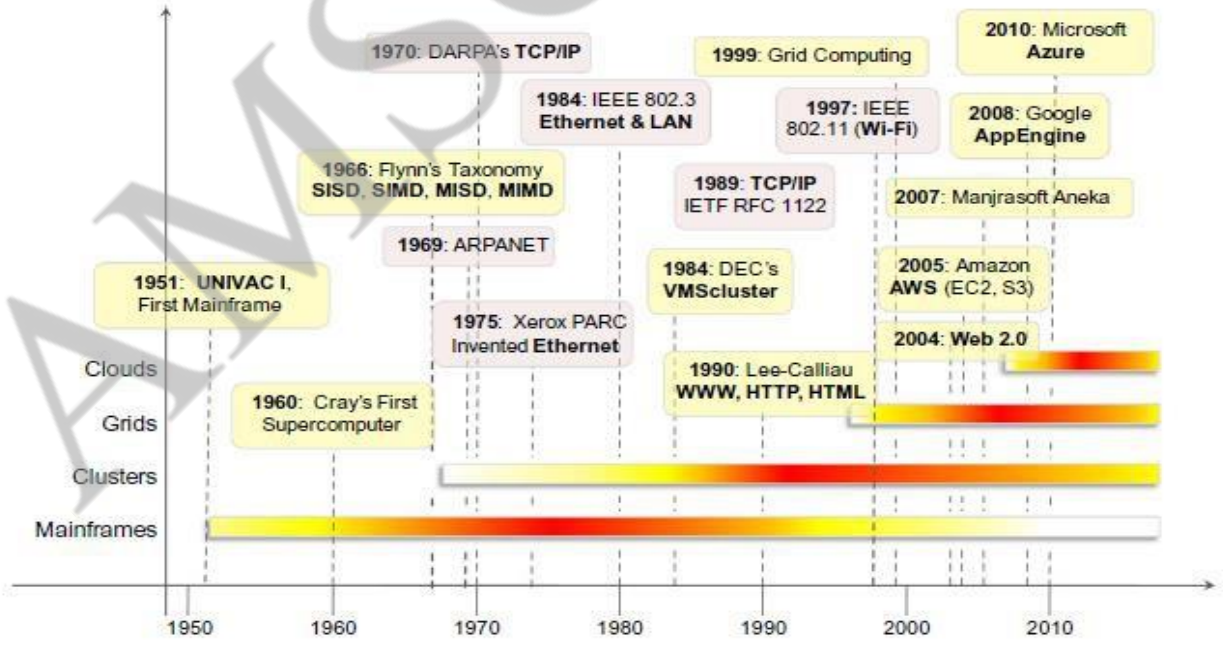
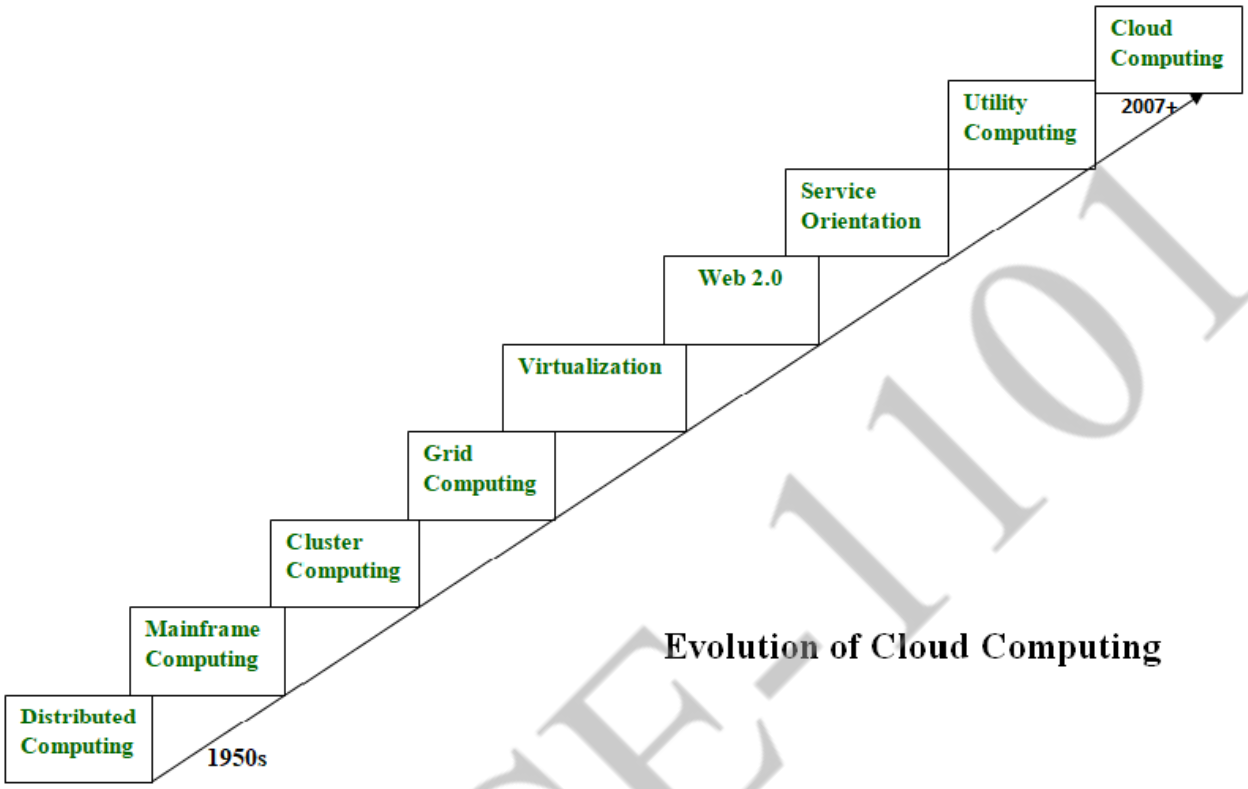


Fig**: Evolution of Distributed Computing Technologies**

The above diagram shown an overview of the evolution of the distributed computing technologies that have influenced cloud computing.



Evolution of Cloud Computing

1.3.1. Distributed systems

Clouds are essentially large distributed computing facilities that make available their services to third parties on demand. A distributed system is a collection of independent computers that appears to its users as a single coherent system.

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.

- The primary purpose of distributed systems is to share resources and utilize them better.
- Distributed systems often exhibit other properties such as heterogeneity, openness, scalability, transparency, concurrency, continuous availability, and independent failures.
- To some extent these also characterize clouds, especially in the context of scalability, concurrency, and continuous availability.
- Clouds are essentially large distributed computing facilities that make available their services to third parties on demand.

Three major milestones have led to cloud computing are:

- Mainframe computing,
- Cluster computing, and
- Grid computing.

Main frame Computing:

These were the first examples of large computational facilities leveraging multiple processing units. Mainframes were powerful, highly reliable computers specialized for large data movement and massive input/output (I/O) operations. They were mostly used by large organizations for bulk data processing tasks such as online transactions, enterprise resource planning, and other operations involving the processing of significant amounts of data. Even though mainframes cannot be considered distributed systems, they offered large computational power by using multiple processors, which were presented as a single entity to users.

One of the most attractive features of mainframes was the ability to be highly reliable computers that were "always on" and capable of tolerating failures transparently. No system shutdown was required to replace failed components, and the system could work without interruption. Batch processing was the main application of mainframes. Now their popularity and deployments have reduced, but evolved versions of such systems are still in use for transaction processing (such as online banking, airline ticket booking, supermarket and telcos, and government services).

Cluster Computing:

- Cluster computing started as a low-cost alternative to mainframes and supercomputers.
- Mainframe and super computers eventually generated an increased availability of cheap commodity machines as a side effect.
- In 1980s, clusters become the standard technology for parallel and high-performance computing. They were i) cheaper than mainframes ii)made high-performance computing available to a large number of groups, including universities and small research labs.

"A computer cluster consists of a set of loosely or tightly connected computers that work together in many respect, they can be viewed as a single system."

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.

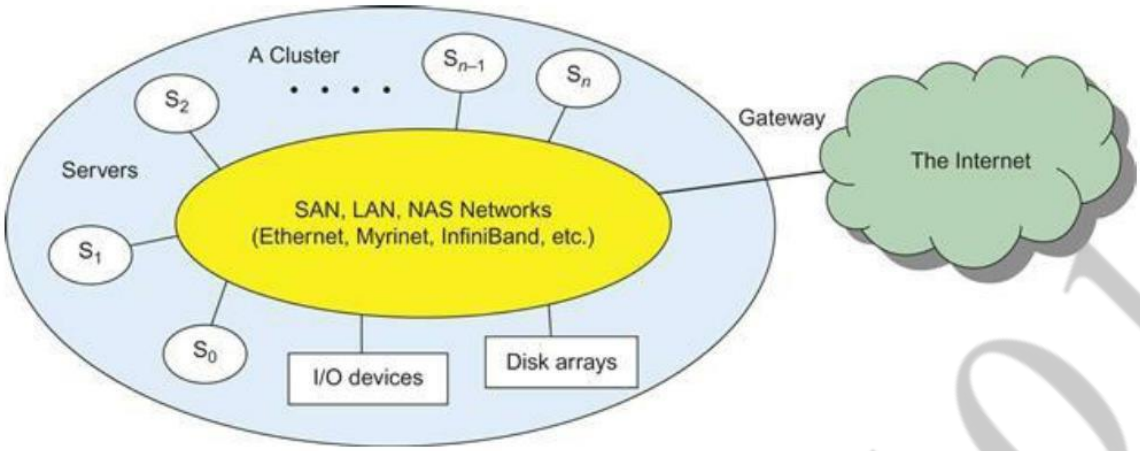


Fig: Cluster Architecture

This network can be as simple as a SAN (e.g., Myrinet) or a LAN (e.g., Ethernet). To build a larger cluster with more nodes, the interconnection network can be built with multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches. Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes. The cluster is connected to the Internet via a virtual private network (VPN) gateway. The gateway IP address locates the cluster. The system image of a computer is decided by the way the OS manages the shared cluster resources. Most clusters have loosely coupled node computers. All resources of a server node are managed by their own OS. Thus, most clusters have multiple system images as a result of having many autonomous nodes under different OS control.

- Cluster technology contributed considerably to the evolution of tools and frameworks for distributed computing, Which includes, Condor – open source, high throughput computing. Used to mange workload on a dedicated cluster of computers. Parallel Virtual Machine (PVM) and (tool for parallel and heterogeneous network of computers).Message Passing Interface (MPI) – message passing standard developed for parallel computing architectures.

Features of clusters :

- Computational power could be leveraged (use something to maximum advantage) to solve problems that were previously manageable only on expensive supercomputers.
- Clusters could be easily extended if more computational power was required.

Grid Computing:

- Grid computing appeared in the early 1990s as an evolution of cluster computing.
- In an analogy to the power grid, grid computing proposed a new approach to access large computational power, huge storage facilities, and a variety of services.
- Grids initially developed as aggregations of geographically dispersed (each location is distant from each other) clusters by means of Internet connections. These clusters belonged to different organizations, and arrangements were made among them to share the

computational power (speed that instructions are carried out- may be sometimes memory and bandwidth).

- Different from a "large cluster" a computing grid was a dynamic aggregation of heterogeneous computing nodes, and its scale was nationwide or even worldwide.

Like an electric utility power grid, a ***computing grid*** offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together. The grid is often constructed across LAN, WAN, or Internet backbone networks at a regional, national, or global scale. Enterprises or organizations present grids as integrated computing resources. They can also be viewed as ***virtual platforms*** to support ***virtual organizations***. The computers used in a grid are primarily workstations, servers, clusters, and supercomputers. Personal computers, laptops, and PDAs can be used as access devices to a grid system.

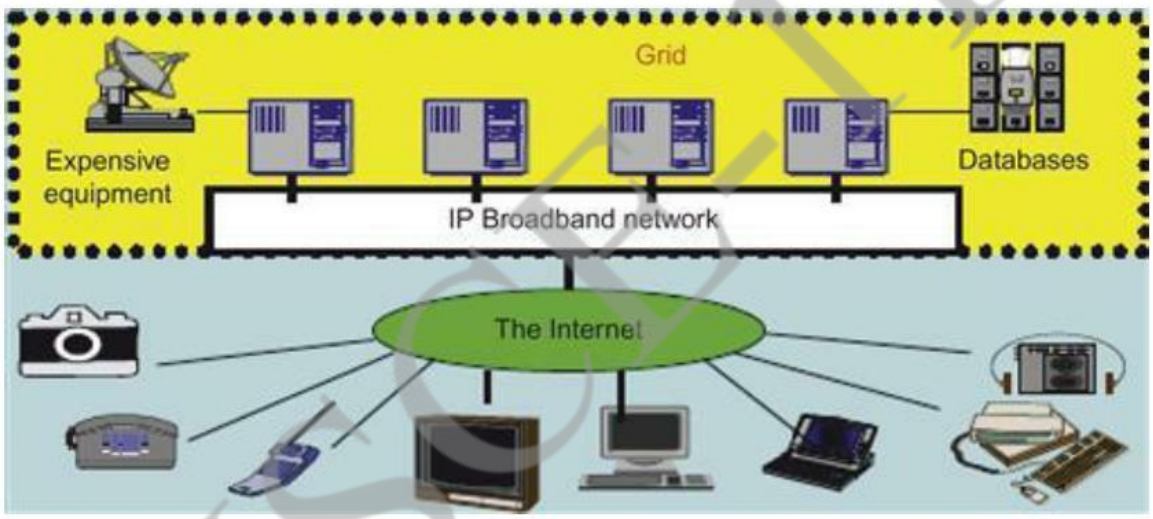


Fig: **Computational** ***grid or data grid providing computing utility, data, and information services through resource sharing and cooperation among participating organizations.***

Hardware virtualization and multicore chips enable the existence of dynamic configurations in the cloud. Utility and grid computing technologies lay the necessary foundation for computing clouds. Recent advances in SOA, Web 2.0, and mashups of platforms are pushing the cloud another step forward. Finally, achievements in autonomic computing and automated data center operations contribute to the rise of cloud computing.

- Several developments made possible the diffusion(spread something more widely) of computing grids:

(a)Clusters became quite common resources;

(b)They were often under utilized;

(c)New problems were requiring computational power that went beyond the capability of single clusters;and

(d)The improvements in networking and the diffusion of the Internet made possible long-distance, high-bandwidth connectivity.

Grid computing is envisioned to allow close interaction among applications running on distant computers simultaneously.

Grid Computing is a technology for coordinating large scale resource sharing and problem solving among various autonomous group. Grid technologies have issues in QoS, data management, scheduling, resource allocation, accounting and performance. Grid applications must demand:

1. High performance transport protocol for bulk data transfer
2. Performance controllability
3. Dynamic network resource allocation and reservation
- Security
- High availability
- Multicast to efficiently distribute data to group of resources

Cloud computing:

Cloud computing is often considered the successor of grid computing. In reality, it embodies aspects of all these three major technologies. Computing clouds are deployed in large datacenters hosted by a single organization that provides services to others. Clouds are characterized by the fact of having virtually infinite capacity, being tolerant to failures, and being always on, as in the case of mainframes. In many cases, the computing nodes that form the infrastructure of computing clouds are commodity machines, as in the case of clusters. The services made available by a cloud vendor are consumed on a pay-per-use basis, and clouds fully implement the utility vision introduced by grid computing.

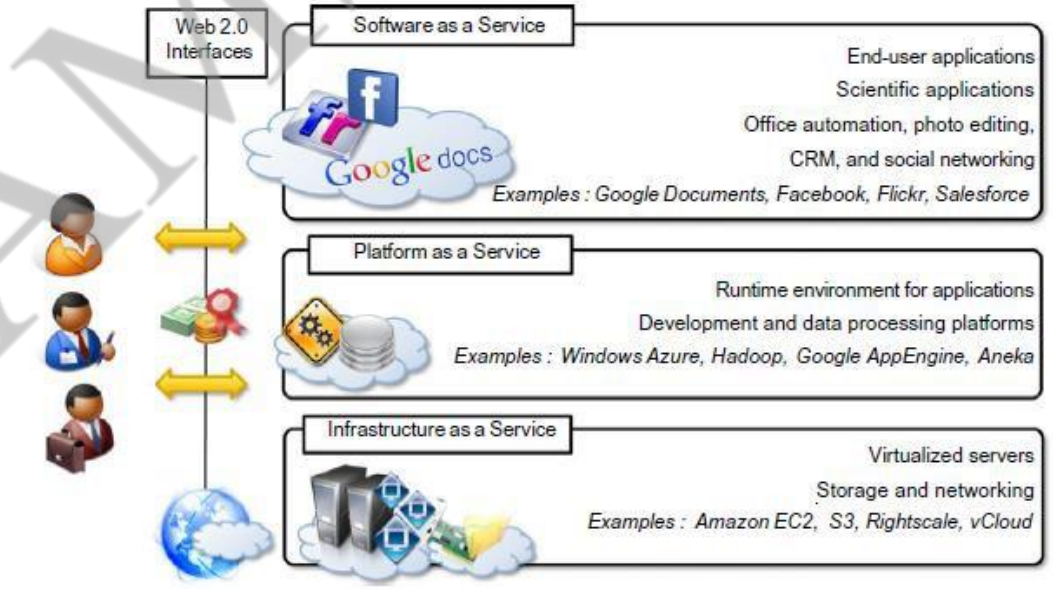


Fig: Cloud Computing Reference Model

Infrastructure providers offer simple methods to provision customized hardware and integrate it into existing systems.

Platform-as-a-Service providers offer runtime environment and programming models that are designed to scale applications.

Software-as-a-Service offerings can be elastically sized on demand without requiring users to provision hardware or to program application for scalability.

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 evolution from Internet to web and grid services is certainly playing a major role in this growth.

1.3.2. Virtualization

Virtualization is another core technology for cloud computing. It encompasses a collection of solutions allowing the abstraction of some of the fundamental elements for computing, such as hardware, runtime environments, storage, and networking.

Virtualization has become a fundamental element of cloud computing. This is particularly true for solutions that provide IT infrastructure on demand. Virtualization confers that degree of customization and control that makes cloud computing appealing for users and, at the same time, sustainable for cloud services providers.

Virtualization is a technology that allows creation of different computing environments. These environments are called virtual because they simulate the interface that is expected by a guest.

The most common example of virtualization is hardware virtualization. This technology allows simulating the hardware interface expected by an operating system. **Hardware virtualization** allows the coexistence of different software stacks on top of the same hardware. These stacks are contained inside virtual machine instances, which operate in complete isolation from each other.

Virtualization technologies are also used to replicate runtime environments for programs. Applications in the case of process virtual machines (which include the foundation of technologies such as Javaor.NET), instead of being executed by the operating system, are run by a specific program called a **virtual machine**.

1.3.3. Web 2.0

The Web is the primary interface through which cloud computing delivers its services. At present,

the Web encompasses a set of technologies and services that facilitate interactive information sharing, collaboration, user-centered design, and application composition. This evolution has transformed the Web in to a rich platform for application development and is known as **Web 2.0.**

Web 2.0 brings interactivity and flexibility in to Web pages, providing enhanced user experience by gaining Web-based access to all the functions that are normally found in desktop applications**.**

Web2.0 applications are extremely dynamic: they improve continuously, and new updates and features are integrated at a constant rate by following the usage trend of the community. There is no need to deploy new software releases on the installed base at the client side. Users can take advantage of the new software features simply by interacting with cloud applications.

Examples of Web2.0 applications are Google Documents, Google Maps, Flickr, Facebook, Twitter, YouTube, de.li.cious, Blogger, and Wikipedia. In particular, social networking Websites take the biggest advantage of Web2.0. The level of interaction in Web sites such as Facebook or Flickr would not have been possible without the support of AJAX, Really Simple Syndication (RSS), and other tools that make the user experience incredibly interactive.

1.3.4. Service-oriented computing(SOC)

Service orientation is the core reference model for cloud computing systems. This approach adopts the concept of services as the main building blocks of application and system development.

Service-orientedcomputing(SOC) supports the development of rapid, low-cost, flexible, interoperable, and evolvable applications and systems.

A service is an abstraction representing a self-describing and platform-agnostic component thatcan perform any function—anything from a simple function to a complex business process. A service is supposed to be loosely coupled, reusable, programming language independent, and location transparent.

Service-oriented computing introduces and diffuses two important concepts, which are also fundamental to cloud computing: quality of service(QoS) and Software-as-a-Service(SaaS).

• Quality of service(QoS) identifies a set of functional and non functional attributes that can be used to evaluate the behavior of a service from different perspectives. These could be performance metrics such as responsetime, or security attributes, transactional integrity, reliability, scalability, and availability.

• The concept of Software-as-a-Service introduces a new delivery model for applications. The term has been inherited from the world of application service providers(ASPs), which deliver software services- based solutions across the wide area network from a central data center and make them available on a subscription or rental basis.

1.3.5. Utility Oriented Computing

Utility computing is a vision of computing that defines a service-provisioning model for compute Services in which resources such as storage, compute power, applications, and infrastructure are Packaged and offered on a pay-per-use basis. The idea of providing computing as a utility like natural gas, water, power, and telephone connection has a long history but has become a reality today with the advent of cloud computing.

The idea of computing as utility remained and extended from the business domain to academia with the advent of cluster computing. Not only businesses but also research institutes became acquainted with the idea of leveraging an external IT infrastructure on demand.

The capillary diffusion of the Internet and the Web provided the technological means to realize utility computing on a worldwide scale and through simple interfaces. Computing grids brought the concept of utility computing to a new level: market orientation.

With utility computing accessible on a wider scale, it is easier to provide a trading infrastructure where grid products—storage, computation, and services—are bid for or sold. Moreover, e-commerce technologies provided the infrastructure support for utility computing.

1.4. Underlying Principles of Parallel and Distributed Computing

The fundamental principles of parallel and distributed computing , various models and conceptual frameworks serve as foundations for building cloud computing systems and applications.

1.4.1. Eras of computing

Two fundamental and dominant models of computing are ***sequential*** and ***parallel***. The sequential computing era began in the 1940s; the 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 enabled the creation of system software such as compilers and operating systems – which support the development of applications.

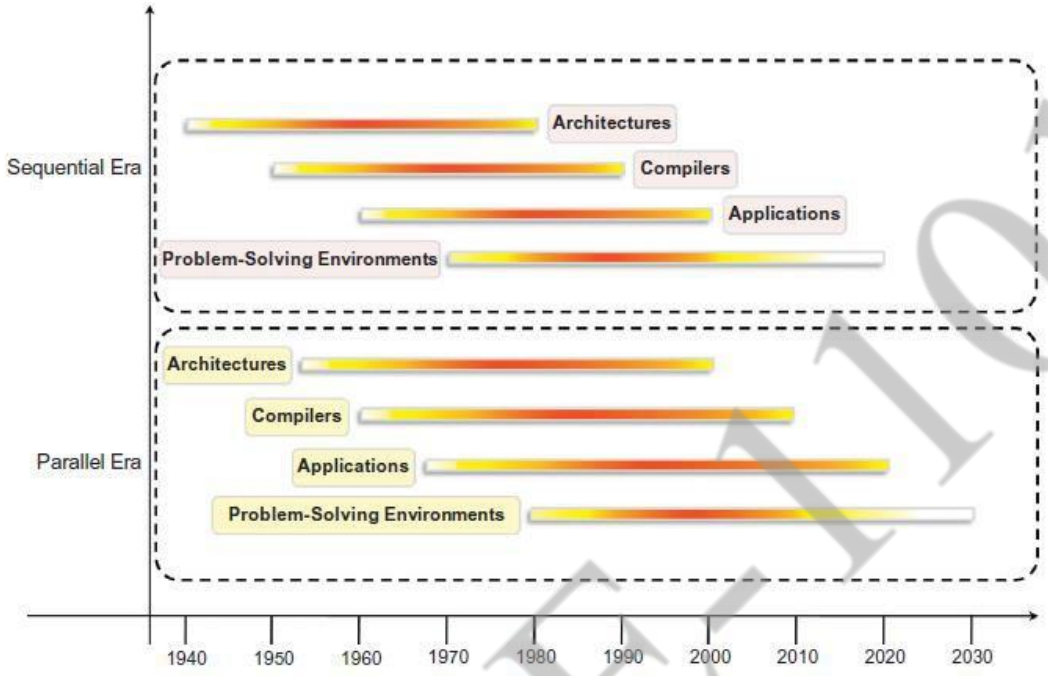


Fig: Eras of Computing

1.4.2. Parallel vs. distributed computing

- 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

1.4.3. 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.
  - The term parallel computing has blurred its edges with the term distributed computing and is often used in place of later term.

1.4.3.1. 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.

- 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: there is already significant R&D work on development tools and environments.
  - Significant development in networking technology is paving the way for heterogeneous computing.

1.4.3.2. 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

Single-instruction, single-data (SISD) systems

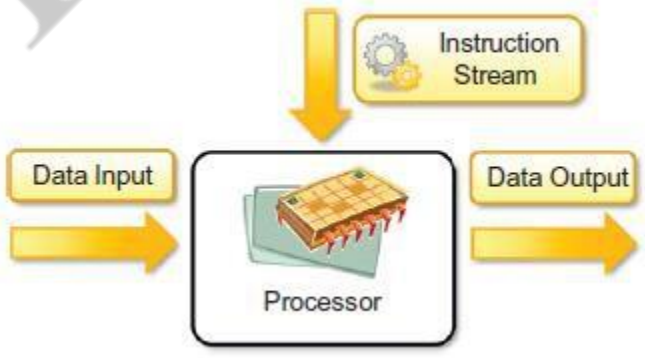


Fig: Single-instruction, single-data (SISD) Architecture

- 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.

Single-instruction, multiple-data (SIMD) systems

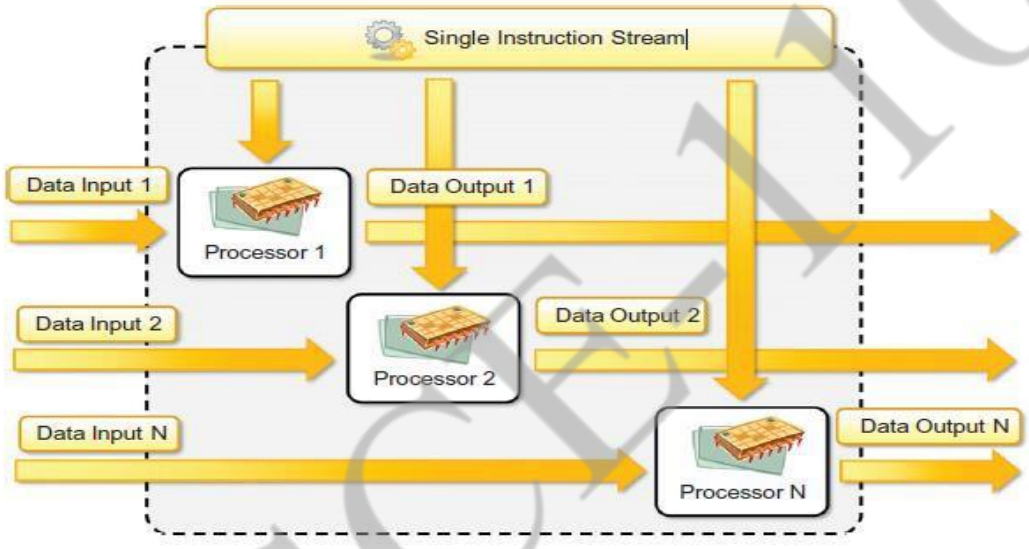


Fig: Single-instruction, multiple-data (SIMD) Architecture

- 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.

Multiple-instruction, single-data (MISD) systems

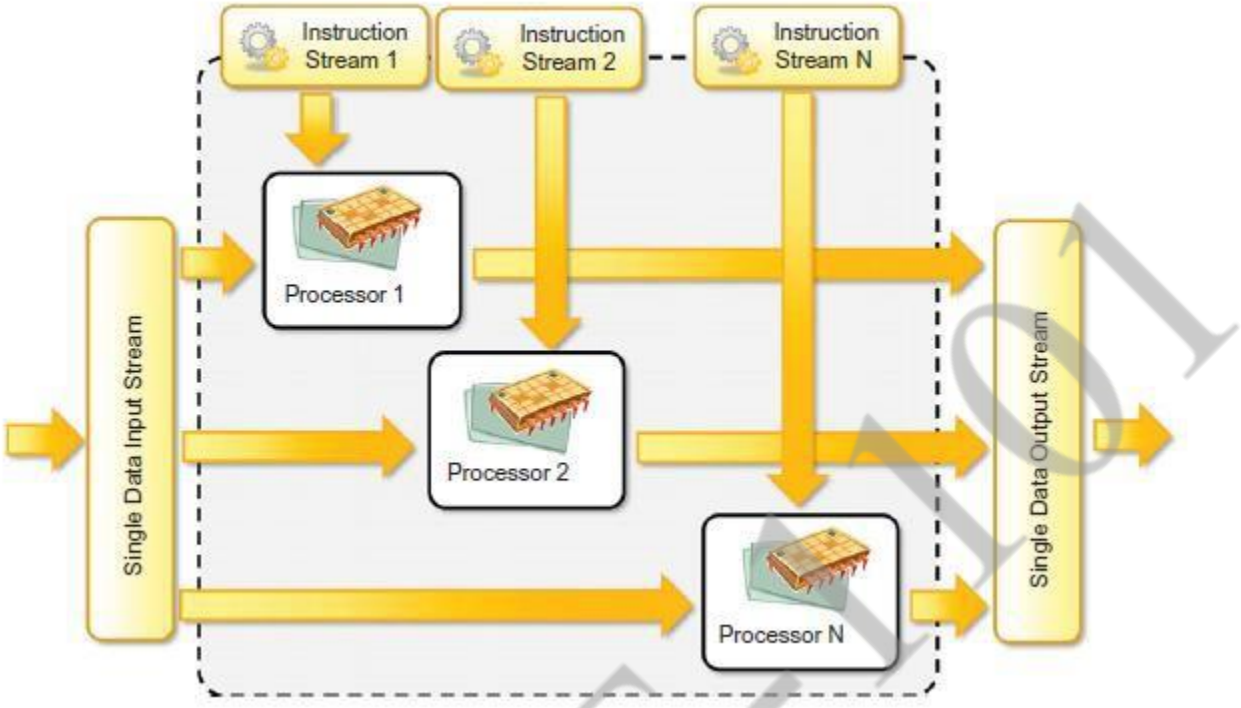


Fig: Multiple-instruction, single-data (MISD) Architecture

- 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.
- For example
  - $y = \sin(x) + \cos(x) + \tan(x)$
- 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.

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.

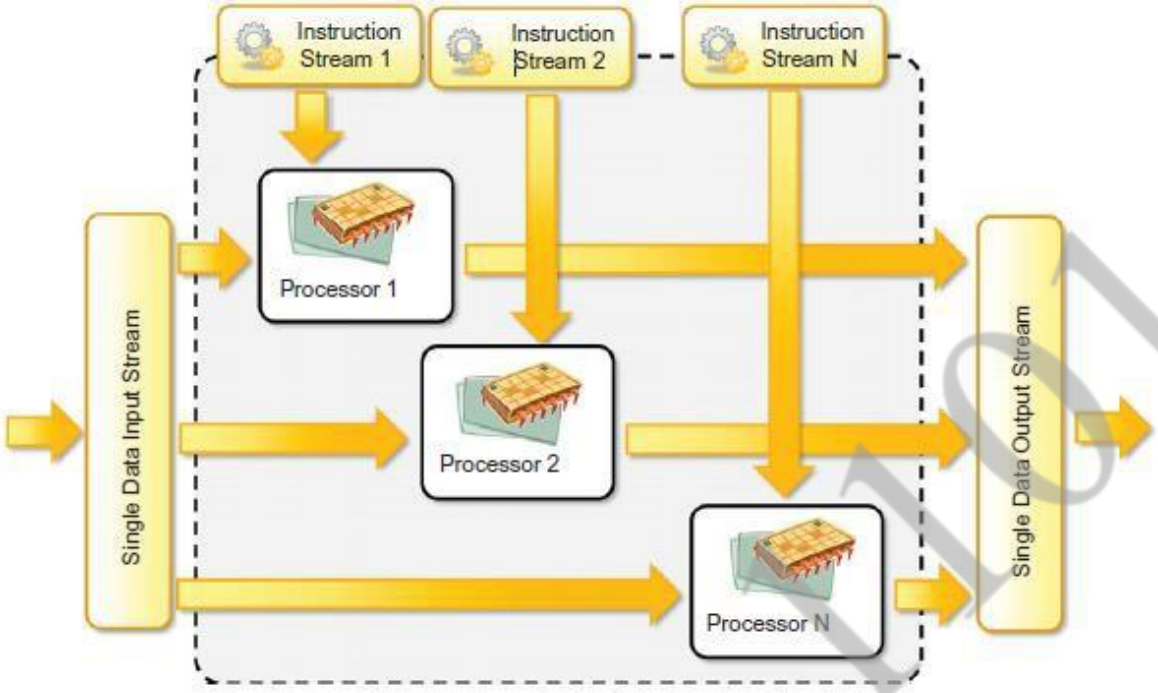


Fig: Multiple-instruction,multiple-data(MIMD) Architecture

- 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.

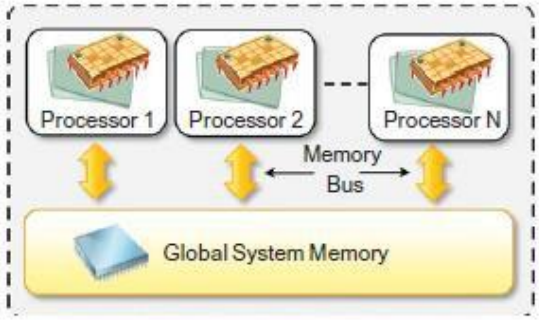


Fig: Shared MIMD

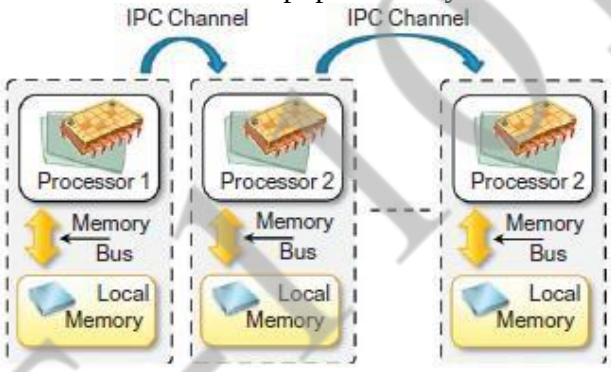


Fig: Distributed MIMD

1.4.3.3. 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.

1.4.3.4. 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

Table: Levels of parallelism

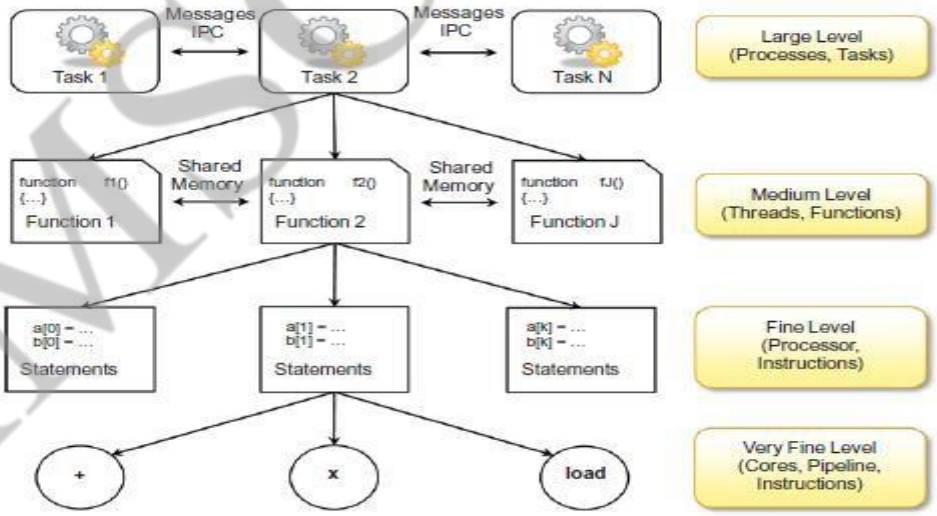


Fig: Levels of Parallelism in application

1.4.3.5. 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)).

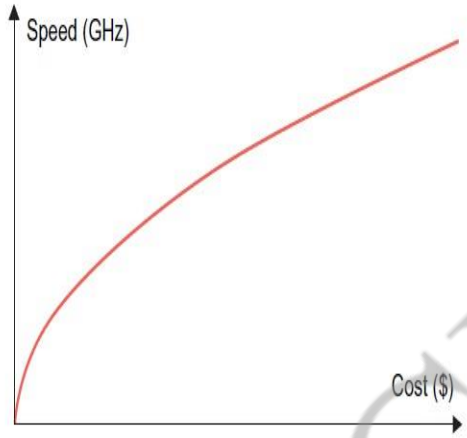


Fig: Speed vs Cost

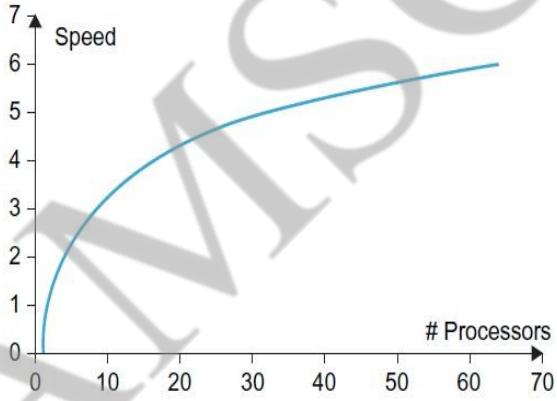


Fig: #Processors vs Speed

1.4.4. Elements of distributed computing

- In Parallel Computing, we discussed techniques and architectures that allow introduction of parallelism within a single machine or system and how parallelism operates at different levels of the computing stack.
- Here extend these concepts and explore how multiple activities can be performed by leveraging systems composed of multiple heterogeneous machines and systems.
- We discuss what is generally referred to as distributed computing and more precisely introduce the most common guidelines and patterns for implementing distributed computing systems from the perspective of the software designer.

1.4.4.1. 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.
- Here, we focus on the architectural models, that are used to harness independent computers and present them as a whole coherent system.
- 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 the encompasses several communication models.

1.4.4.2. Components of a 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.

- 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.

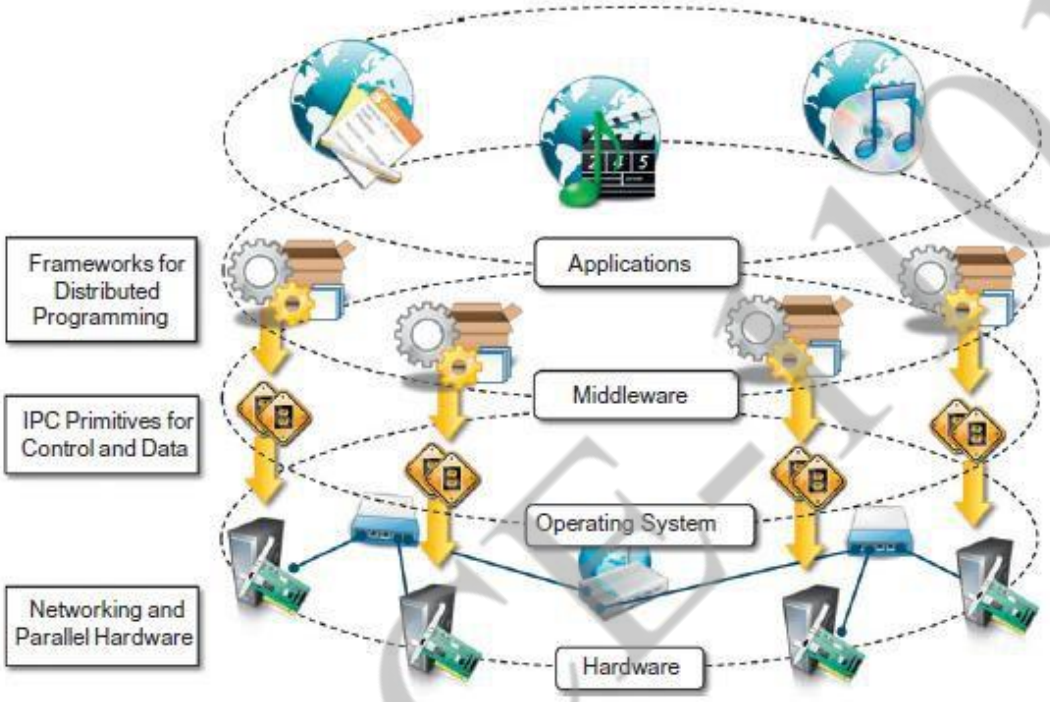


Fig: A layered view of a distributed system

1.4.4.3. Architectural styles for distributed computing

- 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.
- We organize the architectural styles into two major classes:
  - Software architectural styles
  - System architectural styles

1.4.4.3.1. 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.

Category	**Most common Architectural Styles**
Data Centered	Repository , Blackboard
Data Flow	Pipe and filter, Batch Sequential
Virtual Machine	Rule based, Interpreter
Call and return	Main program and subroutine call/top-down systems , Layered Systems
Independent Components	Communicating Processes, Event Systems

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.
- In the repository systems, the dynamics of the system is controlled by independent components, which by issuing an operation on the central repository, trigger the selection of specific processes that operate on data.

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.
- Knowledge sources represent the intelligent agents sharing the blackboard, react opportunistically to changes in the knowledge base, almost in the same way that a group of specialists brainstorm in a room in front of a blackboard.
- Blackboard models have become popular and widely used for artificial intelligent applications in which the blackboard maintains the knowledge about a domain in the form of assertion and rules, which are entered by domain experts.
- These operate through a control shell that controls the problem-solving activity of the system. Particular and successful applications of this model can be found in the domains of speech recognition and signal processing.

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.

Comparison between Batch Sequential and Pipe-and-Filter Styles

Batch Sequential	**Pipe-and-Filter**
Coarse grained	File grained

High latency	Reduced latency due to the incremental processing of input
External access to input	Localized input
No concurrency	Concurrency possible
Non interactive	Interactivity awkward but possible

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 core feature of the interpreter style is the presence of an engine that is used to interpret a pseudo-program expressed in a format acceptable for the interpreter. The interpretation of the pseudo-program constitutes the execution of the program itself. Systems modeled according to this style exhibit four main components: the interpretation engine that executes the core activity of this style, an internal memory that contains the pseudo-code to be interpreted, a representation of the current state of the engine, and a representation of the current state of the pro- gram being executed.

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.

1.4.4.3.2. System Architectural Styles:

- System architectural styles cover the physical organization of components and processes over a distributed infrastructure.
- Two fundamental reference style
  - Client / Server
    - 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.

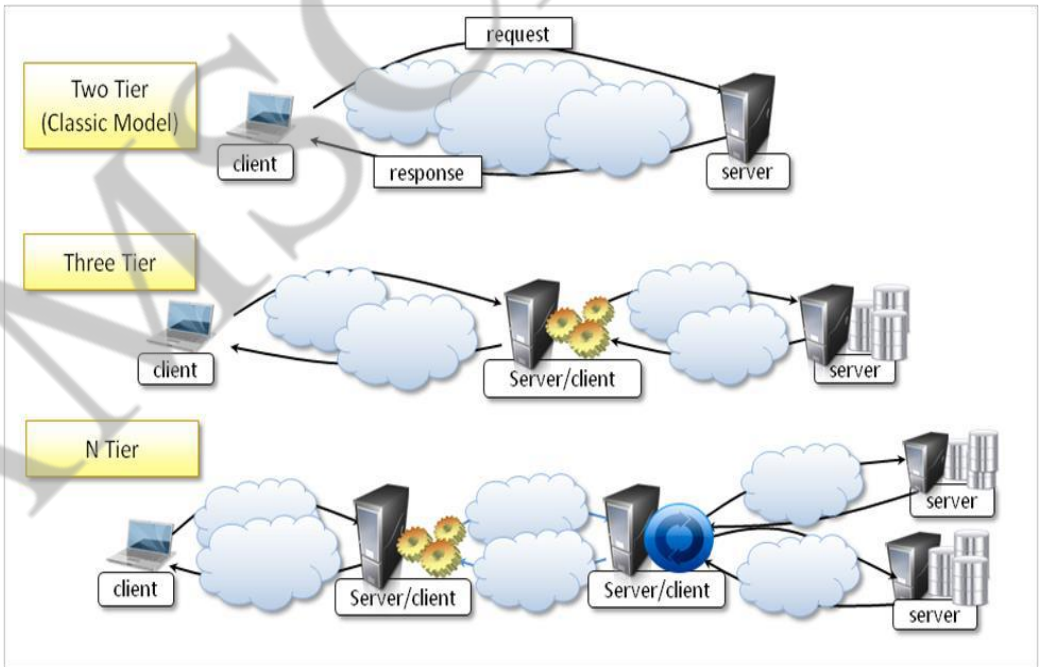


Fig: Client / Server architectural Styles

– Peer- to – Peer
    - 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.

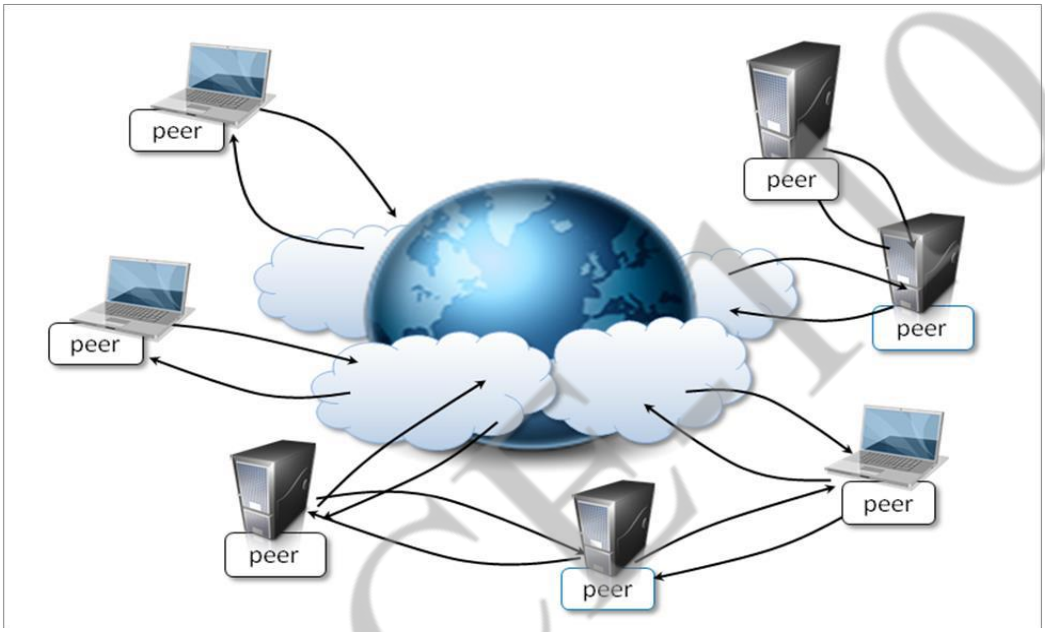


Fig: Peer-to-Peer architectural Style

1.4.4.4. 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.

1.4.4.4.1. 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 message, in this case, identifies 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, where as 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).

1.4.4.4.2. Models for message-based communication

Point-to-point message model

- This model organizes the communication among single components. Each message is sent from one component to another, and there is a direct addressing to identify the message receiver. In a point-to-point communication model it is necessary to know the location of or how to address another component in the system. There is no central infrastructure that dispatches the messages, and the communication is initiated by the message sender.

Publish-and-subscribe message model

This model introduces a different strategy, one that is based on notification among components. There are two major roles: the publisher and the subscriber. The former provides facilities for the latter 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. A 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** – In this case it is the responsibility of the publisher to notify all the subscribers— for example, with a method invocation.
• **Pull strategy** - In this case the publishers imply makes available the message for a specific event, and it is responsibility of the subscribers to check whether there are messages on the events that are registered. The publish-and-subscribe model is very suitable for implementing systems based on the one- to-many communication model and simplifies the implementation of indirect communication patterns.

It is, infact, not necessary for the publisher to know the identity of the subscribers to make the communication happen.

Request-reply message model

The request-reply message model identifies all communication models in which, for each message sent by a process, there is a reply. This model is quite popular and provides a different classification that does not focus on the number of the components involved in the communication but rather on how the dynamic of the interaction evolves. Point-to-point message models are more likely to be based on a request-reply interaction, especially in the case of direct communication. Publish- and-subscribe models are less likely to be based on request-reply since they rely on notifications.

1.4.5. Technologies for distributed computing

1.4.5.1. Remote Procedure Call (RPC)

- RPC is the fundamental abstraction enabling the execution procedures on clients' request.
- The called procedure and calling procedure may be on the same system or they may be on different systems.
- The important aspect of RPC is marshalling and unmarshalling.

- An important aspect of RPC is marshaling, which identifies the process of converting parameter and return values in to a form that is more suitable to be transported over a network through a sequence of bytes. The term unmarshaling refers to the opposite procedure. Marshaling and unmarshaling are performed by the RPC runtime infrastructure, and the client and server user code does not necessarily have to perform these tasks.
- RPC has been a dominant technology for IPC for quite a long time, and several programming languages and environments support this interaction pattern in the form of libraries and additional packages.

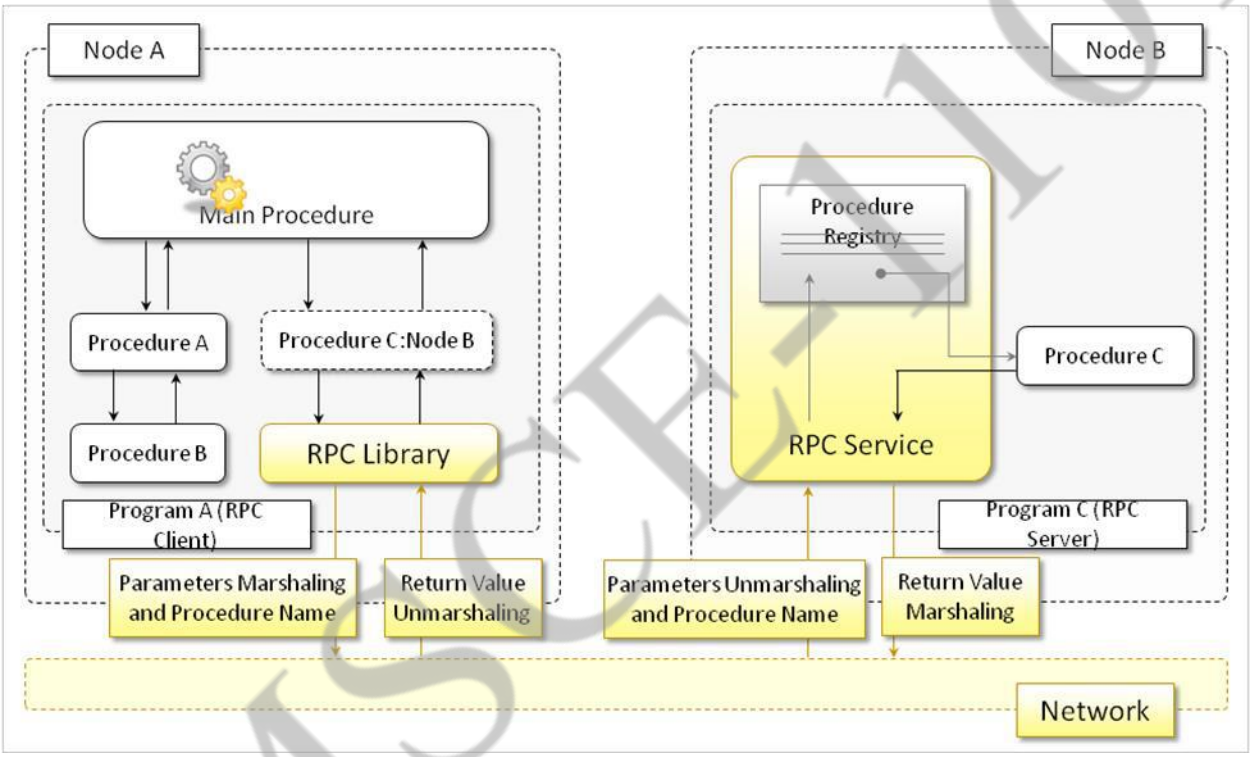


Fig: RPC Reference Model

1.4.5.2. 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.
- Distributed object frameworks leverage the basic mechanism introduced with RPC and extend it to enable the remote invocation of object methods and to keep track of references to objects made available through a network connection.

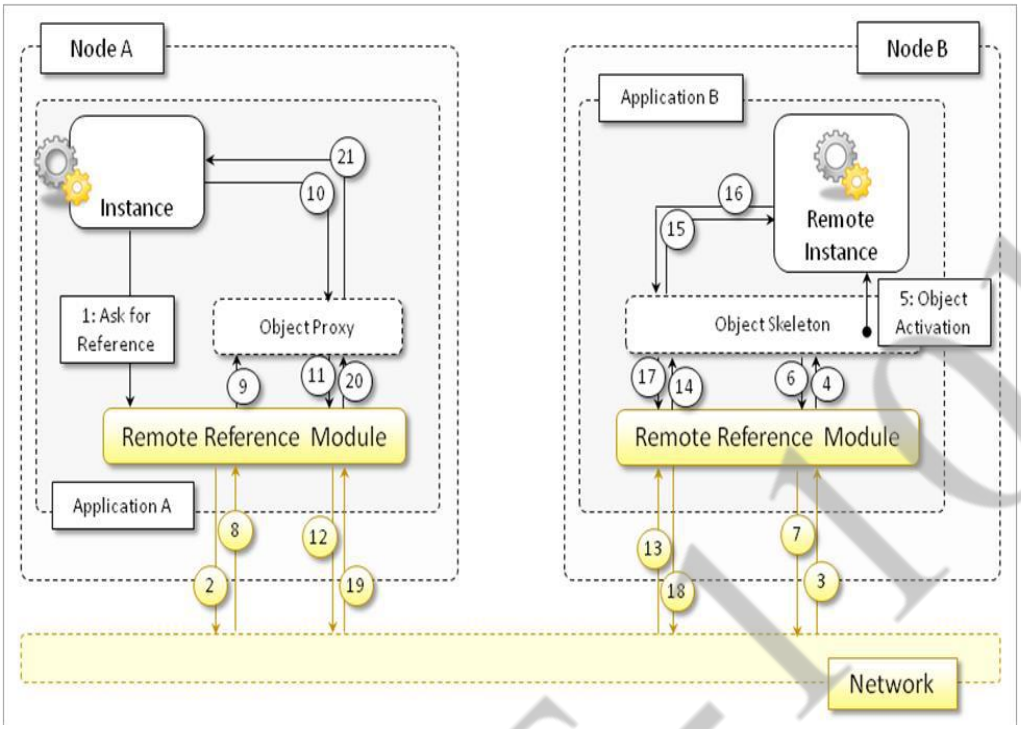


Fig: Distributed Object Framework Model

Examples of distributed Object frameworks

- Common Object Request Broker Architecture (CORBA): cross platform and cross language interoperability among distributed components.
- Distributed Component Object Model (DCOM/COM+) : Microsoft technology for distributed object programming before the introduction of .NET technology.
- Java Remote Method Invocation (RMI): technology provided by Java for enabling RPC among distributed Java objects.
- .NET Remoting: IPC among .NET applications, a uniform platform for accessing remote objects from within any application developed in any of the languages supported by .NET.

1.4.5.3. Service-oriented computing

- Service – oriented computing organizes distributed systems in terms of services, which represent the major abstraction for building systems.
- Service orientation expresses applications and software systems as an aggregation of services that are coordinated within a service oriented architecture (SOA).
- Even though there is no designed technology for the development of service-oriented software systems, web services are the de facto approach for developing SOA.
- Web services, the fundamental component enabling Cloud computing systems, leverage the Internet as the main interaction channel between users and the system.

Service-Oriented Architecture (SOA)

- SOA is an architectural style supporting service orientation. It organizes a software system into a collection of interacting services.
- SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
- SOA based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.
- There are two major roles within SOA:
  - Service Provider
  - Service Consumer

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.

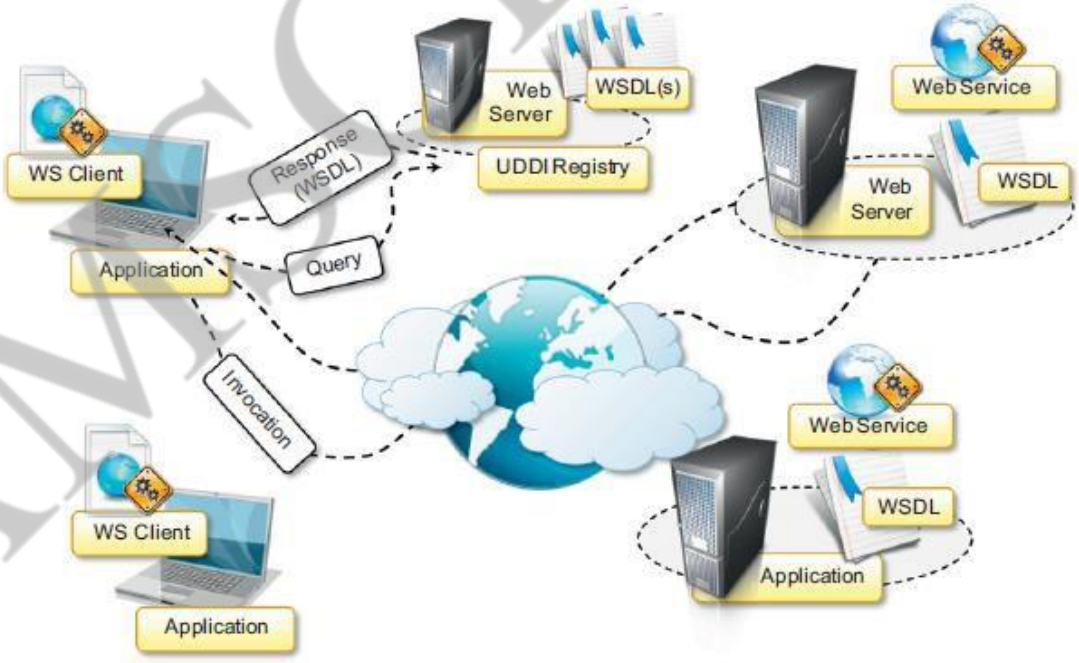


Fig: Web Services Interaction Reference Scenario

1.5. Characteristics of Cloud Computing:

Cloud computing has some interesting characteristics that bring benefits to both cloud service consumers (CSCs) and cloud service providers (CSPs).

These characteristics are:

- No up-front commitments
- On-demand access
- Nice pricing
- Simplified application acceleration and scalability
- Efficient resource allocation
- Energy efficiency
- Seamless creation and use of third-party services.

The most evident benefit from the use of cloud computing systems and technologies is the increased economical return due to the reduced maintenance costs and operational costs related to IT software and infrastructure. This is mainly because IT assets, namely software and infrastructure, are turned into utility costs, which are paid for as long as they are used, not paid for upfront.

Capital costs are costs associated with assets that need to be paid in advance to start a business activity. Before cloud computing, IT infrastructure and software generated capital costs, since they were paid up front so that business start-ups could afford a computing infrastructure, enabling the business activities of the organization. The revenue of the business is then utilized to compensate over time for these costs.

Organizations always minimize capital costs, since they are often associated with depreciable values. Minimizing capital costs, then, is fundamental. Cloud computing transforms IT infrastructure and software into utilities, thus significantly contributing to increasing a company's net gain.

Moreover, cloud computing also provides an opportunity for small organizations and start-ups: these do not need large investments to start their business, but they can comfortably grow with it.

Finally, maintenance costs are significantly reduced: by renting the infrastructure and the application services, organizations are no longer responsible for their maintenance. This task is the responsibility of the cloud service provider, who, thanks to economies of scale, can bear the maintenance costs.

Increased agility in defining and structuring software systems is another significant benefit of cloud computing. Since organizations rent IT services, they can more dynamically and flexibly compose their software systems, without being constrained by capital costs for IT assets.

There is a reduced need for capacity planning, since cloud computing allows organizations to react to unplanned surges in demand quite rapidly.

For example, organizations can add more servers to process workload spikes and dismiss them when they are no longer needed.

Ease of scalability is another advantage. By leveraging the potentially huge capacity of cloud computing, organizations can extend their IT capability more easily. Scalability can be leveraged across the entire computing stack.

End users can benefit from cloud computing by having their data and the capability of operating on it always available, from anywhere, at any time, and through multiple devices. Information and services stored in the cloud are exposed to users by Web-based interfaces that make them accessible from portable devices as well as desktops at home.

Since the processing capabilities (that is, office automation features, photo editing, information management, and so on) also reside in the cloud, end users can perform the same tasks that previously were carried out through considerable software investments. The cost for such opportunities is generally very limited, since the cloud ser- vice provider shares its costs across all the tenants that he is servicing.

Multi-tenancy allows for better utilization of the shared infrastructure that is kept operational and fully active. The concentration of IT infrastructure and services into large datacenters also provides opportunity for considerable optimization in terms of resource allocation and energy efficiency, which eventually can lead to a less impacting approach on the environment.

Finally, service orientation and on-demand access create new opportunities for composing systems and applications with a flexibility not possible before cloud computing. New service offerings can be created by aggregating together existing services and concentrating on added value.

Since it is possible to provision on demand any component of the computing stack, it is easier to turn ideas into products with limited costs and by concentrating technical efforts on what matters: the added value.

1.6. Elasticity in Cloud

- In cloud computing, elasticity is defined as “degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an automatic

manner. Such that the available resources match the current demand as closely as possible.

- It is a characteristic which differentiate cloud computing from other computing paradigms, such as grid computing etc.
- The first advantage is the elasticity of the cloud infrastructure that can grow and shrink according to the requests served. As a result, doctors and hospitals do not have to invest in large computing infrastructures designed after capacity planning, thus making more effective use of budgets.
- **Elastic Computing** – The dynamic adaptation of capacity to meet a varying workload is called Elastic Computing. i.e., ability to quickly expand or decrease computer processing, memory to meet changing demands without worrying about capacity, planning etc. Ex: by altering the use of computing resources.
- Elasticity is a characteristic in cloud computing in which computing resources can be scaled up and down easily by cloud service provider.
- Cloud Service provider gives you the provision to flexible computing power when and where required.
- The elasticity of the resources depends upon the following factors such as processing power, storage, bandwidth etc.

These services involve:

• Elasticity and scaling. By means of the dynamic provisioning service, Aneka supports dynamically upsizing and downsizing of the infrastructure available for applications.

Elasticity means the resource availability and performance can be automatically increased or decreased to meet changing customer need whereas scalabity describes the way a system is designed to meet changing demand.

There are various types of resource provisioning methods. They are :

- Overprovisioning - causes heavy resource waste
- Underprovisioning of resources results in losses by both user and provider in that paid demand by the users is not served and wasted resources still exist for those demanded areas below the provisioned capacity.
- Constant provisioning - resources with fixed capacity to a declining user demand could result in even worse resource waste.

The user may give up the service by canceling the demand, resulting in reduced revenue for the provider. Both the user and provider may be losers in resource provisioning without elasticity.

1.7. On-demand provisioning

The key features of cloud technology that make it an attractive solution in several domains are elastic scalability, on-demand resource provisioning and ubiquity.

The need to manage multiple applications in a datacenter creates the challenge of on-demand resource provisioning and allocation in response to time-varying workloads.

The on-demand access creates new opportunities for composing systems and applications with a flexibility not possible before cloud computing. New service offerings can be created by aggregating together existing services and concentrating on added value. Since it is possible to provision on demand any component of the computing stack, it is easier to turn ideas into products with limited costs and by concentrating technical efforts on what matters: the added value.

The cloud offers significant benefit to IT companies by freeing them from the low-level task of Setting up the hardware (servers) and managing the system software. Cloud computing applies a virtual platform with elastic resources put together by on-demand provisioning of hardware, software, and data sets, dynamically.

- On-demand provisioning is a system in which computing resources are made available to the users as needed. The resources may be maintained within the user enterprise or made available by a cloud service provider.
- On-demand functionality of cloud computing is ensured via internet and by delivering configurable computing resources from a shared pool of serves, storage devices, networks and software applications.
- It was developed to cater to varying demands of computing resources by users. It is not feasible to maintain optimum resources because there can be huge fluctuations in demand.
- Due to on-demand functionality, organizations need not worry about availability of resources at any given time. It is the foundation of utility based services provided to cloud computing users.
- In the absence of on-demand functionality, organizations will find it difficult to maintain huge computing resources to cater a large number of computer systems. It helped both small and large organizations to reap the benefits of cloud computing.

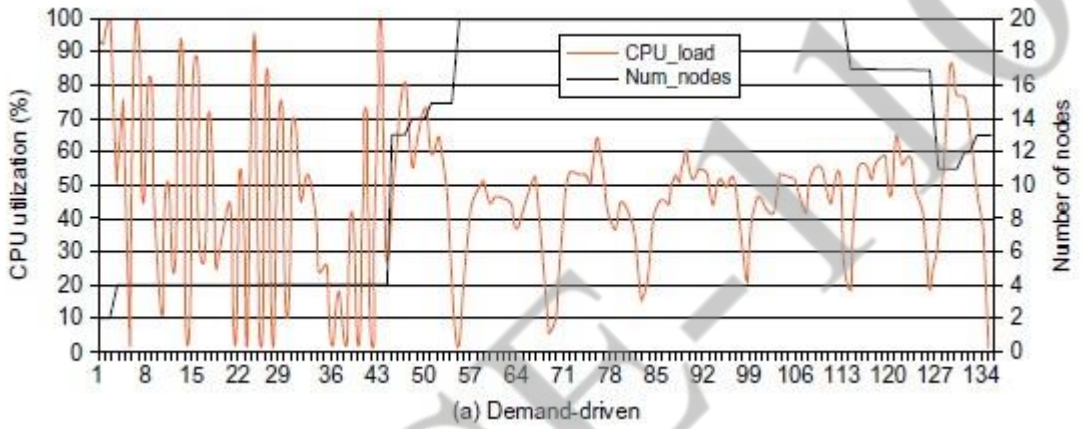
Demand-Driven Resource Provisioning

This method adds or removes computing instances based on the current utilization level of the allocated resources. The demand-driven method automatically allocates two Xeon processors for

the user application, when the user was using one Xeon processor more than 60 percent of the time for an extended period.

In general, when a resource has surpassed a threshold for a certain amount of time, the scheme increases that resource based on demand. When a resource is below a threshold for a certain amount of time, that resource could be decreased accordingly.

Amazon implements such an auto-scale feature in its EC2 platform. This method is easy to implement. The utilization rate becomes more stabilized with a maximum of 20 VMs (100 percent utilization) provided for demand-driven provisioning in below figure:



(a) Demand-driven

The scheme does not work out right if the workload changes abruptly. The x-axis in figure is the time scale in milliseconds. In the beginning, heavy fluctuations of CPU load are encountered.

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. SERVICE ORIENTED ARCHITECTURE

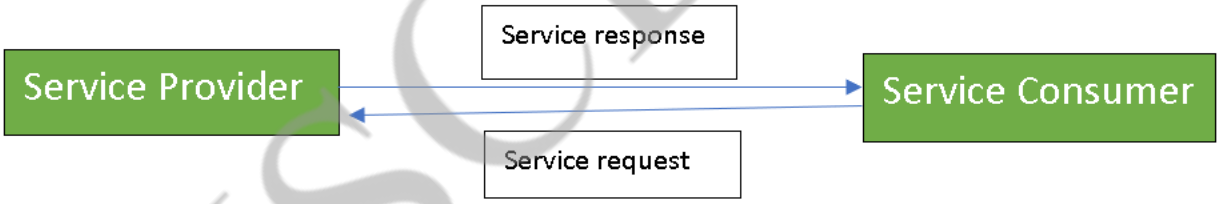
SOA is an architectural style supporting service orientation. It organizes a software system into a collection of interacting services. SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system. SOA-based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

A service-oriented architecture is essentially a collection of services. These services communicate with each other.

The communication can involve either simple data passing or it could involve two or more services coordinating some activity.

Services - A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services.

Connections - The technology of Web Services is the most likely connection technology of service-oriented architectures. The service consumer sends a service request message to a service provider. The service returns a response message to the service consumer. The request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider.



There are two major roles with in SOA: the service provider and the service consumer.

The service provider is the maintainer of the service and the organization that makes available one or more services for others to use. To advertise services, the provider can publish them in a registry, together with a service contract that specifies the nature of the service, how to use it, the requirements for the service, and the fees charged.

The service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service. Service providers and consumers can belong to different organization bodies or business domains. It is very common in SOA-based computing systems that components play the roles of both service provider and service consumer.

SOA provides a reference model for architecting several software systems, especially enterprise business applications and systems. In this context, interoperability, standards, and service contracts play a fundamental role. In particular, the following guiding principles, which characterize SOA platforms, are winning features within an enterprise context:

- Standardized service contract.
- Loose coupling.
- Abstraction

- Reusability
- Autonomy
- Lack of State
- Discoverability
- Composability

Together with these principles, other resources guide the use of SOA for enterprise application integration(EAI)

SOA can be realized through several technologies. The first implementations of SOA have leveraged distributed object programming technologies such as CORBA and DCOM. In particular, CORBA has been a suitable platform for realizing SOA systems because it fosters interoperability among different implementations and has been designed as a specification supporting the development of industrial applications. Nowadays, SOA is mostly realized through Web services technology, which provides an interoperable platform for connecting systems and applications.

2.1.1. Layered Architecture for Web Services and Grids

In grids/web services, Java, and CORBA, an entity is, respectively, a service, a Java object, and a CORBA distributed object in a variety of languages. These architectures build on the traditional seven Open Systems Interconnection (OSI) layers that provide the base networking abstractions.

On top of this we have a base software environment, which would be .NET or Apache Axis for web services, the Java Virtual Machine for Java, and a broker network for CORBA. On top of this base environment one would build a higher level environment reflecting the special features of the distributed computing environment. This starts with entity interfaces and inter-entity communication, which rebuild the top four OSI layers but at the entity and not the bit level.

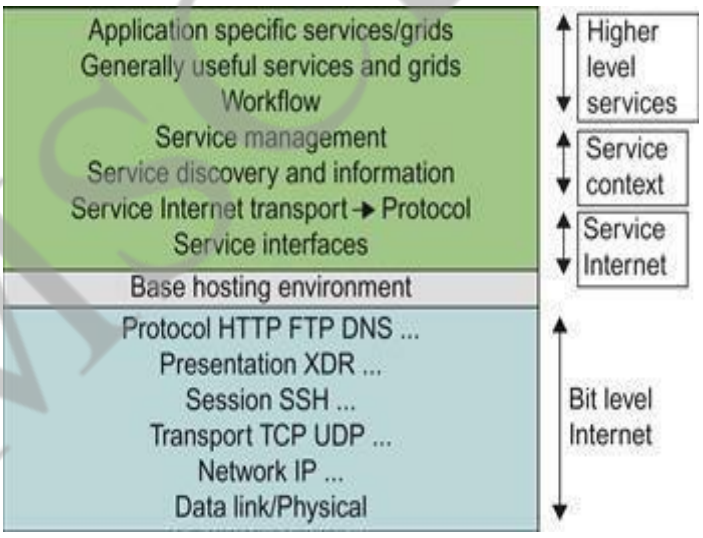


Fig:Layered *architecture for web services and the grids.*

2.1.2. The Evolution of SOA

S***ervice-oriented architecture (SOA)*** has evolved over the years. SOA applies to building grids, clouds, grids of clouds, clouds of grids, clouds of clouds (also known as interclouds), and systems of systems in general. A large number of sensors provide data-collection services, denoted in the figure as ***SS (sensor service)***. A sensor can be a ZigBee device, a Bluetooth device, a WiFi access point, a personal computer, a GPA, or a wireless phone, among other things. Raw data is collected by sensor services. All the SS devices interact with large or small computers, many forms of grids, databases, the compute cloud, the storage cloud, the filter cloud, the discovery cloud, and so on. ***Filter services*** (***fs*** in the figure) are used to

eliminate unwanted raw data, in order to respond to specific requests from the web, the grid, or web services.

A collection of filter services forms a filter cloud. SOA aims to search for, or sort out, the useful data from the massive amounts of raw data items. Processing this data will generate useful information, and subsequently, the knowledge for our daily use. In fact, wisdom or intelligence is sorted out of large knowledge bases. Finally, we make intelligent decisions based on both biological and machine wisdom.

Most distributed systems require a web interface or portal. For raw data collected by a large number of sensors to be transformed into useful information or knowledge, the data stream may go through a sequence of compute, storage, filter, and discovery clouds. Finally, the inter-service messages converge at the portal, which is accessed by all users. Two example portals, OGFCE and HUB zero using both web service (portlet) and Web 2.0 (gadget) technologies. Many distributed programming models are also built on top of these basic constructs.

2.2. REST and System of Systems

- REST as a successor of SOAP. SOAP is a web server technology which has complex definition for messages that are transmitted or received . These complex messages contains additional information like encryption and other security measures. We can say this additional information message has envelopes.
- This SOAP technique highly relies on XML and it has been used has a solid solution for making web services until REST introduced. REST introduced to make this services easier.
- When REST developed, just with HTTP protocols and by using HTTP commands we can ask for data. So No complex envelopes and no other protocols anymore. So REST protocols solved the complexity of SOAP technology.
- REST is a software architecture style for distributed systems, particularly distributed hypermedia systems, such as the World Wide Web.

- It has recently gained popularity among enterprises such as Google, Amazon, Yahoo!, and especially social networks such as Facebook and Twitter because of its simplicity, and its ease of being published and consumed by clients.

In general, **a grid system applies static resources**, while a **cloud emphasizes elastic resources.** For some researchers, the **differences between grids and clouds** are limited only in **dynamic resource allocation based on virtualization** and autonomic computing.

One **can build a grid out of multiple clouds**. This type of grid can do a beer j ob than a pure cloud, because it can explicitly support negotiated resource allocation. Thus one may end up building with a ***system of systems*:** such as a ***cloud of clouds***, ***a grid of clouds***, or a ***cloud of grids***, or ***inter-clouds*** as a basic SOA architecture.

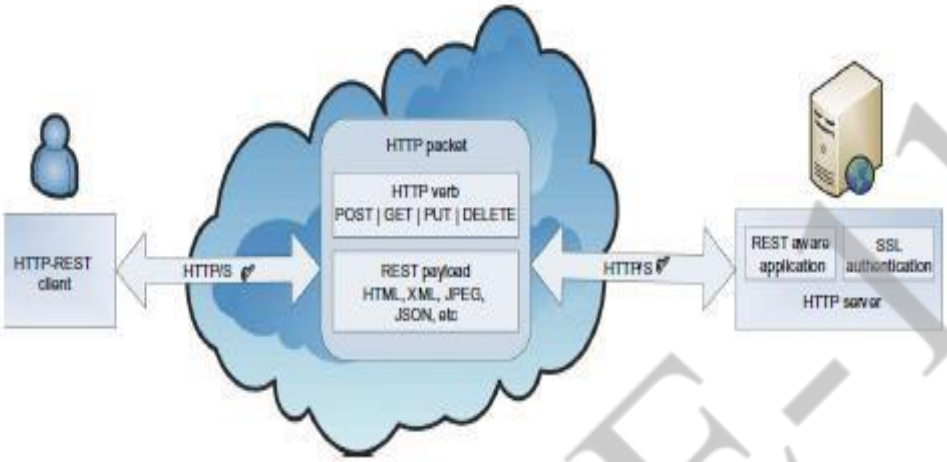


Fig: A simple REST interaction between user and server in HTTP specification.

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) which 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, providing 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, and DELETE. PUT creates a new resource, which 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 (e.g., 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, such as cache control, transmission error detection, authentication or authorization, and access control.

Stateless Interactions:

The REST interactions are "stateless" in the sense that the meaning of a message does not depend on the state of the conversation. Stateless communications improve visibility, since a monitoring system does not have to look beyond a single request data field in order to determine the full nature of the request reliability as it facilitates the task of recovering from partial failures, and increases scalability as discarding state between requests allows the server component to quickly free resources. However, stateless interactions may decrease network performance by increasing the repetitive data (per-interaction overhead).

Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be embedded in response messages to point to valid future states of the interaction.

• Such lightweight infrastructure, where services can be built with minimal development tools, is inexpensive and easy to adopt.
• The effort required to build a client to interact with a RESTful service is small , as developers can begin testing such services from an ordinary web browser, without having to develop custom client-side software.
• From an operational point of view, a stateless RESTful web service is scalable to serve a very large number of clients, as a result of REST support for caching, clustering, and load balancing.
• RESTful web services can be considered an alternative to SOAP stack or "big web services,"because of their simplicity, lightweight nature, and integration with HTTP.
• With the help of URIs and hyperlinks, REST has shown that it is possible to discover web resources without an approach based on registration to a centralized repository.
• Recently, the web Application Description Language (WADL) has been proposed as an XML vocabulary to describe RESTful web services, enabling them to be discovered and accessed immediately by potential clients.
• However, there are not a variety of toolkits for developing RESTful applications.
• Also,restrictions on GET length, which does not allow encoding of more than 4 KB of data in theresource URI, can create problems because the server would reject such malformed URIs, or may even be subject to crashes.
• REST is not a standard. It is a design and architectural style for largescale distributed systems.

Table 5.1 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

Data Elements - Data elements are Key aspect of REST is the state of data elements.
REST identifies six data elements.Its components communicate by transferring representations of the current state of data elements.
Components - The various software that interacts with one another are called components.
Components communicate with each other via connectors.

Connectors - Connectors represents the activities involved in accessing resources and transferring representations.

2.3. 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, XML, and WSDL .
- Third, they pro- vide an intuitive and simple way to connect heterogeneous software systems, enabling the quick composition of services in a distributed environment.
- Finally, they provide the features required by enterprise business applications to be used in an industrial environment.

- The web has becomes a medium for connecting remote clients with applications for years, and more recently, integrating applications across the Internet has gained in popularity.
- The term "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. Once a web service is deployed, other applications and other web services can discover and invoke the deployed service.

The concept behind a Web service is very simple. Using as a basis the object-oriented abstrac- tion, a Web service exposes a set of operations that can be invoked by leveraging Internet-based protocols.
The semantics for invoking Web service methods is expressed through interoperable stan- dards such as XML and WSDL
Web services are made accessible by being hosted in a Web server; therefore, HTTP is the most popular transport protocol used for interacting with Web services.

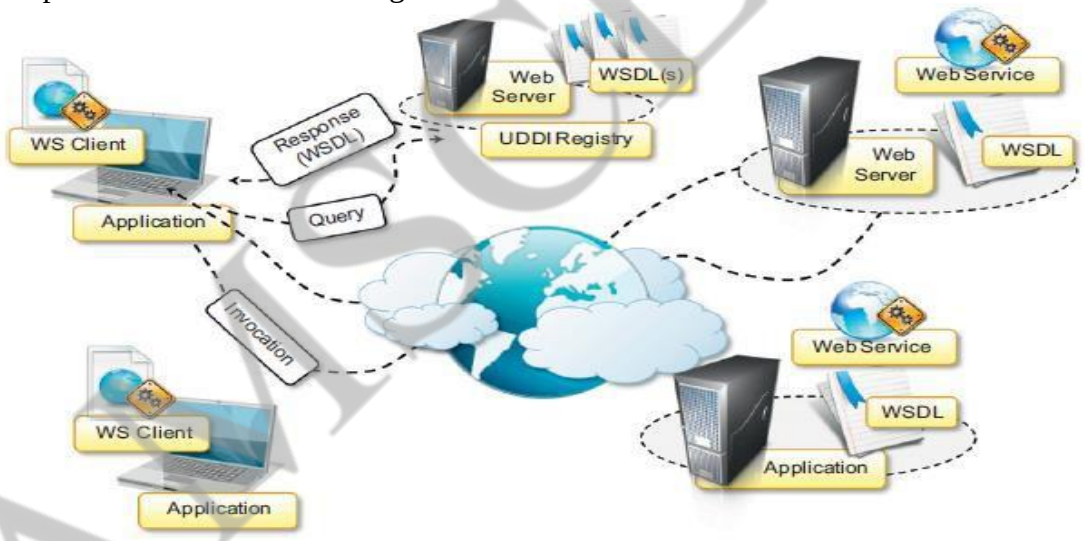


Fig: Web Services Interaction Reference Scenario

System architects develop a Web service with their technology of choice and deploy it in compatible Web or application servers. The service description document, expressed by means of Web Service Definition Language (WSDL), can be either uploaded to a global registry or attached as a metadata to the service itself.
Service consumers can look up and discover services in global catalogs using Universal Description Discovery and Integration (UDDI) or, most likely, directly retrieve the service metadata by interrogating the Web service first. The Web service description document allows service consumers to automatically generate clients for the given service and embed them in their existing application.
Moreover, being interoperable, Web services constitute a better solution for SOA with respect to several distributed object frame-works, such as .NET Remoting, JavaRMI ,and DCOM/COM1, which limit their applicability to a single platform or environment.

Web services encompass several technologies that put together and facilitate the integration of heterogeneous applications and enable service-oriented computing.
The backbone of all these technologies is XML, which is also one of the causes of Web services' popularity and ease of use. XML-based languages are used to manage the low-level interaction for Web service method calls (SOAP), for providing metadata about the services (WSDL), for discovery services (UDDI), and other core operations. In practice, the core components that enable Web services are SOAP and WSDL.

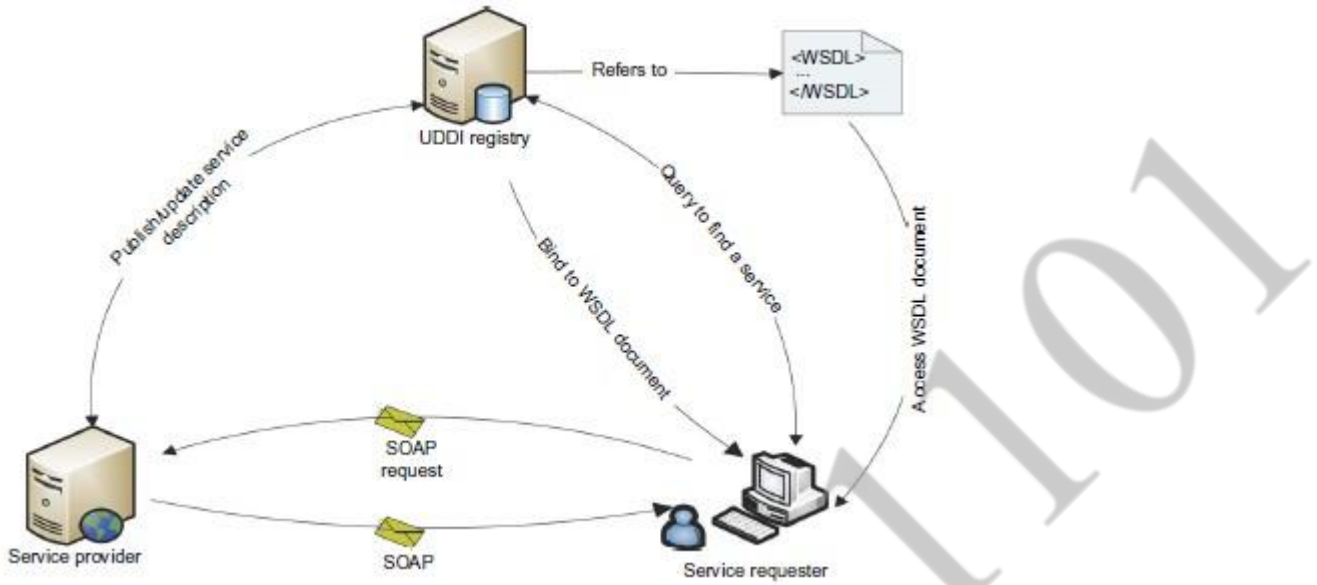


Fig: A simple web service interaction among provider, user, and the UDDI registry.

The technologies that make up the core of today's web services are as follows:
• 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. By having such a standard message format, heterogeneous middleware systems can achieve interoperability.

A SOAP message consists of a root element called envelope, which contains a header: a container that can be extended by intermediaries with additional application-level elements such as routing information, authentication, transaction management, message parsing instructions, and Quality of Service (QoS) configurations, as well as a 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, by searching for names, identifiers, categories, or the specification implemented by the web service.

In addition, there are many evolving standards related to Web Services Interoperability (WS-I) that also can be applied to and bring value to grid environments, standards, and proposed standards.

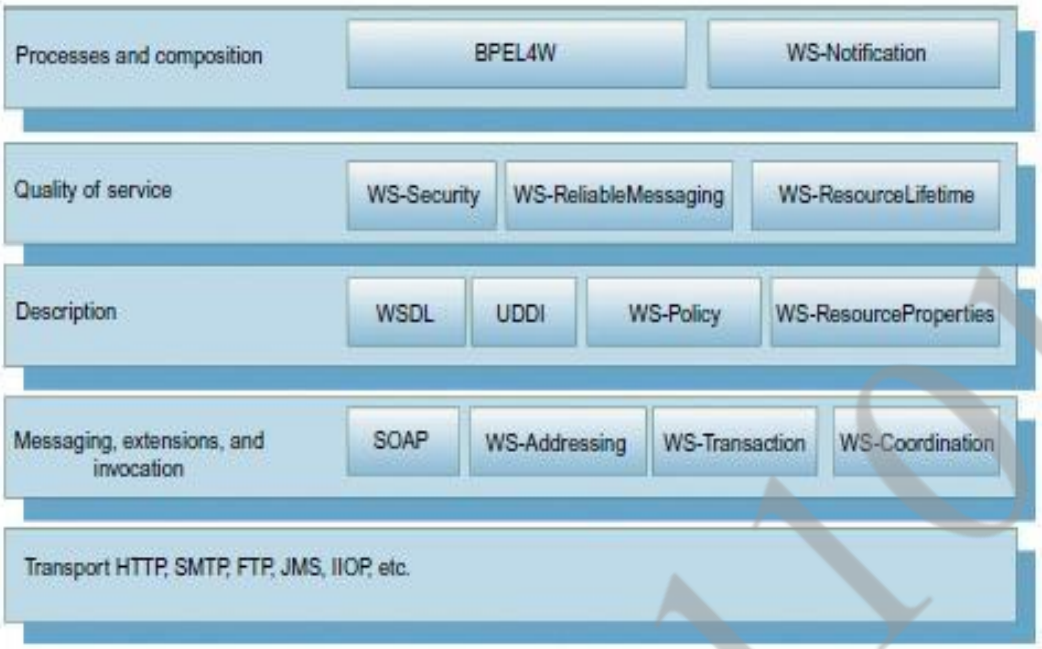


Fig: WS-I protocol stack and its related specifications.

Business Process Execution Language for Web Services (BPEL4WS) is 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

2.4. PUBLISH-SUBSCRIBE MODEL

• An important concept here is "publish-subscribe" which describes a particular model for linking source and destination for a message bus.
• Here the producer of the message (publisher) labels the message in some fashion; often this is done by associating one or more topic names from a (controlled) vocabulary.
• Then the receivers of the message (subscriber) will specify the topics for which they wish to receive associated messages.
• Alternatively, one can use content-based delivery systems where the content is queried in some format such as SQL.
• The use of topic or content-based message selection is termed message filtering.
• Publish-subscribe is a design pattern that enables asynchronous interaction among distributed applications .
• Publish-subscribe messaging middleware allows straightforward implementation of notification or event-based programming models.
• In a publish-subscribe interaction, event subscribers register to particular event types and receive notifications from the event publishers when they generate such events.
• There is a dynamic, many-to-many relationship between event publishers and event subscribers, as there can be any number of publishers/subscribers for any type of event which can vary at any time.
• Publish-subscription adds dynamicity to static the nature of databases.
• publish-subscribe pattern was first implemented in centralized client/server systems, current research focuses mainly on distributed versions.

• Publish-subscribe systems are classified as either topic-based or content-based.
• In topic-based systems, publishers generate events with respect to a topic or subject.
• Subscribers then specify their interest in a particular topic, and receive all events published on that topic. Defining events in terms of topic names only is inflexible and requires subscribers to filter events belonging to general topics.
• Content-based systems solve this problem by introducing a subscription scheme based on the contents of events.
• Content-based systems are preferable as they give users the ability to express their interest by specifying predicates over the values of a number of well-defined attributes.
• The matching of publications (events) to subscriptions (interest) is done based on the content. Distributed solutions are mainly focused on topic-based publish-subscribe systems.

Publish-and-subscribe message model

This model introduces a different strategy, one that is based on notification among components.
There are two major roles: the publisher and the subscriber. The former provides facilities for the latter 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. A 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** – In this case it is the responsibility of the publisher to notify all the subscribers— for example, with a method invocation.
• **Pull strategy** - In this case the publishers imply makes available the message for a specific event, and it is responsibility of the subscribers to check whether there are messages on the events that are registered. The publish-and-subscribe model is very suitable for implementing systems based on the one- to-many communication model and simplifies the implementation of indirect communication patterns.

It is, infact, not necessary for the publisher to know the identity of the subscribers to make the communication happen.

Databases and Publish-Subscribe

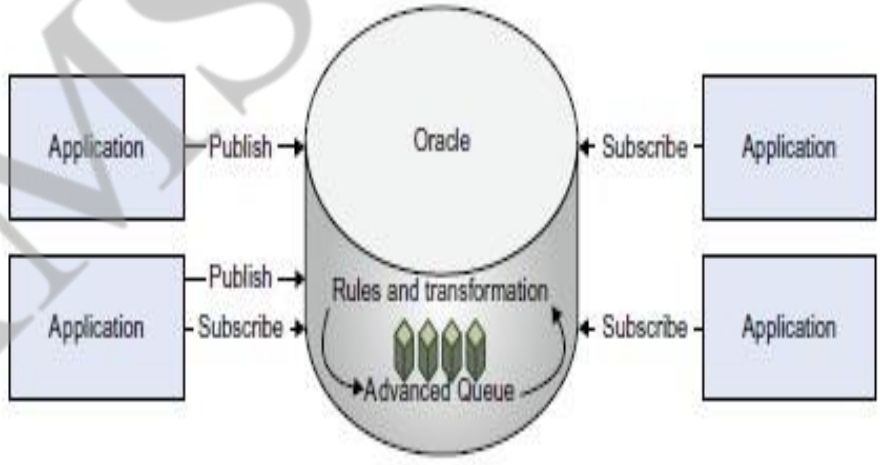


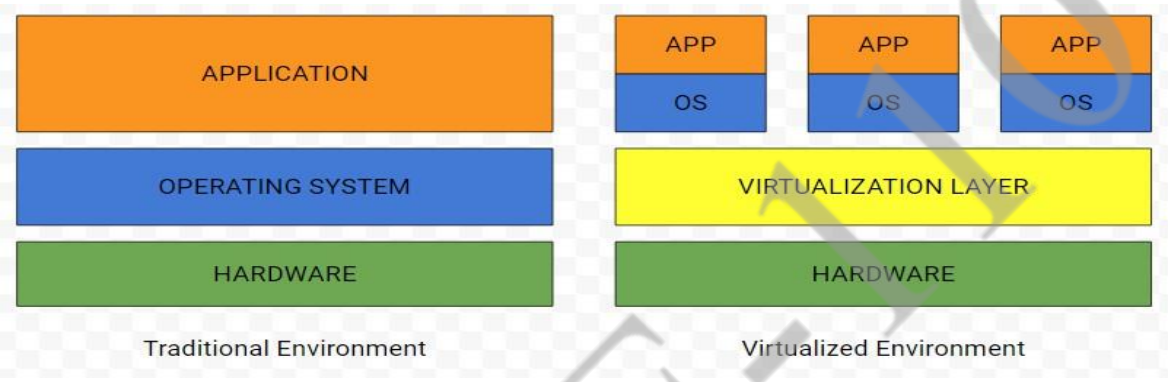
Fig: Oracle publish-subscribe model.

Database systems provide many features that a messaging-based architecture can exploit, such as reliable storage, transactions, and triggers. On the other hand, integrated publish-subscribe capabilities in the database account for information-sharing systems that are simpler to deploy and maintain. However, since publish-subscribe and database technology have evolved independently, designing and implementing

database-publish-subscribe-aware systems requires bringing together concepts and functionality from two separate worlds.

2.5. BASICS OF VIRTUALIZATION

Virtualization is the creation of virtual servers, infrastructures, devices and computing resources. Virtualization changes the hardware-software relations and is one of the foundational elements of cloud computing technology that helps utilize the capabilities of cloud computing to the full. Virtualization techniques allow companies to turn virtual their networks, storage, servers, data, desktops and applications.



A great example of how virtualization works in your daily life is the separation of your hard drive into different parts. While you may have only one hard drive, your system sees it as two, three or more different and separate segments. Similarly, this technology has been used for a long time. It started as the ability to run multiple operating systems on one hardware set and now it is a vital part of testing and cloud-based computing.

A technology called the Virtual Machine Monitor — also called virtual manager — encapsulates the very basics of virtualization in cloud computing. It is used to separate the physical hardware from its emulated parts. This often includes the CPU's memory, I/O and network traffic. A secondary operating system that is usually interacting with the hardware is now a software emulation of that hardware, and often the guest operating system has no idea it's on the virtualized hardware. Despite the fact that the performance of the virtual system is not equal to the functioning of the "true hardware" operating system, the technology still works because most secondary OSs and applications don't need the full use of the underlying hardware. This allows for greater flexibility, control and isolation by removing the dependency on a given hardware platform.

The layer of software that enables this abstraction is called "hypervisor". Hypervisor as "a software layer that can monitor and virtualize the resources of a host machine conferring to the user requirements." The most common hypervisor is referred to as Type 1. By talking to the hardware directly, it virtualizes the hardware platform that makes it available to be used by virtual machines. There's also a Type 2 hypervisor, which requires an operating system. Most often, you can find it being used in software testing and laboratory research.

Virtualization vs. Cloud Computing

Unlike virtualization, cloud computing refers to the service that results from that change. It describes the delivery of shared computing resources, SaaS and on-demand services through the Internet. Most of the confusion occurs because virtualization and cloud computing work together to provide different types of services, as is the case with private clouds.

The cloud often includes virtualization products as a part of their service package. The difference is that a true cloud provides the self-service feature, elasticity, automated management, scalability and pay-as-you-go service that is not inherent to the technology.

2.6. TYPES OF VIRTUALIZATION

There are six virtualization techniques in cloud computing. They are:

- **Network Virtualization**
- **Storage Virtualization**
- **Server Virtualization**
- **Data Virtualization**
- **Desktop Virtualization**
- **Application Virtualization**

Network Virtualization

Network virtualization in cloud computing is a method of combining the available resources in a network by splitting up the available bandwidth into different channels, each being separate and distinguished. They can be either assigned to a particular server or device or stay unassigned completely — all in real time. The idea is that the technology disguises the true complexity of the network by separating it into parts that are easy to manage, much like your segmented hard drive makes it easier for you to manage files.

Storage Virtualization

Using this technique gives the user an ability to pool the hardware storage space from several interconnected storage devices into a simulated single storage device that is managed from one single command console. This storage technique is often used in storage area networks. Storage manipulation in the cloud is mostly used for backup, archiving, and recovering of data by hiding the real and physical complex storage architecture. Administrators can implement it with software applications or by employing hardware and software hybrid appliances.

Server Virtualization

This technique is the masking of server resources. It simulates physical servers by changing their identity, numbers, processors and operating systems. This spares the user from continuously managing complex server resources. It also makes a lot of resources available for sharing and utilizing, while maintaining the capacity to expand them when needed.

Data Virtualization

This kind of cloud computing virtualization technique is abstracting the technical details usually used in data management, such as location, performance or format, in favor of broader access and more resiliency that are directly related to business needs.

Desktop Virtualization

As compared to other types of virtualization in cloud computing, this model enables you to emulate a workstation load, rather than a server. This allows the user to access the desktop remotely. Since the workstation is essentially running in a data center server, access to it can be both more secure and portable.

Application Virtualization

Software virtualization in cloud computing abstracts the application layer, separating it from the operating system. This way the application can run in an encapsulated form without being dependent upon the operating system underneath. In addition to providing a level of isolation, an application created for one OS can run on a completely different operating system.

2.7. 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. This virtualization technology has been revitalized as the demand for distributed and cloud computing increased sharply in recent years.

The idea is to separate the hardware from the software to yield better system efficiency. Virtualization techniques can be applied to enhance the use of compute engines, networks, and storage. In this chapter we will discuss VMs and their applications for building distributed systems. With sufficient storage, any computer platform can be installed in another host computer, even if they use processors with different instruction sets and run with distinct operating systems on the same hardware.

Levels of Virtualization Implementation

A traditional computer runs with a host operating system specially tailored for its hardware architecture. 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 . 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.

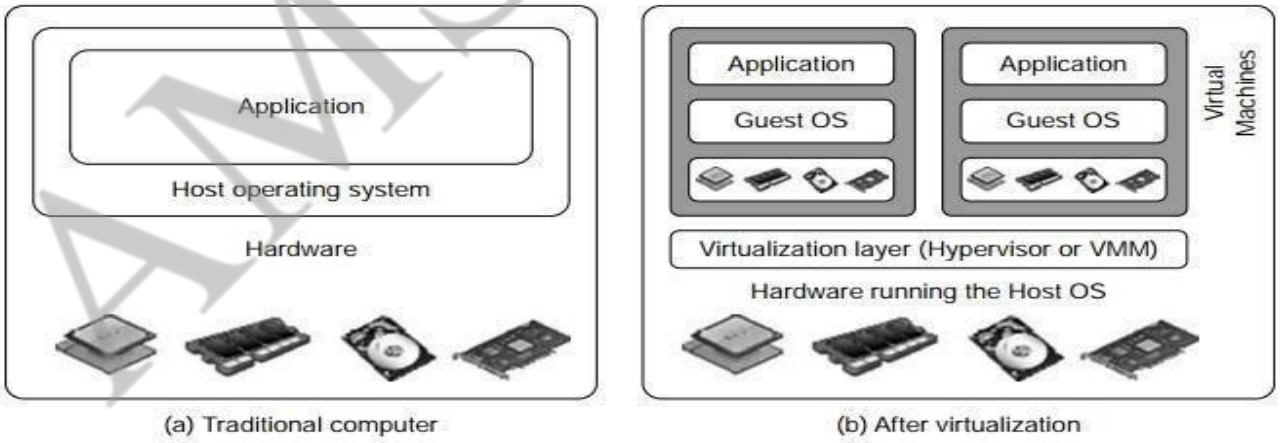


FIGURE 3.1

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

The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs, exclusively. This can be implemented at various operational levels, as we will discuss shortly. The virtualization software creates the abstraction of VMs by interposing a virtualization layer at various levels of a computer system. Common virtualization layers include the instruction set architecture (ISA) level, hardware level, operating system level, library support level, and application level (see Figure 3.2).

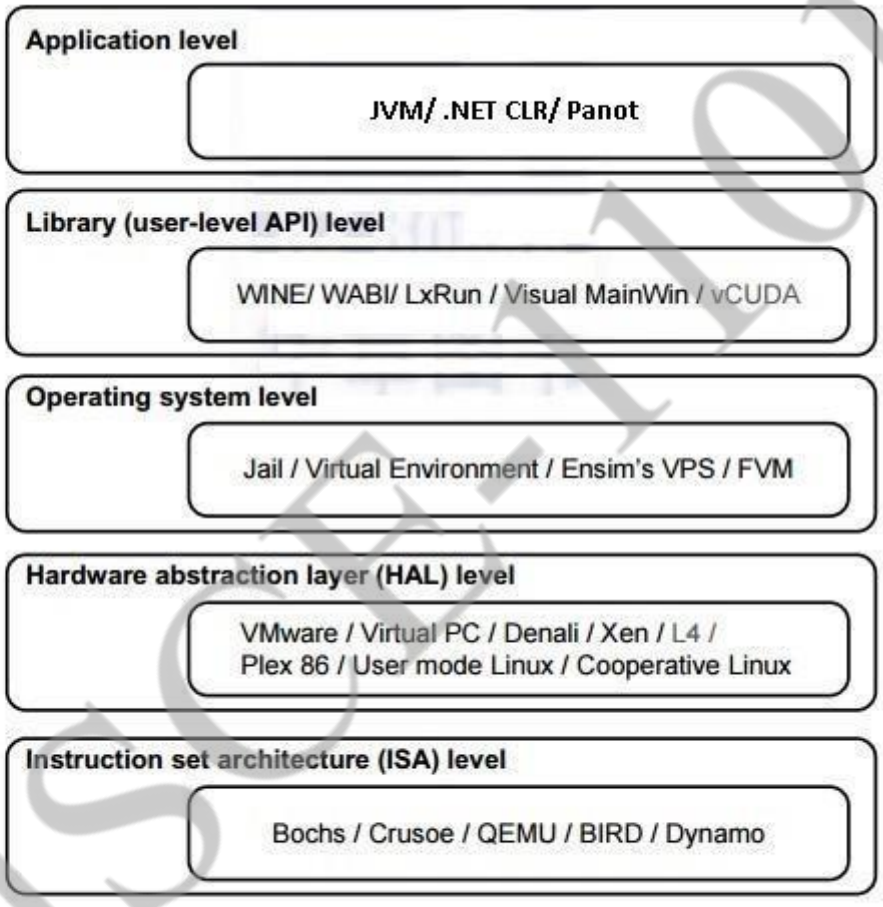


FIGURE 3.2

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. One source instruction may require tens or hundreds of native target instructions to perform its function. Obviously, this process is relatively slow. For better performance, dynamic binary translation is desired. This approach translates basic blocks of dynamic source instructions to target instructions. The basic blocks can also be extended to program traces or super blocks to increase translation efficiency. Instruction set emulation requires binary translation and optimization. A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.

- **Hardware Abstraction Level**

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

- **Operating System Level**

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

- **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. In this scenario, the virtualization layer sits as an application program on top of the operating system, and the layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition. Any program written in the HLL and compiled for this VM will be able to run on it. The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM.

Other forms of application-level virtualization are known as application isolation, application sandboxing, or application streaming. The process involves wrapping the application in a layer that is isolated from the host OS and other applications. The result is an application that is much easier to distribute and remove from user workstations. An example is the LANDesk application virtualization platform, which deploys software applications as self-contained, executable files in an isolated environment without requiring installation, system modifications, or elevated security privileges.

Relative Merits of Different Approaches

Table 3.1 compares the relative merits of implementing virtualization at various levels. The column headings correspond to four technical merits. "Higher Performance" and "Application Flexibility" are self-explanatory. "Implementation Complexity" implies the cost to implement that particular vir-tualization level. "Application Isolation" refers to the effort required to isolate resources committed to different VMs. Each row corresponds to a particular level of virtualization.

The number of X's in the table cells reflects the advantage points of each implementation level. Five X's implies the best case and one X implies the worst case. Overall, hardware and OS support will yield the highest performance. However, the hardware and application levels are also the most expensive to implement. User isolation is the most difficult to achieve. ISA implementation offers the best application flexibility.

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

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

2.8. VIRTUALIZATION STRUCTURES – TOOLS AND MECHANISMS

In general, there are three typical classes of VM architecture. Figure 3.1 showed the architectures of a machine before and after virtualization. Before virtualization, the operating system manages the hardware.

After virtualization, a virtualization layer is inserted between the hardware and the operating system. 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.

Hypervisor and Xen Architecture

The hypervisor supports hardware-level virtualization (see Figure 3.1(b)) on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physical hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor.

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

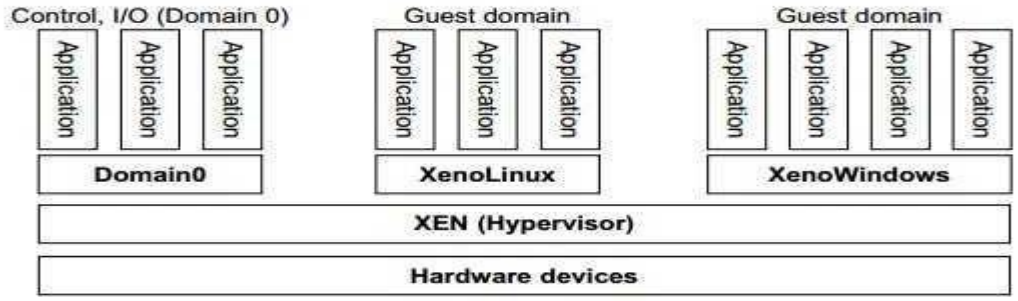


FIGURE 3.5

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

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

Xen is an open source hypervisor program developed by Cambridge University.

Xen is a micro- kernel hypervisor, which separates the policy from the mechanism. The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0, as shown in Figure 3.5.

Xen does not include any device drivers natively. It just provides a mechanism by which a guest OS can have direct access to the physical devices.

As a result, the size of the Xen hypervisor is kept rather small. Xen provides a virtual environment located between the hardware and the OS. A number of vendors are in the process of developing commercial Xen hypervisors, among them are Citrix XenServer and Oracle VM.

The core components of a Xen system are the hypervisor, kernel, and applications. The organization of the three components is important. Like other virtualization systems, many guest OSes can run on top of the hypervisor.

However, not all guest OSes are created equal, and one in particular controls the others. The guest OS, which has control ability, is called Domain 0, and the others are called Domain U.

Domain 0 is a privileged guest OS of Xen. It is first loaded when Xen boots without any file system drivers being available. Domain 0 is designed to access hardware directly and manage devices.

Therefore, one of the responsibilities of Domain 0 is to allocate and map hardware resources for the guest domains (the Domain U domains).

For example, Xen is based on Linux and its security level is C2. Its management VM is named Domain 0, which has the privilege to manage other VMs implemented on the same host. If Domain 0 is compromised, the hacker can control the entire system. So, in the VM system, security

policies are needed to improve the security of Domain 0. Domain 0, behaving as a VMM, allows users to create, copy, save, read, modify, share, migrate, and roll back VMs as easily as manipulating a file, which flexibly provides tremendous benefits for users. Unfortunately, it also brings a series of security problems during the software life cycle and data lifetime.

Traditionally, a machine's lifetime can be envisioned as a straight line where the current state of the machine is a point that progresses monotonically as the software executes. During this time, configuration changes are made, software is installed, and patches are applied. In such an environment, the VM state is akin to a tree: At any point, execution can go into N different branches where multiple instances of a VM can exist at any point in this tree at any given time. VMs are allowed to roll back to previous states in their execution (e.g., to fix configuration errors) or rerun from the same point many times (e.g., as a means of distributing dynamic content or circulating a "live" system image).

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, nonvirtualizable instructions. The guest OSes and their applications consist of noncritical and critical instructions. In a host-based system, both a host OS and a guest OS are used. A virtualization software layer is built between the host OS and guest OS. These two classes of VM architecture are introduced next.

- ***Full Virtualization***

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

- ***Binary Translation of Guest OS Requests Using a VMM***

VMware and many other software companies implemented this approach. As shown in Figure 3.6, VMware puts the VMM at Ring 0 and the guest OS at Ring 1. The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions. The method used in this emulation is called binary translation. Therefore, full

virtualization combines binary translation and direct execution. The guest OS is completely decoupled from the underlying hardware. Consequently, the guest OS is unaware that it is being virtualized.

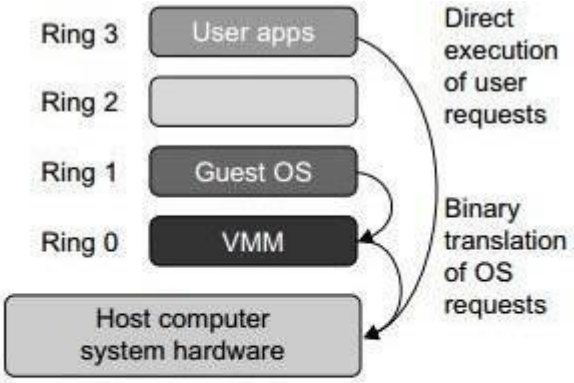


FIGURE 3.6

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

The performance of full virtualization may not be ideal, because it involves binary translation, which is rather time-consuming. In particular, the full virtualization of I/O-intensive applications is a really a big challenge. Binary translation employs a code cache to store translated hot instructions to improve performance, but it increases the cost of memory usage. At the time of this writing, the performance of full virtualization on the x86 architecture is typically 80 percent to 97 percent that of the host machine.

- ***Host-Based Virtualization***

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

Second the host-based approach appeals to many host machine configurations. Compared to the hypervisor/VMM architecture, the performance of the host-based architecture may also be low. When an application requests hardware access, it involves four layers of mapping which downgrades performance significantly. When the ISA of a guest OS is different from the ISA of the

underlying hardware, binary translation must be adopted. Although the host-based architecture has flexibility, the performance is too low to be useful in practice.

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.

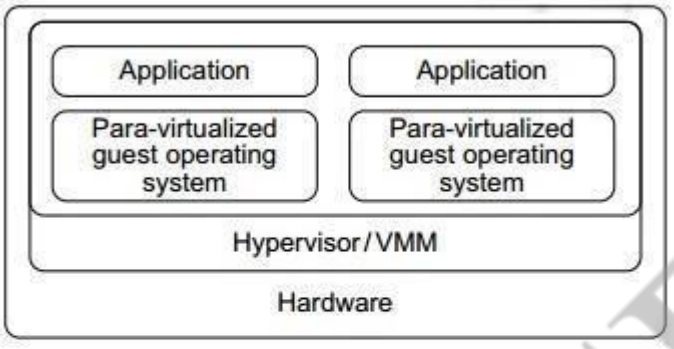


FIGURE 3.7

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

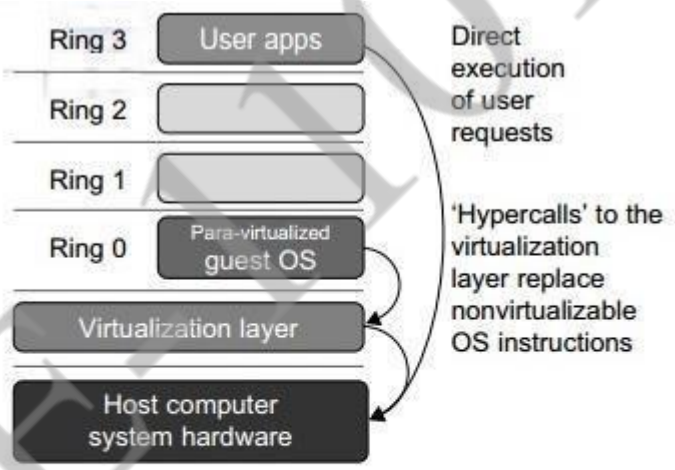


FIGURE 3.8

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

(*Courtesy of VMWare [71]*)

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

Figure 3.7 illustrates the concept of a para-virtualized VM architecture. The guest operating systems are para-virtualized.

They are assisted by an intelligent compiler to replace the nonvirtualizable OS instructions by hypercalls as illustrated in Figure 3.8.

The traditional x86 processor offers four instruction execution rings: Rings 0, 1, 2, and 3. The lower the ring number, the higher the privilege of instruction being executed. The OS is responsible for managing the hardware and the privileged instructions to execute at Ring 0, while user-level applications run at Ring 3. The best example of para-virtualization is the KVM to be described below.

Para-Virtualization Architecture

When the x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS. According to the x86 ring definition, the virtualization layer should also be installed at Ring 0. Different instructions at Ring 0 may cause some problems.

In Figure 3.8, we show that para-virtualization replaces nonvirtualizable instructions with hypercalls that communicate directly with the hypervisor or VMM. However, when the guest OS kernel is modified for virtualization, it can no longer run on the hardware directly.

Although para-virtualization reduces the overhead, it has incurred other problems. First, its compatibility and portability may be in doubt, because it must support the unmodified OS as well. Second, the cost of maintaining para-virtualized OSes is high, because they may require deep OS kernel modifications. Finally, the performance advantage of para-virtualization varies greatly due to workload variations. Compared with full virtualization, para-virtualization is relatively easy and more practical. The main problem in full virtualization is its low performance in binary translation. To speed up binary translation is difficult. Therefore, many virtualization products employ the para-virtualization architecture. The popular Xen, KVM, and VMware ESX are good examples.

- ✓ ***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.

2.9. VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

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

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.

At the time of this writing, many hardware virtualization products were available. The

VMware many hardware virtualization products were available. The VMware Workstation is a VM software suite for x86 and x86-64 computers. This software suite allows users to set up multiple x86 and x86-64 virtual computers and to use one or more of these VMs simultaneously with the host operating system. The VMware Work station assumes the host-based virtualization. Xen is a hypervisor for use in IA-32, x86-64, Itanium, and PowerPC 970 hosts. Actually, Xen modifies Linux as the lowest and most privileged layer, or a hypervisor.

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

A CPU architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode. When the privileged instructions including control- and behavior-sensitive instructions of a VM are executed, they are trapped in the VMM. 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. However, not all CPU architectures are virtualizable. RISC CPU architectures can be naturally virtualized because all control- and behavior-sensitive instructions are privileged instructions. On the contrary, x86 CPU architectures are not primarily designed to support virtualization. This is because about 10 sensitive instructions, such as SGDT and SMSW, are not privileged instructions. When these instructions execute in virtualization, they cannot be trapped in the VMM.

On a native UNIX-like system, a system call triggers the 80h interrupt and passes control to the OS kernel. The interrupt handler in the kernel is then invoked to process the system call. On a para-virtualization system such as Xen, a system call in the guest OS first triggers the 80h interrupt

normally. Almost at the same time, the 82h interrupt in the hypervisor is triggered. Incidentally, control is passed on to the hypervisor as well. When the hypervisor completes its task for the guest OS system call, it passes control back to the guest OS kernel. Certainly, the guest OS kernel may also invoke the hypercall while it's running. Although paravirtualization of a CPU lets unmodified applications run in the VM, it causes a small performance penalty.

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.

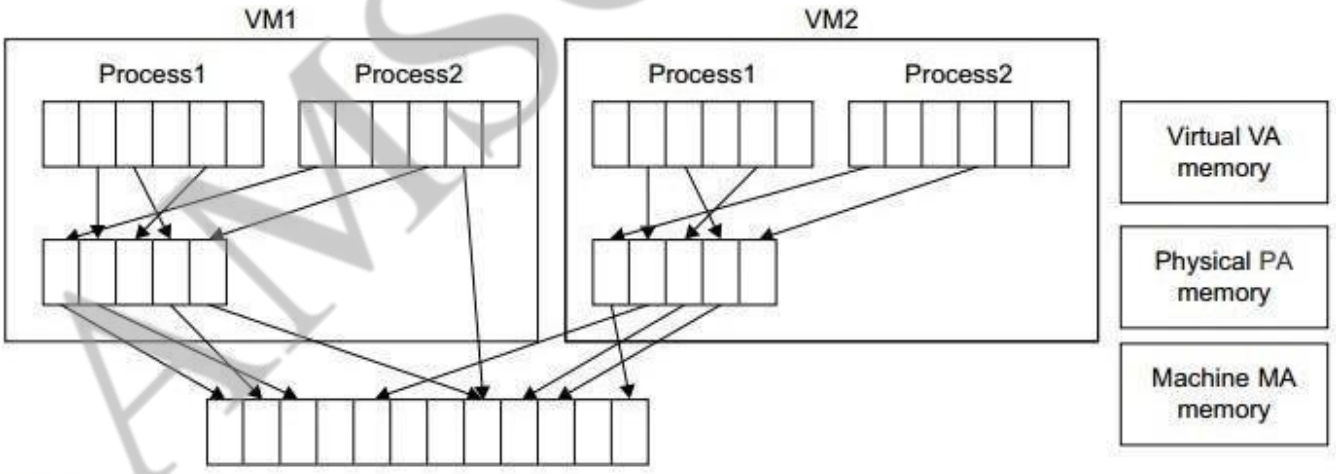


FIGURE 3.12

Two-level memory mapping procedure.

Figure 3.12 shows the two-level memory mapping procedure.corresponding to it, the VMM page table is called the shadow page table. Nested page tables add another layer of indirection to virtual memory. The MMU already handles virtual-to-physical translations as defined by the OS.

Then the physical memory addresses are translated to machine

addresses using another set of page tables defined by the hypervisor. Since modern operating systems maintain a set of page tables for every process, the shadow page tables will get flooded. Consequently, the performance overhead and cost of memory will be very high.

VMware uses shadow page tables to perform virtual-memory-to-machine-memory address translation. Processors use TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access. When the guest OS changes the virtual memory to a physical memory mapping, the VMM updates the shadow page tables to enable a direct lookup. The AMD Barcelona processor has featured hardware-assisted memory virtualization since 2007. It provides hardware assistance to the two-stage address translation in a virtual execution environment by using a technology called nested paging.

I/O Virtualization

I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware. At the time of this writing, there are three ways to implement I/O virtualization: full device emulation, para-virtualization, and direct I/O. Full device emulation is the first approach for I/O virtualization. Generally, this approach emulates well-known, real-world devices. All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated in software. This software is located in the VMM and acts as a virtual device. The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices. The full device "emulation approach is shown in Figure 3.14.

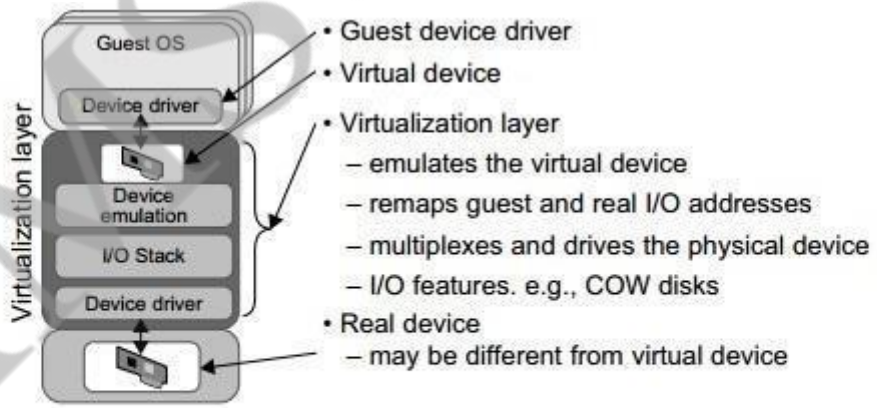


FIGURE 3.14

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

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.

Direct I/O virtualization lets the VM access devices directly. It can achieve close-to-native performance without high CPU costs. However, current direct I/O virtualization implementations focus on networking for mainframes. There are a lot of challenges for commodity hardware devices. For example, when a physical device is reclaimed (required by workload migration) for later reassignment, it may have been set to an arbitrary state (e.g., DMA to some arbitrary memory locations) that can function incorrectly or even crash the whole system. Since software-based I/O virtualization requires a very high overhead of device emulation, hardware-assisted I/O virtualization is critical. Intel VT-d supports the remapping of I/O DMA transfers and device- generated interrupts. The architecture of VT-d provides the flexibility to support multiple usage models that may run unmodified, special-purpose, or "virtualization-aware" guest OSes.

2.10. VIRTUALIZATION SUPPORT AND DISASTER RECOVERY

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

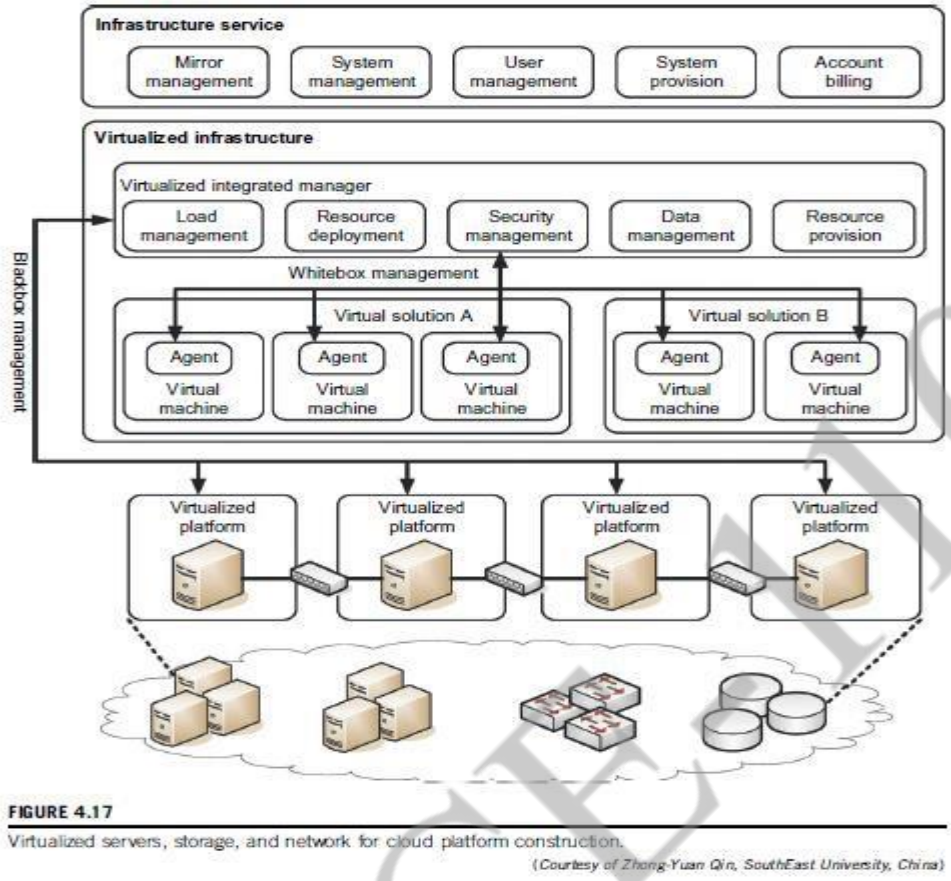


FIGURE 4.17

Virtualized servers, storage, and network for cloud platform construction.

(Courtesy of Zhong-Yuan Qin, SouthEast University, China)

Fig: Infrastructure needed to virtualize the servers in a data center for implementing specific cloud applications.

2.10.1. 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.

The below table lists some of the system virtualization software in wide use at the time of this writing.

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

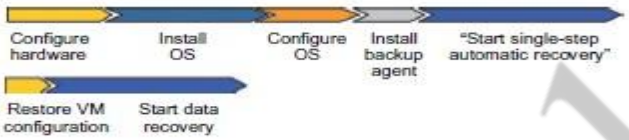


FIGURE 4.18

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

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.

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.

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.

Each platform carries out a virtual solution to a user job. All cloud services are managed in the boxes at the top.

2.10.2 Virtualization Support in Public Clouds

Mainly, three public clouds in the context of virtualization support: AWS, Microsoft Azure, and GAE.

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.

Virtualization leads to High Avalilability, 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.

2.10.3 Storage Virtualization for Green Data Centers

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.

Green data centers and benefits of storage virtualization are considered to further strengthen the synergy of green computing.

2.10.4. 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.

2.10.5. 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.

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.

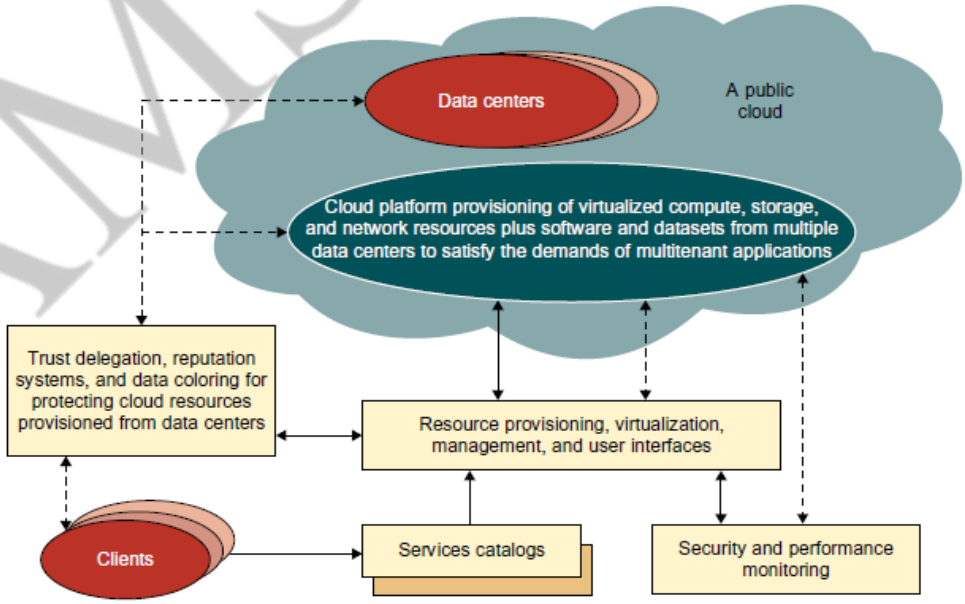
- Clouds are enabled by the progress in hardware, software and networking technologies

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.

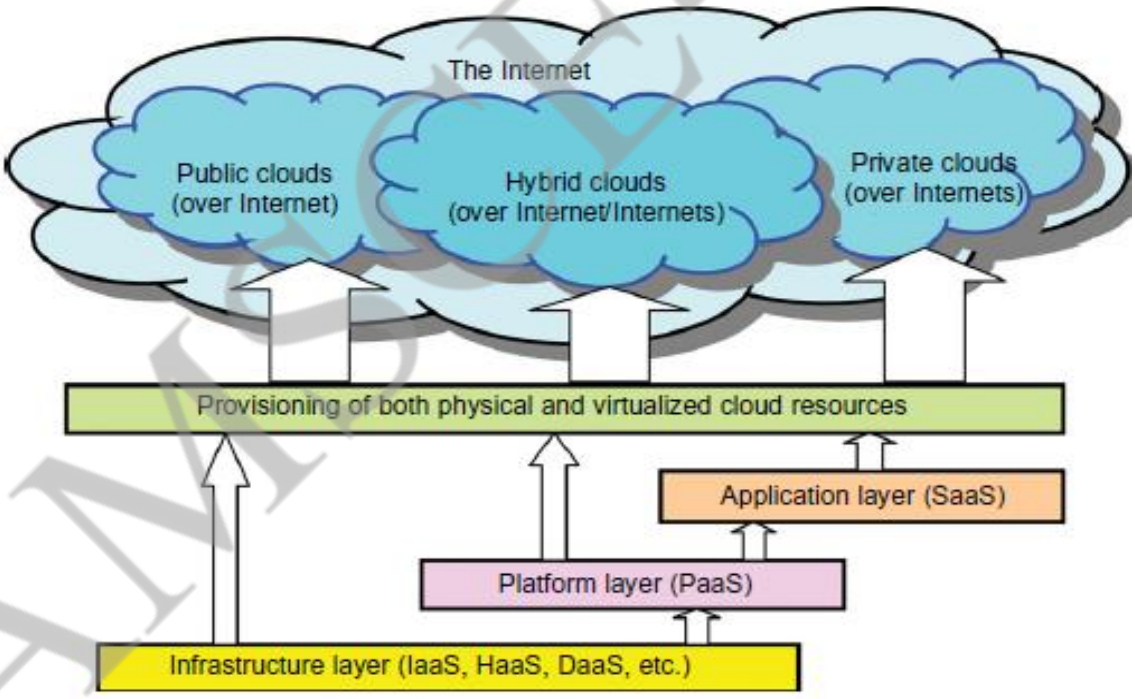


- The cloud computing resources are built into the data centers.
- Data centers are typically owned and operated by a third-party provider.

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

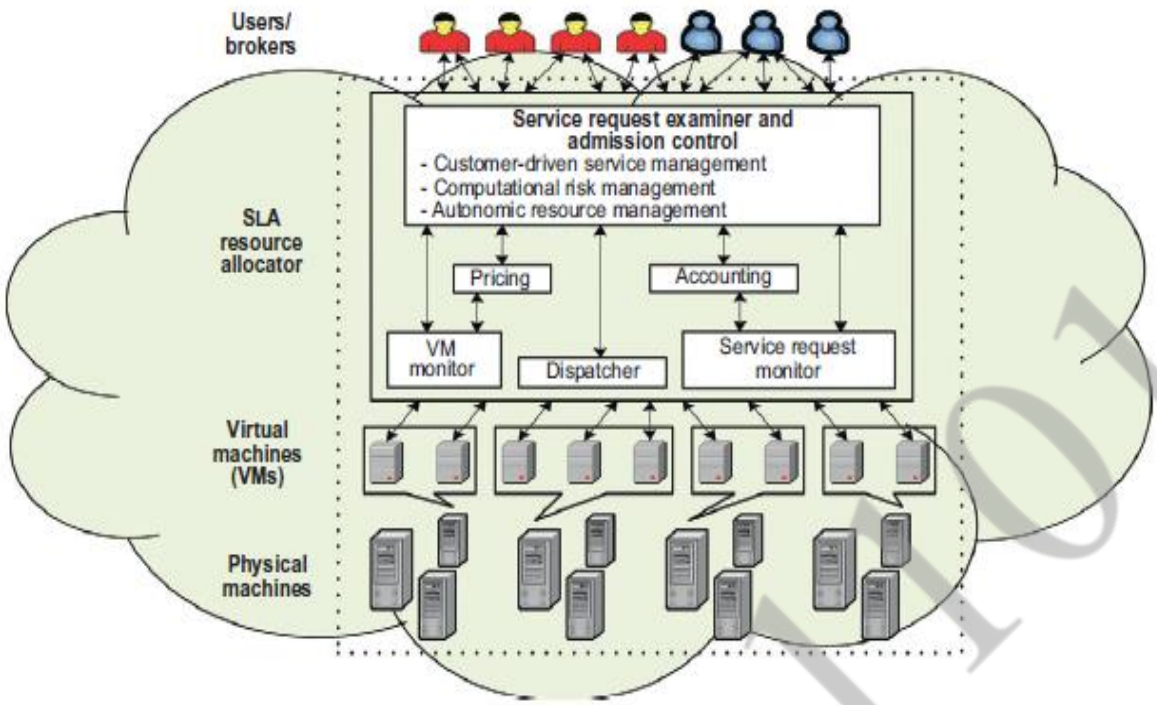
Application Layer

- 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.

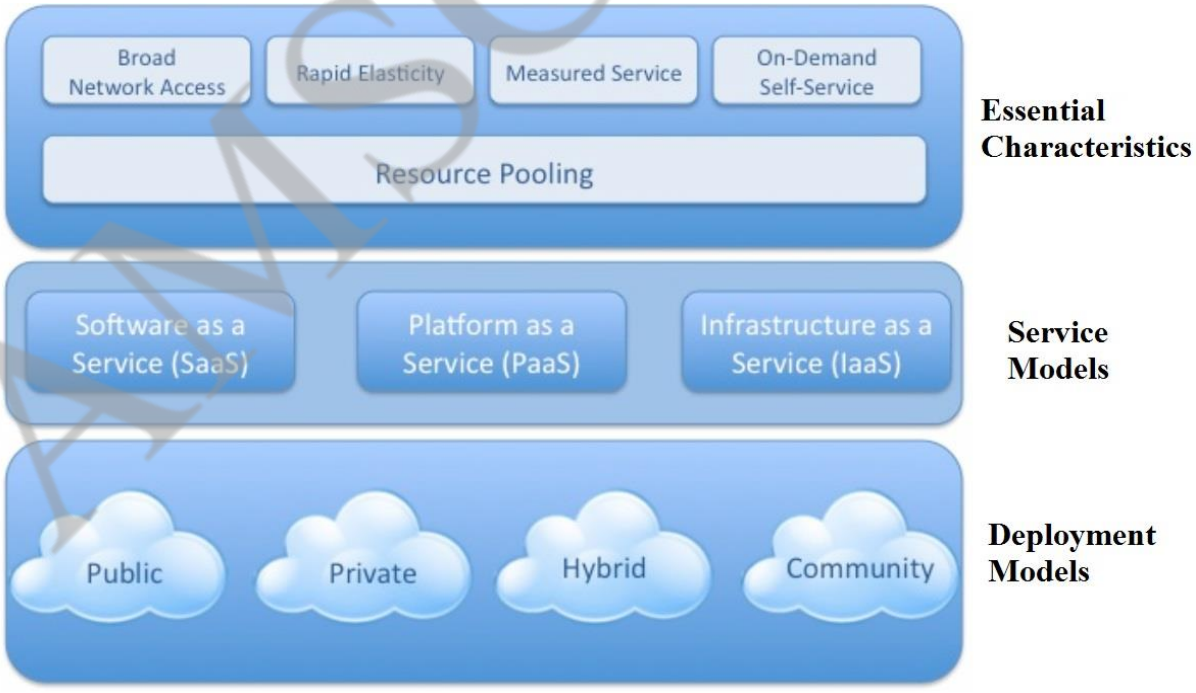
NIST Cloud Definition:

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

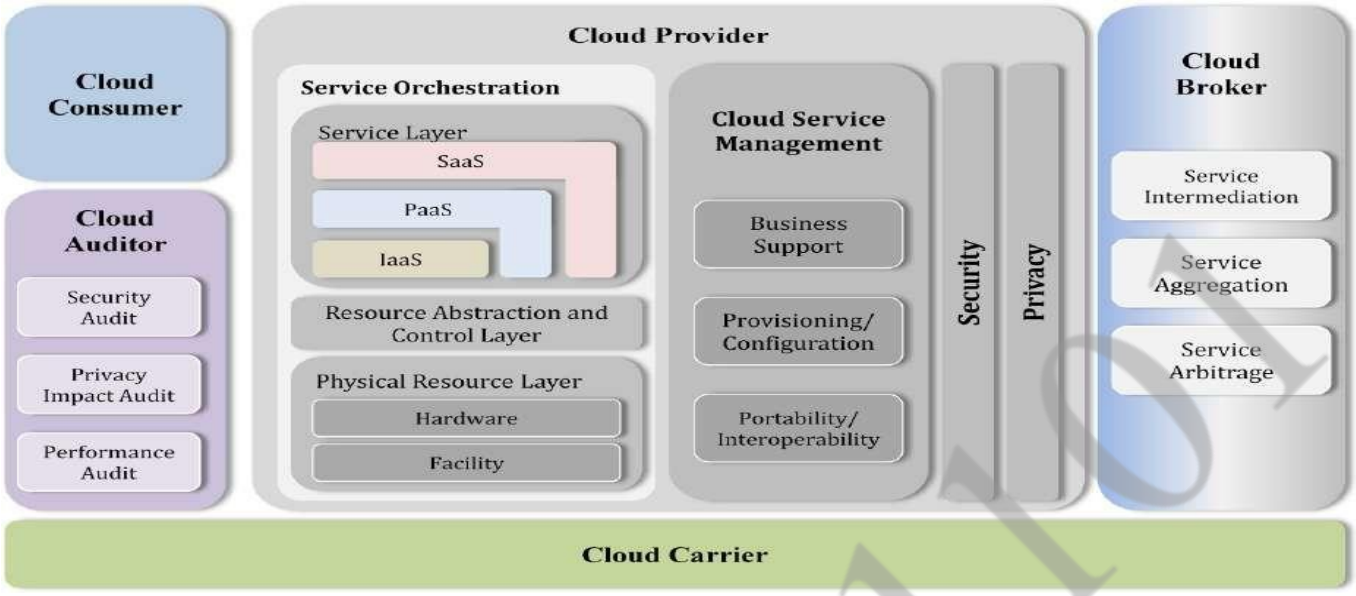
- **Measured service.**
    - Cloud systems automatically control and optimize resource usage by leveraging a metering capability at some level of abstraction appropriate to the type of service.

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)

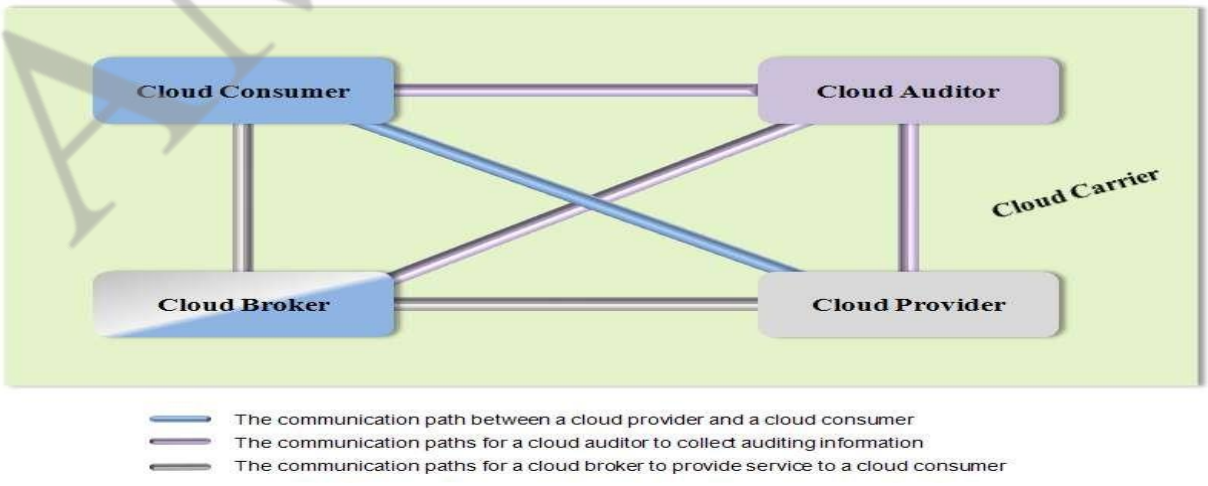
The goal is to accelerate the federal government's adoption of secure and effective cloud computing to reduce costs and improve services.

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

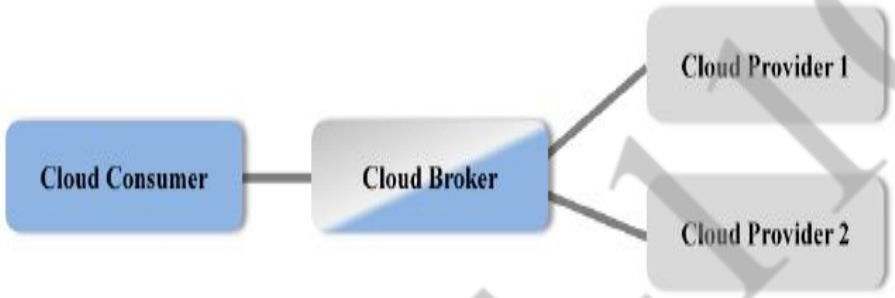


Example Usage Scenario 1:

- 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.

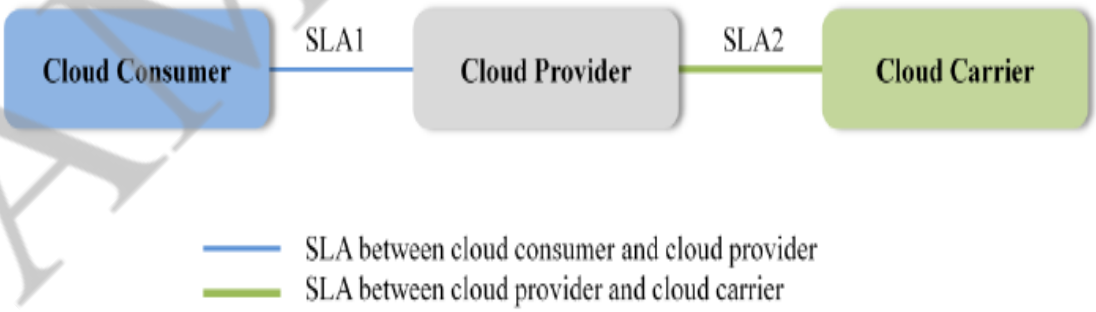


Example Usage Scenario 2

- 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

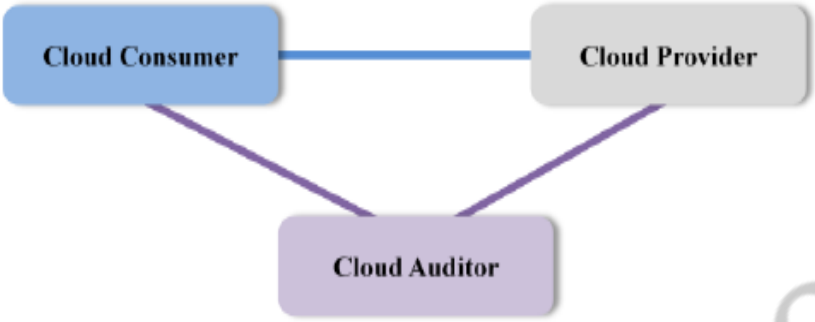
- A cloud provider arranges service level agreements (SLAs) with a cloud carrier.
- Request dedicated and encrypted connections to ensure the cloud services.



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.

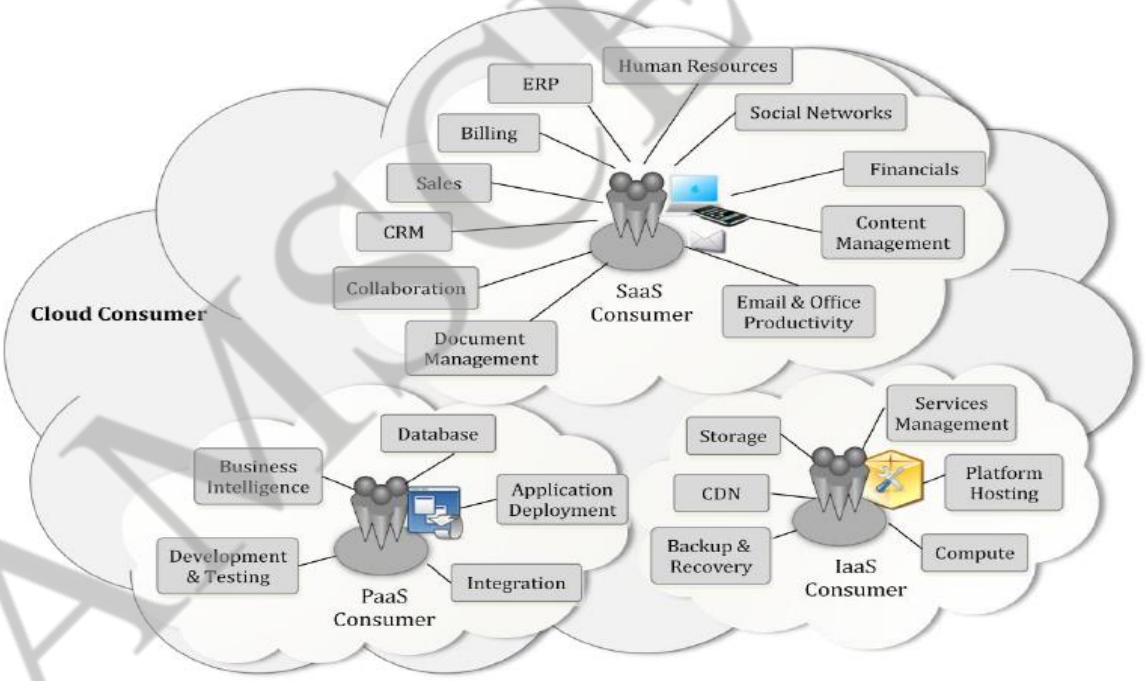


Cloud Consumer

- 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.

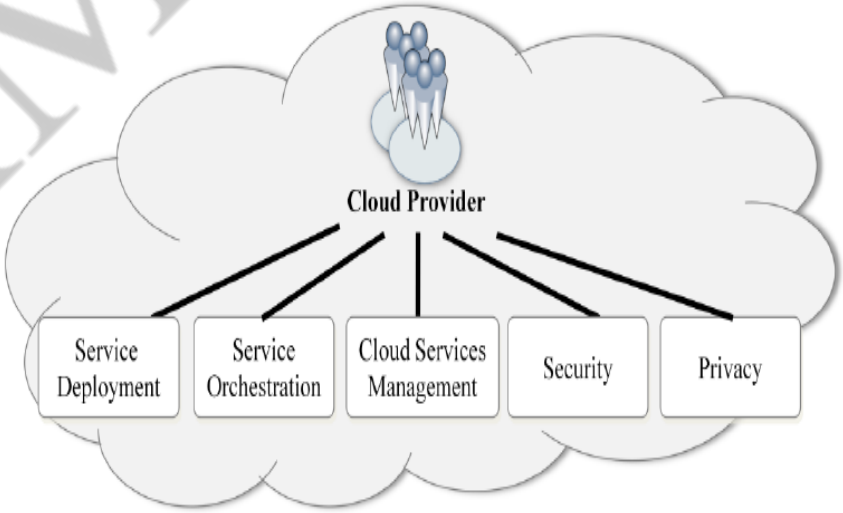
- Cloud consumers of PaaScan employ the tools and execution resources provided by cloud providers to develop, test, deploy and manage the applications.
- PaaS consumers can be application developers or application testers who run and test applications in cloud-based environments,.
- PaaS consumers can 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.

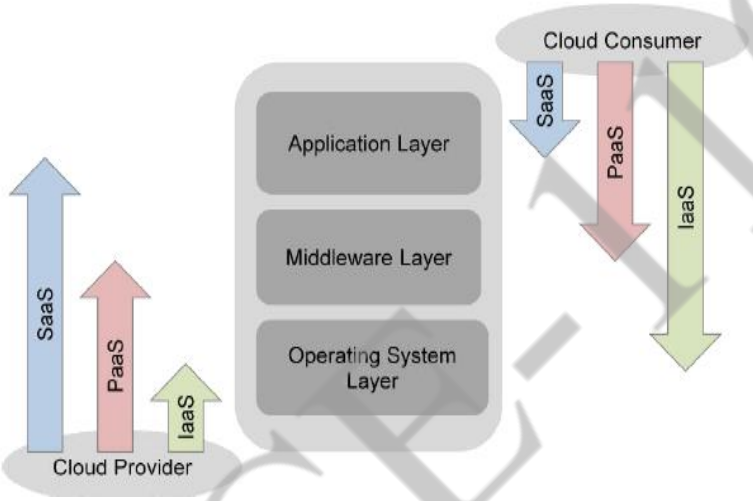
Cloud Carrier

- 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.

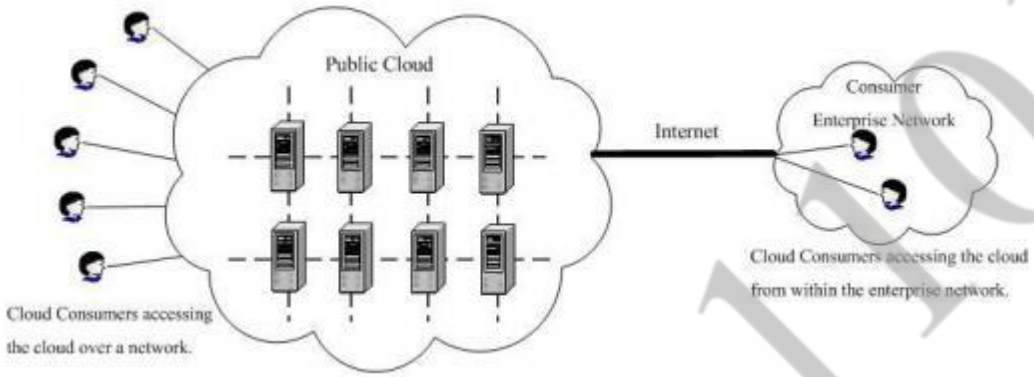
- The middleware layer provides software building blocks (e.g., libraries, database, and Java virtual machine) for developing application software in the cloud.
- Used by PaaS consumers, installed/ managed/ maintained by IaaS consumers or PaaS providers, and hidden from SaaS consumers.
- The OS layer 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

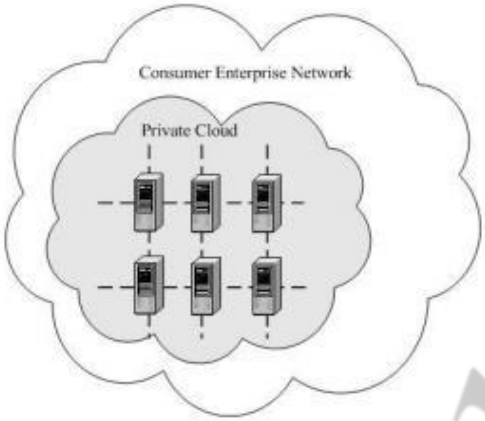
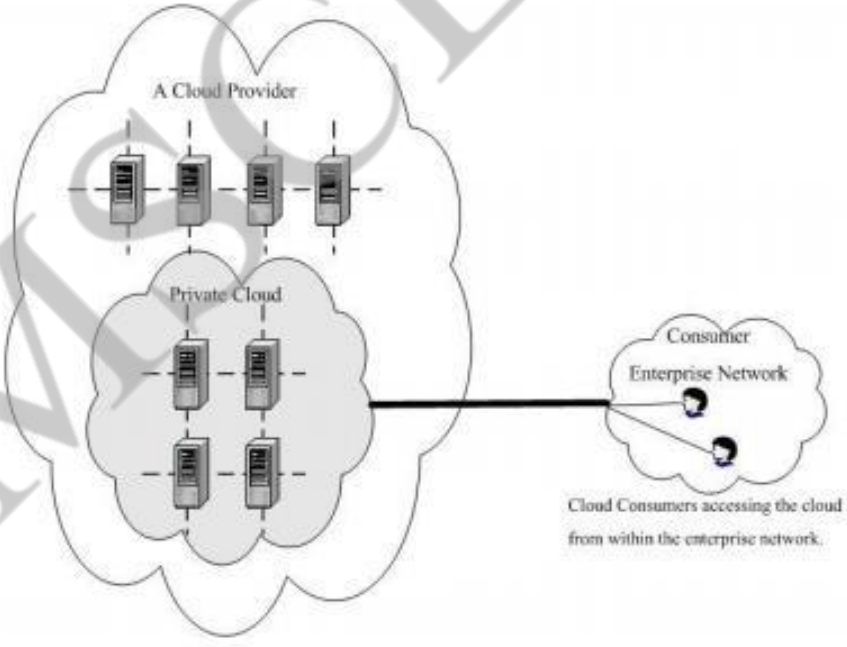


Fig: On-site private clouds

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.**

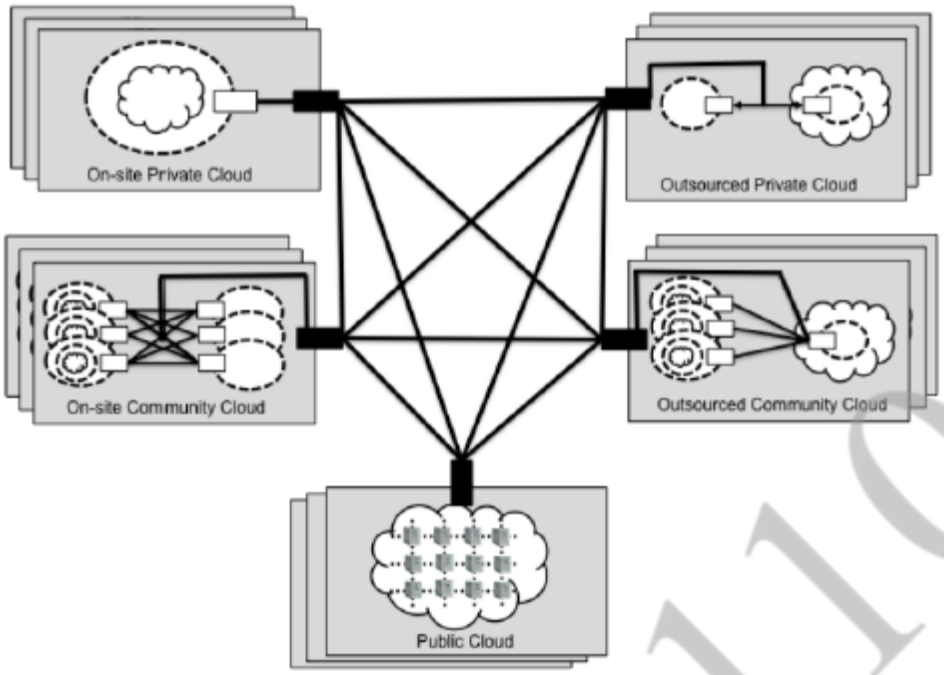


Fig:Hybrid Cloud

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

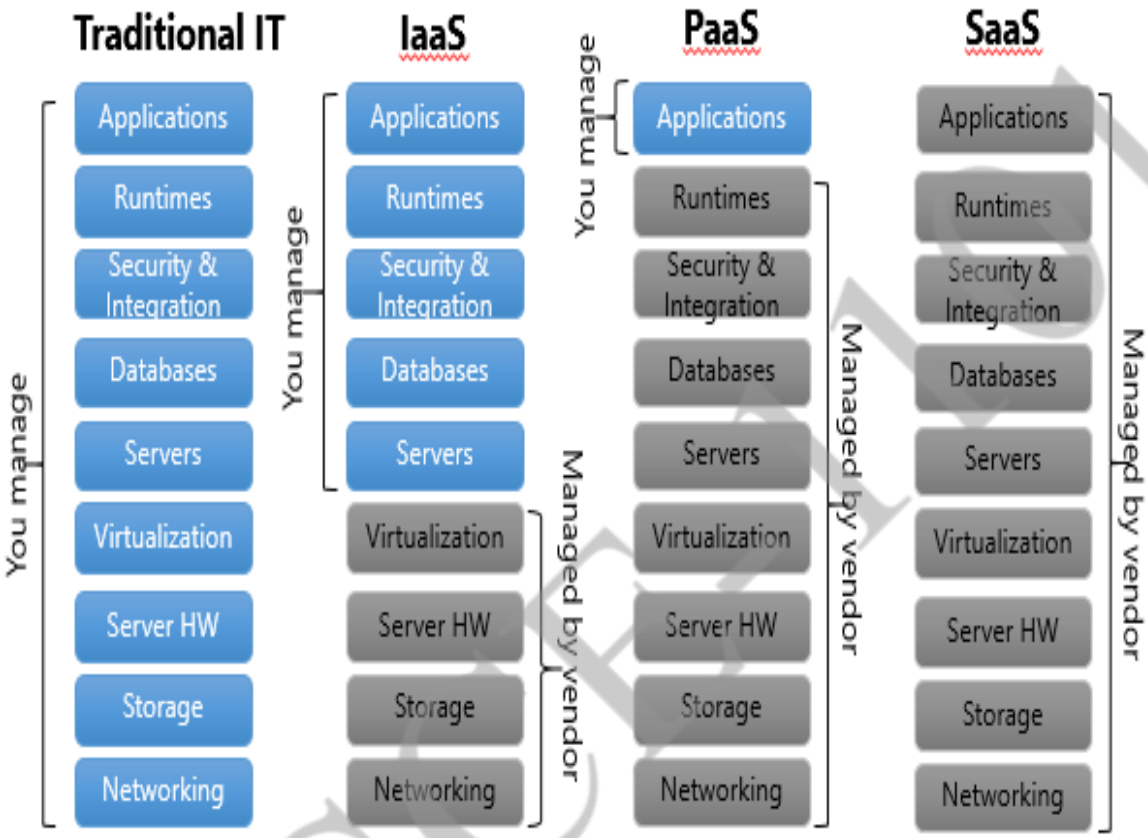
Disadvantage

- 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

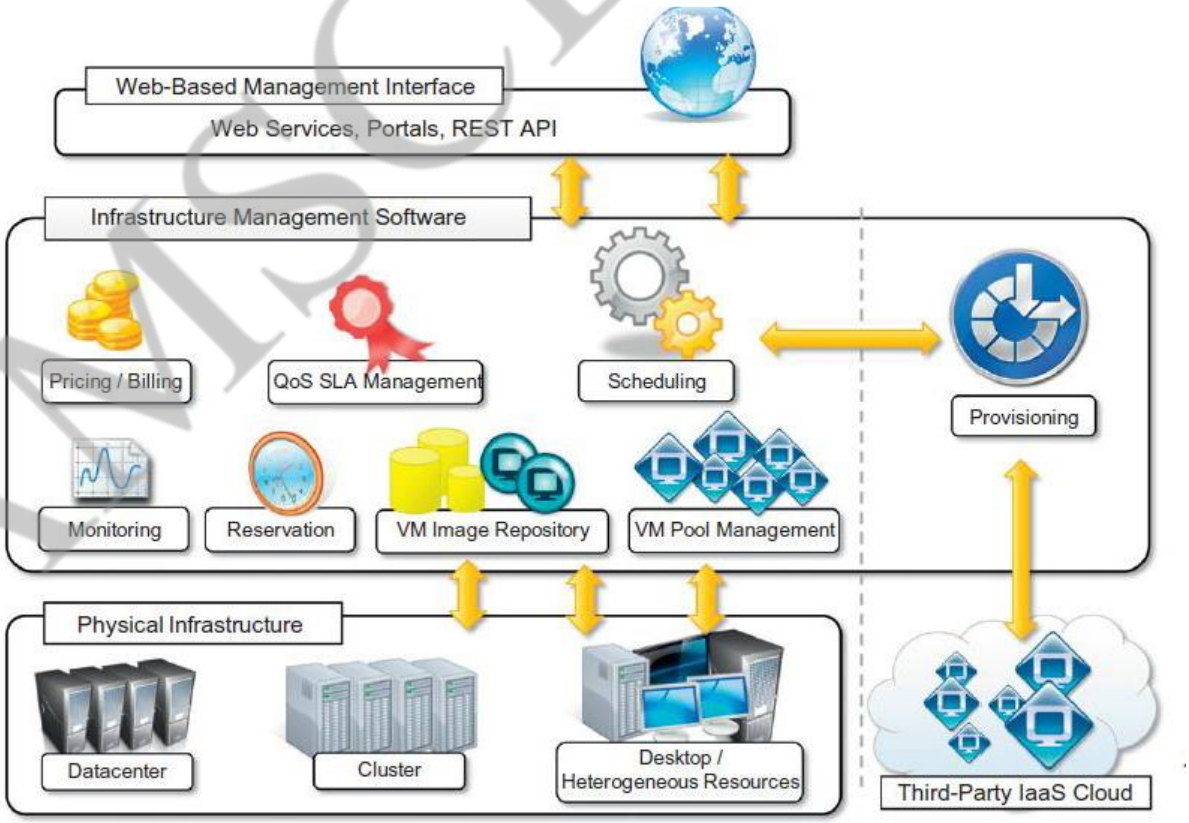
3.4.2Infrastructure as a Service (IaaS) (Hardware offerings on the cloud)

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.

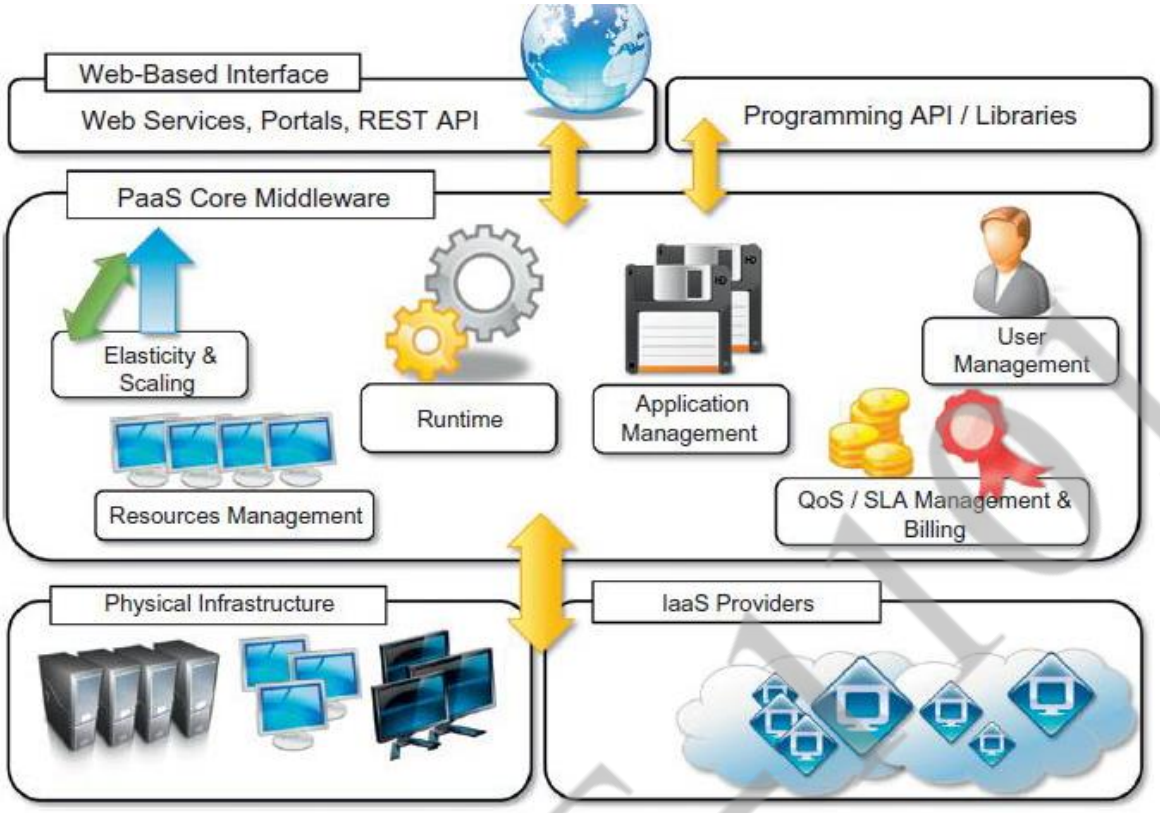
Have control over the deployed applications and possibly application hosting environment configurations.

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.

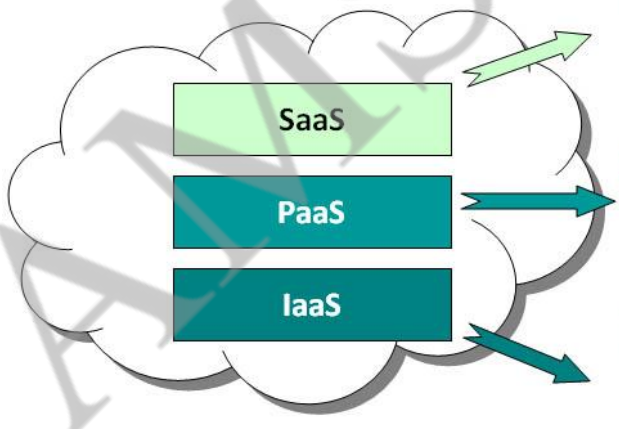
 Cloud services: helps the developers to simplify the creation and delivery cloud applications.

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

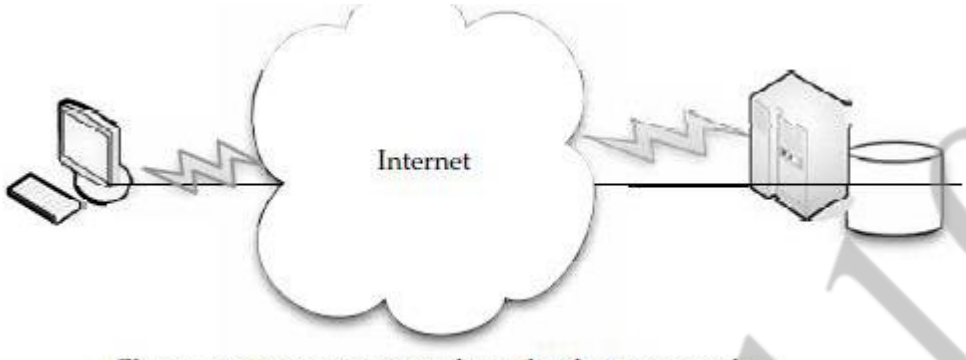
3.6.1 Storage as a Service

- 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.

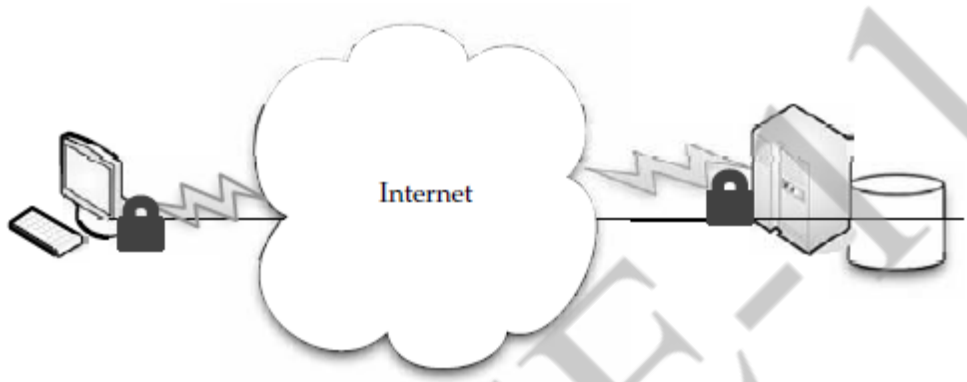
Authentication processes

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.

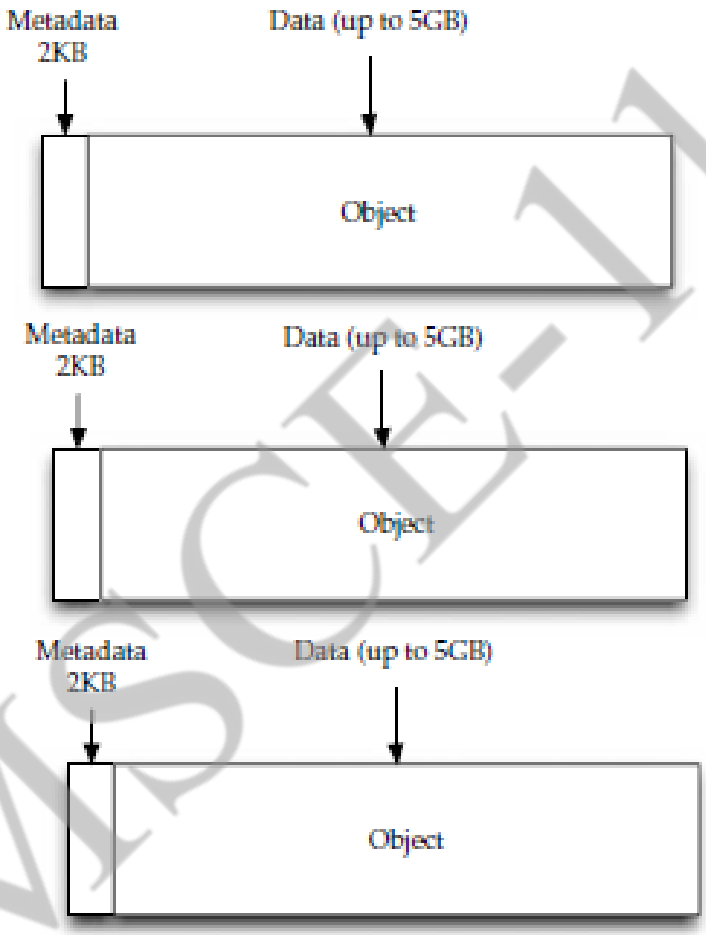
• **Small, well-understood building blocks** Do not try to provide a single service that does everything for everyone, but instead build small components that can be used as building blocks for other services.

• **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.

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

This section characterizes the various cloud service models and their extensions. Cloud resource management and intercloud resource exchange schemes are reviewed.

4.1.1 Extended Cloud Computing Services

The cloud platform provides PaaS, which sits on top of the IaaS infrastructure. The top layer offers SaaS. These must be implemented on the cloud platforms provided.

Fig 4.1.1 shows six layers of cloud services, ranging from hardware, network, and collocation to infrastructure, platform, and software applications.

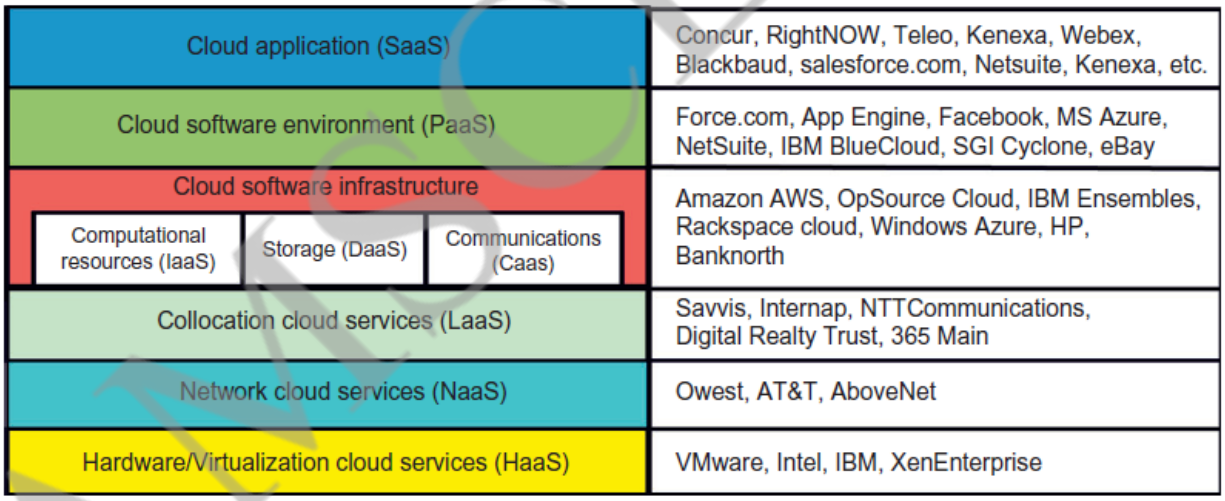


Fig: 4.1.1 A stack of six layers of cloud services and their providers.

The bottom three layers are more related to physical requirements. The bottommost layer provides Hardware as a Service (HaaS). The next layer is for interconnecting all the hardware components, and is simply called Network as a Service (NaaS). Virtual LANs fall within the scope of NaaS. The next layer up offers Location as a Service (LaaS), which provides a

collocation service to house, power, and secure all the physical hardware and network resources. The cloud infrastructure layer can be further subdivided as Data as a Service (DaaS) and Communication as a Service (CaaS) in addition to compute and storage in IaaS.

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

Table 4.1 Cloud Differences in Perspectives of Providers, Vendors, and Users

As shown in Table 4.1 cloud players are divided into three classes: (1) cloud service providers and IT administrators, (2) software developers or vendors, and (3) end users or business users. From the software vendors' perspective, application performance on a given cloud platform is most important. From the providers' perspective, cloud infrastructure performance is the primary concern. From the end users' perspective, the quality of services, including security, is the most important.

4.1.1.1 Cloud Service Tasks and Trends

Cloud services are introduced in five layers. The top layer is for SaaS applications, as further subdivided into the five application areas, mostly for business applications. CRM is heavily practiced in business promotion, direct sales, and marketing services. CRM offered the first SaaS on the cloud successfully. The approach is to widen market coverage by investigating customer behaviors and revealing opportunities by statistical analysis. SaaS tools also apply to distributed collaboration, and financial and human resources management.

Collocation services require multiple cloud providers to work together to support supply chains in manufacturing. Network cloud services provide communications such as those by AT&T, Qwest, and AboveNet.

4.1.1.2 Software Stack for Cloud Computing

Developers have to consider how to design the system to meet critical requirements such as high throughput, HA, and fault tolerance. Based on the observations of some typical cloud computing instances, such as Google, Microsoft, and Yahoo!, the overall 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 just as the traditional software stack does.

The platform for running cloud computing services can be either physical servers or virtual servers. By using VMs, the platform can be flexible, that is, the running services are not bound to specific hardware platforms. This brings flexibility to cloud computing platforms.

4.1.1.3 Runtime Support Services

Cluster monitoring is used to collect the runtime status of the entire cluster. The scheduler queues the tasks submitted to the whole cluster and assigns the tasks to the processing nodes according to node availability. The distributed scheduler for the cloud application has special characteristics that can support cloud applications, such as scheduling the programs written in MapReduce style. The runtime support system keeps the cloud cluster working properly with high efficiency.

As a result, 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. The customer data is stored in the cloud that is either vendor proprietary or a publicly hosted cloud supporting PaaS and IaaS.

4.2. Resource Provisioning and Platform Deployment

The emergence of computing clouds suggests fundamental changes in software and hardware architecture. Cloud architecture puts more emphasis on the number of processor cores or VM instances. Parallelism is exploited at the cluster node level.

4.2.1 Provisioning of Compute Resources (VMs)

Providers supply cloud services by signing SLAs with end users. The SLAs must commit sufficient resources such as CPU, memory, and bandwidth that the user can use for a preset period. Underprovisioning of resources will lead to broken SLAs and penalties. Overprovisioning of resources will lead to resource underutilization, and consequently, a decrease in revenue for the provider. Deploying an autonomous system to efficiently provision resources to users is a challenging problem. The difficulty comes from the unpredictability of consumer demand, software and hardware failures, heterogeneity of services, power management, and conflicts in signed SLAs between consumers and service providers.

Resource provisioning schemes also demand fast discovery of services and data in cloud computing infrastructures. In a virtualized cluster of servers, this demands efficient installation of VMs, live VM migration, and fast recovery from failures. To deploy VMs, users treat them as physical hosts with customized operating systems for specific applications.

The provider should offer resource-economic services. Power-efficient schemes for caching, query processing, and thermal management are mandatory due to increasing energy waste by heat dissipation from data centers Public or private clouds promise to streamline the on-demand provisioning of software, hardware, and data as a service, achieving economies of scale in IT deployment and operation.

4.2.2 Resource Provisioning Methods

Fig shows three cases of static cloud resource provisioning policies. In case (a), overprovisioning with the peak load causes heavy resource waste (shaded area). In case (b), underprovisioning (along the capacity line) of resources results in losses by both user and provider in that paid demand by the users (the shaded area above the capacity) is not served and wasted resources still exist for those demanded areas below the provisioned capacity. In case (c), the constant provisioning of resources with fixed capacity to a declining user demand could result in even worse resource waste.

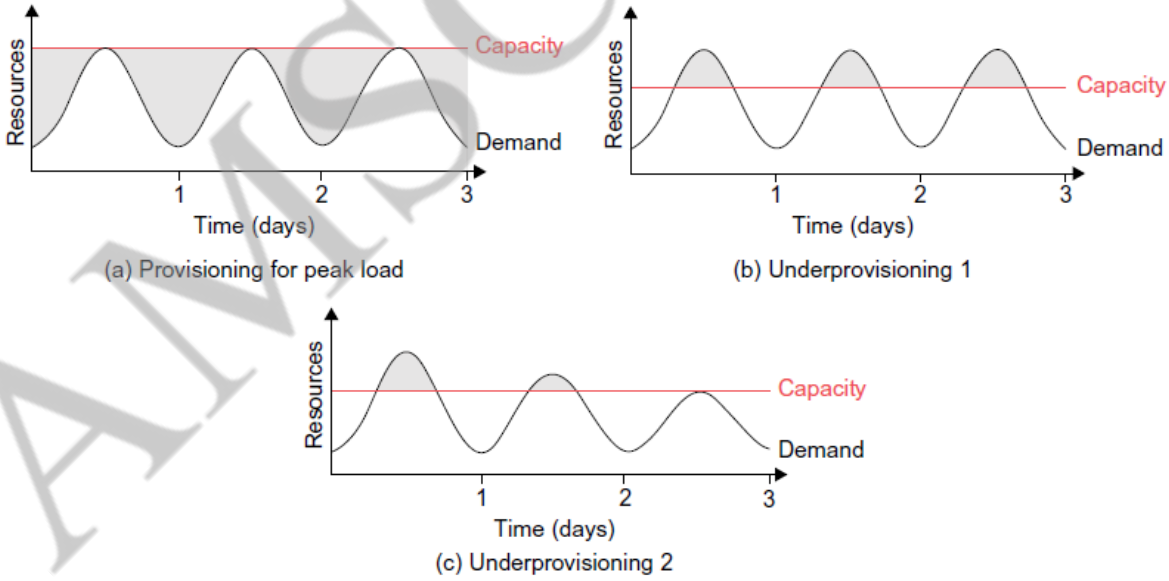
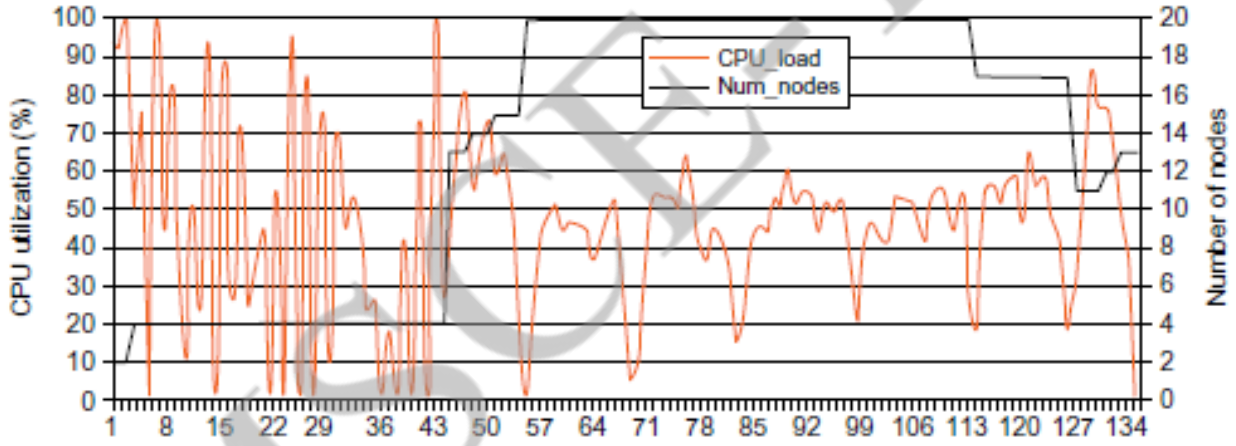


Fig: Three cases of cloud resource provisioning without elasticity: (a) heavy waste due to overprovisioning, (b) underprovisioning and (c) under- and then overprovisioning.

The user may give up the service by canceling the demand, resulting in reduced revenue for the provider. Both the user and provider may be losers in resource provisioning without elasticity. Three resource-provisioning methods are presented in the following sections. The *demand-driven method* provides static resources and has been used in grid computing for many years. The *eventdriven method* is based on predicted workload by time. The *popularity-driven method* is based on Internet traffic monitored.

4.2.2.1 Demand-Driven Resource Provisioning

- This method adds or removes computing instances based on the current utilization level of the allocated resources.
- The demand-driven method automatically allocates two Xeon processors for the user application, when the user was using one Xeon processor more than 60 percent of the time for an extended period.
- In general, when a resource has surpassed a threshold for a certain amount of time, the scheme increases that resource based on demand.
- When a resource is below a threshold for a certain amount of time, that resource could be decreased accordingly.
- Amazon implements such an auto-scale feature in its EC2 platform. This method is easy to implement.



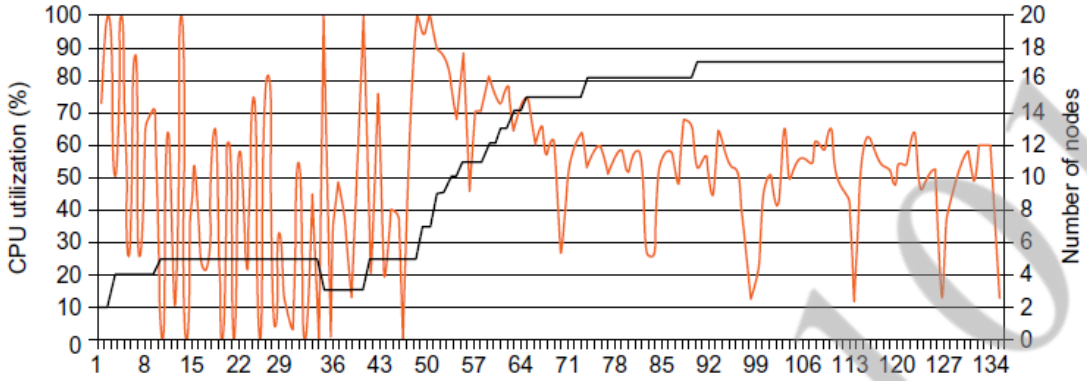
(a) Demand-driven

The x-axis in Fig(a) is the time scale in milliseconds. In the beginning, heavy fluctuations of CPU load are encountered. All three methods have demanded a few VM instances initially. Gradually, the utilization rate becomes more stabilized with a maximum of 20 VMs (100 percent utilization) provided for demand-driven provisioning in Fig(a). However, the event-driven method reaches a stable peak of 17 VMs toward the end of the event and drops quickly in Fig(b).

4.2.2.2 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

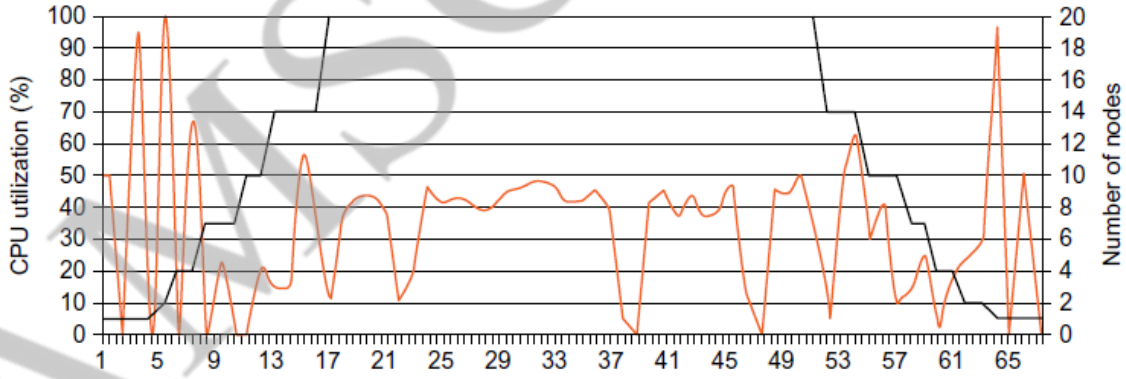
New Year in the East. During these events, the number of users grows before the event period and then decreases during the event period. This scheme anticipates peak traffic before it happens. The method results in a minimal loss of QoS, if the event is predicted correctly. Otherwise, wasted resources are even greater due to events that do not follow a fixed pattern.



(b) Event-driven

4.2.2.3 Popularity-Driven Resource Provisioning

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



(c) Popularity-drivern

In Fig(c) EC2 performance by CPU utilization rate (the dark curve with the percentage scale shown on the left) is plotted against the number of VMs provisioned (the light curves with scale shown on the right, with a maximum of 20 VMs provisioned).

4.2.3 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. The Inter- Grid is a Java-implemented software system that lets users create execution cloud environments on top of all participating grid resources. Peering arrangements established between gateways enable the allocation of resources from multiple grids to establish the execution environment.

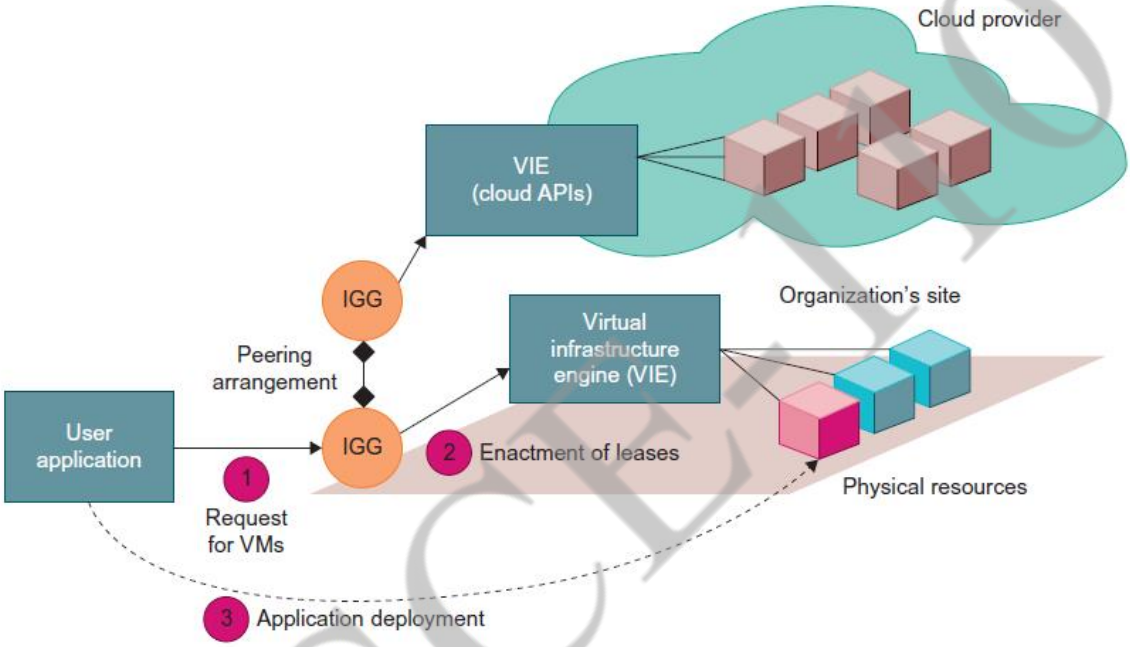


Fig Cloud resource deployment using an IGG (intergrid gateway) to allocate the VMs from a Local cluster to interact with the IGG of a public cloud provider

In Fig. a scenario is illustrated by which an intergrid gateway (IGG) allocates resources from a local cluster to deploy applications in three steps: (1) requesting the VMs, (2) enacting the leases, and (3) deploying the VMs as requested.

Under peak demand, this IGG interacts with another IGG that can allocate resources from a cloud computing 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. Received messages are handled in parallel by a thread pool.

4.2.4 Provisioning of Storage Resources

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

Table: Storage Services in Three Cloud Computing Systems

A distributed file system is very important for storing large-scale data. However, other forms of data storage also exist. Some data does not need the namespace of a tree structure file system, and instead, databases are built with stored data files. In cloud computing, another form of data storage is (Key, Value) pairs. Amazon S3 service uses SOAP to access the objects stored in the cloud.

Many cloud computing companies have developed large-scale data storage systems to keep huge amount of data collected every day. For example, Google's GFS stores web data and some other data, such as geographic data for Google Earth. A similar system from the open source community is the Hadoop Distributed File System (HDFS) for Apache. Hadoop is the open source implementation of Google's cloud computing infrastructure. Similar systems include Microsoft's Cosmos file system for the cloud.

Thus, such developers can think in the same way they do for traditional software development.

Hence, in cloud computing, it is necessary to build databases like large-scale systems based on data storage or distributed file systems.

4.3 Global Exchange of Cloud Resources

In order to support a large number of application service consumers from around the world, cloud infrastructure providers (i.e., IaaS providers) have established data centers in multiple geographical locations to provide redundancy and ensure reliability in case of site failures.

For example, Amazon has data centers in the United States (e.g., one on the East Coast and another on the West Coast) and Europe. However, currently Amazon expects its cloud customers (i.e., SaaS providers) to express a preference regarding where they want their application services to be hosted. Amazon does not provide seamless/automatic mechanisms for scaling its hosted services across multiple geographically distributed data centers.

In addition, no single cloud infrastructure provider will be able to establish its data centers at all possible locations throughout the world. As a result, cloud application service (SaaS) providers will have difficulty in meeting QoS expectations for all their consumers. Hence, they would like to make use of services of multiple cloud infrastructure service providers who can provide better support for their specific consumer needs. This kind of requirement often arises in enterprises with global operations and applications such as Internet services, media hosting, and Web 2.0 applications. This necessitates federation of cloud infrastructure service providers for seamless provisioning of services across different cloud providers.

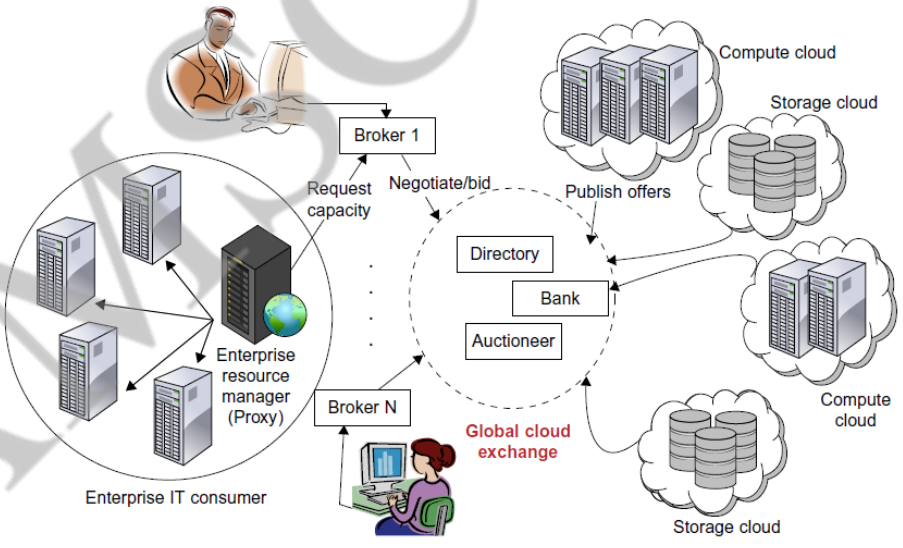


Fig: Inter-cloud exchange of cloud resources through brokering

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.

The availability of a banking system within the market ensures that financial transactions pertaining to SLAs between participants are carried out in a secure and dependable environment.

4.4 Cloud Security

Lacking trust between service providers and cloud users has hindered the universal acceptance of cloud computing as a service on demand. In the past, trust models have been developed to protect mainly e-commerce and online shopping provided by eBay and Amazon. For web and cloud services, trust and security become even more demanding, because leaving user applications completely to the cloud providers has faced strong resistance by most PC and server users. Cloud platforms become worrisome to some users for lack of privacy protection, security assurance, and copyright protection. Trust is a social problem, not a pure technical issue. However, the social problem can be solved with a technical approach.

Common sense dictates that technology can enhance trust, justice, reputation, credit, and assurance in Internet applications. As a virtual environment, the cloud poses new security threats that are more difficult to contain than traditional client and server configurations. To solve these trust problems, a new data-protection model is presented in this section. In many cases, one can extend the trust models for P2P networks and grid systems to protect clouds and data centers.

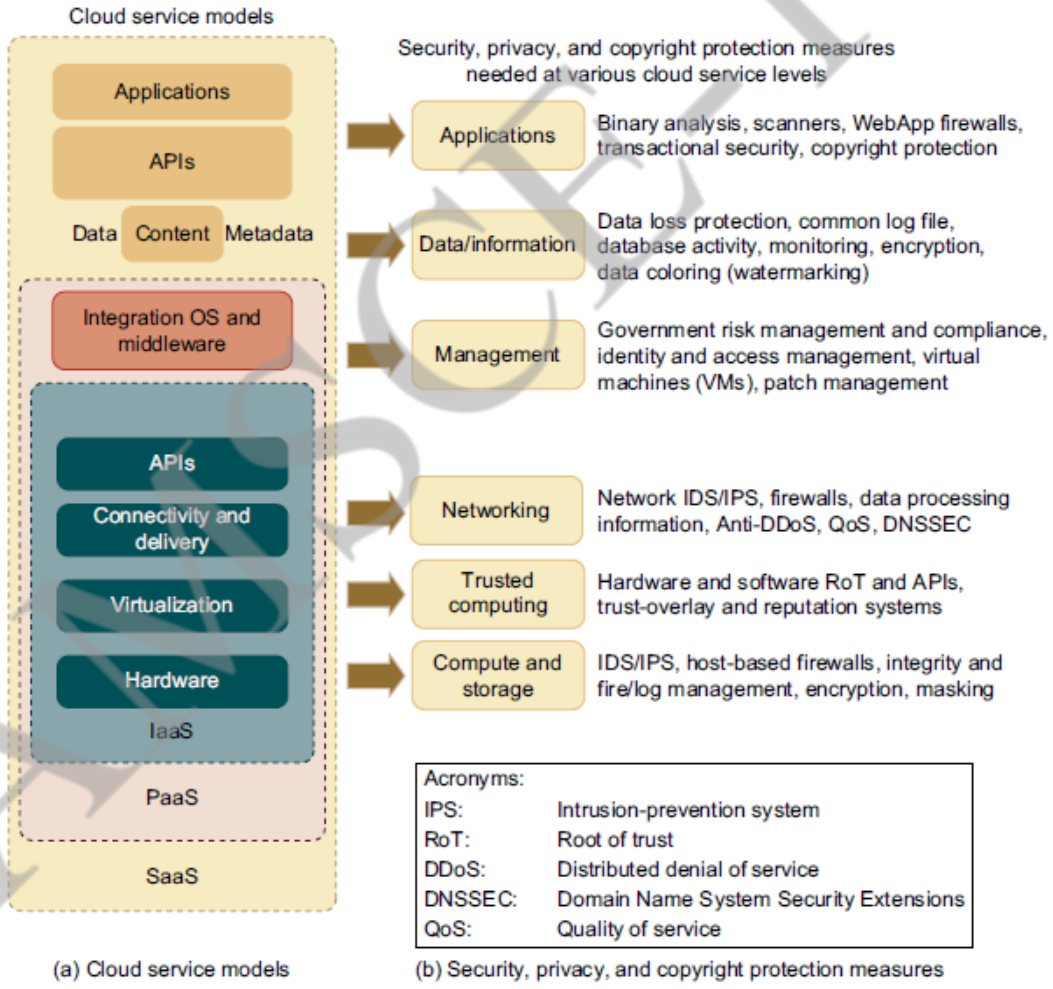
4.4.1 Cloud Security Defense Strategies

A healthy cloud ecosystem is desired to free users from abuses, violence, cheating, hacking, viruses, rumors, pornography, spam, and privacy and copyright violations. The security demands

of three cloud service models, IaaS, PaaS, and SaaS, are described in this section. These security models are based on various SLAs between providers and users.

Three basic cloud security enforcements are expected.

- First, facility security in data centers demands on-site security year round. Biometric readers, CCTV (close-circuit TV), motion detection, and man traps are often deployed.
- Also, network security demands fault-tolerant external firewalls, intrusion detection systems (IDSes), and third-party vulnerability assessment.
- Finally, platform security demands SSL and data decryption, strict password policies, and system trust certification.



(a) Cloud service models

(b) Security, privacy, and copyright protection measures

Fig: Cloud service models on the left (a) and corresponding security measures on the right (b); the IaaS is at the innermost level, PaaS is at the middle level, and SaaS is at the outermost level, including all hardware, software, datasets, and networking resources.

Thus, security defenses are needed to protect all cluster servers and data centers. Here are some cloud components that demand special security protection:

• Protection of servers from malicious software attacks such as worms, viruses, and malware

• Protection of hypervisors or VM monitors from software-based attacks and vulnerabilities

• Protection of VMs and monitors from service disruption and DoS attacks

• Protection of data and information from theft, corruption, and natural disasters

• Providing authenticated and authorized access to critical data and services

Protection Schemes	Brief Description and Deployment Suggestions
Secure data centers and computer buildings	Choose hazard-free location, enforce building safety. Avoid windows, keep buffer zone around the site, bomb detection, camera surveillance, earthquake-proof, etc.
Use redundant utilities at multiple sites	Multiple power and supplies, alternate network connections, multiple databases at separate sites, data consistency, data watermarking, user authentication, etc.
Trust delegation and negotiation	Cross certificates to delegate trust across PKI domains for various data centers, trust negotiation among certificate authorities (CAs) to resolve policy conflicts
Worm containment and DDoS defense	Internet worm containment and distributed defense against DDoS attacks to secure all data centers and cloud platforms
Reputation system for data centers	Reputation system could be built with P2P technology; one can build a hierarchy of reputation systems from data centers to distributed file systems
Fine-grained file access control	Fine-grained access control at the file or object level; this adds to security protection beyond firewalls and IDSes
Copyright protection and piracy prevention	Piracy prevention achieved with peer collusion prevention, filtering of poisoned content, nondestructive read, alteration detection, etc.
Privacy protection	Uses double authentication, biometric identification, intrusion detection and disaster recovery, privacy enforcement by data watermarking, data classification, etc.

Table: Physical and Cyber Security Protection at Cloud/Data Centers

4.5. Cloud Security Challenges

Although virtualization and cloud computing can help companies accomplish more by breaking the physical bonds between an IT infrastructure and its users, heightened security threats must be overcome in order to benefit fully from this new computing paradigm.

This is particularly true for the SaaS provider.

For example, in the cloud, you lose control over assets in some respects, so your security model must be reassessed. Enterprise security is only as good as the least reliable partner, department, or vendor.

With the cloud model, you lose control over physical security.

In a public cloud, you are sharing computing resources with other companies.

Simply because you share the environment in the cloud, may put your data at risk of seizure.

"sticky services"—services that an end user may have difficulty transporting from one cloud vendor to another if Storage services provided by one cloud vendor may be incompatible (e.g., Amazon's "Simple Storage Service" [S3] is incompatible with IBM's Blue Cloud, or Google, or Dell).

If information is encrypted while passing through the cloud, who controls the encryption/decryption keys. Is it the customer or the cloud vendor?
Most customers probably want their data encrypted both ways (customer & Cloud vendor) across the Internet using SSL (Secure Sockets Layer protocol).
They also most likely want their data encrypted while it is at rest in the cloud vendor's storage pool. Be sure that you, the customer, control the encryption/decryption keys, just as if the data were still resident on your own servers.

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. Ensuring the integrity of the data really means that it changes only in response to authorized transactions.

Using SaaS offerings in the cloud means that there is much less need for software development. Our development tool of choice should have a security model embedded in it to guide developers during the development phase and restrict users only to their authorized data when the system is deployed into production.

As more and more mission-critical processes are moved to the cloud, SaaS suppliers will have to provide log data in a real-time, straightforward manner, probably for their administrators as well as their customers' personnel. Someone has to be responsible for monitoring for security and compliance, and unless the application and data are under the control of end users, they will not be able to.
Since the SaaS provider's logs are internal and not necessarily accessible externally or by clients or investigators, monitoring is difficult.

The speed at which applications will change in the cloud will affect both the SDLC and security. Even worse, a secure SLDC will not be able to provide a security cycle that keeps up with changes that occur so quickly.

One of the key challenges in cloud computing is data-level security. There is a huge body of standards that apply for IT security and compliance, governing most business interactions that will, over time, have to be translated to the cloud.
SaaS makes the process of compliance more complicated, since it may be difficult for a customer to discern where its data resides on a network controlled by its SaaS provider, or a partner of that provider, which raises all sorts of compliance issues of data privacy, segregation, and security.

Some countries have strict limits on what data about its citizens can be stored and for how long, and some banking regulators require that customers' financial data remain in their home country.

There is a perception that cloud computing removes data compliance responsibility; however, it should be emphasized that the data owner is still fully responsible for compliance. Those who adopt cloud computing must remember that it is the responsibility of the data owner, not the service provider, to secure valuable data.

Security managers will need to pay particular attention to systems that contain critical data such as corporate financial information or source code during the transition to server virtualization in production environments.

Security managers will need to work with their company's legal staff to ensure that appropriate contract terms are in place to protect corporate data and provide for acceptable service-level agreements.

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.

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. Administrative access is through the Internet, this increases risk and exposure and will require stringent monitoring for changes in system control and access control restriction.

The dynamic and fluid nature of virtual machines will make it difficult to maintain the consistency of security and ensure the audit ability of records.
Proving the security state of a system and identifying the location of an insecure virtual machine will be challenging. The intrusion detection and prevention systems will need to be able to detect malicious activity at virtual machine level.

Virtual machines are vulnerable as they move between the private cloud and the public cloud. A fully or partially shared cloud environment is expected to have a greater attack surface and therefore can be considered to be at greater risk than a dedicated resources environment.

Enterprises are often required to prove that their security compliance is in accord with regulations, standards, and auditing practices, regardless of the location of the systems at which the data resides. Data is fluid in cloud computing and may reside in on-premises physical servers, on-premises virtual machines, or off-premises virtual machines running on cloud computing resources, and this will require some rethinking on the part of auditors and practitioners alike.

In the rush to take advantage of the benefits of cloud computing, not least of which is significant cost savings, many corporations are likely rushing into cloud computing without a serious consideration of the security implications.

To establish zones of trust in the cloud, the virtual machines must be self-defending, effectively moving the perimeter to the virtual machine itself. Enterprise perimeter security (i.e., firewalls, demilitarized zones [DMZs], network segmentation, intrusion detection and prevention systems [IDS/IPS], monitoring tools, and the associated security policies) only controls the data that resides and transits behind the perimeter. In the cloud computing world, the cloud computing provider is in charge of customer data security and privacy.

4.6. Software-as-a-Service 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 cloud computing are creating not only new technologies and business operational processes but also new security requirements and challenges.

As the most recent evolutionary step in the cloud service model (see Figure 6.2), SaaS will likely remain the dominant cloud service model for the foreseeable future and the area where the most critical need for security practices and oversight will reside.

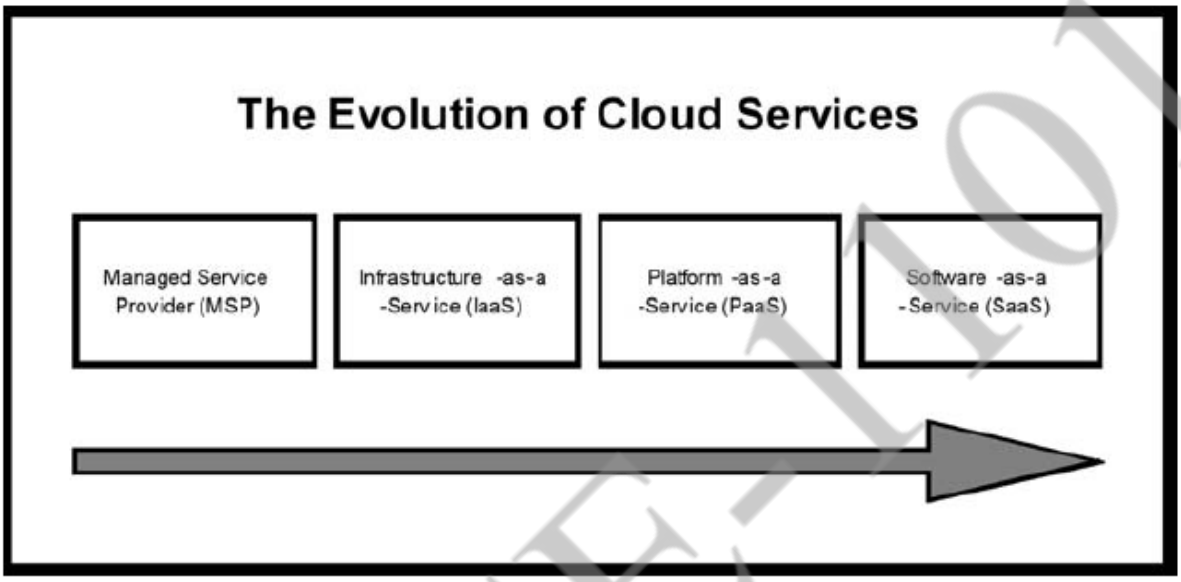


Figure 6.2 The evolution of cloud services.

Just as with an managed service provider, corporations or end users will need to research vendors' policies on data security before using vendor services to avoid losing or not being able to access their data. The technology analyst and consulting firm Gartner lists seven security issues which one should discuss with a 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?

Determining data security is harder today, so data security functions have become more critical than they have been in the past.

A tactic not covered by Gartner is to encrypt the data yourself.

If we encrypt the data using a trusted algorithm, then regardless of the service provider's security and encryption policies, the data will only be accessible with the decryption keys.

Of course, this leads to a follow-on problem: How do you manage private keys in a pay-on-demand computing infrastructure?

To address the security issues listed above, SaaS providers will need to incorporate and enhance security practices used by the managed service providers and develop new ones as the cloud computing environment evolves.

4.7. Security Governance

A security steering committee should be developed whose objective is to focus on providing guidance about security initiatives and alignment 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, application security, 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 because the security team is not focused on the key security functions and activities that are critical to the business.

4.8. Virtual Machine Security

In the cloud environment, physical servers are consolidated to multiple virtual machine instances on virtualized servers. Not only can data center security teams replicate typical security controls for the data center at large to secure the virtual machines, they can also advise their customers on how to prepare these machines for migration to a cloud environment when appropriate.

Firewalls, intrusion detection and prevention, integrity monitoring, and log inspection can all be deployed as software on virtual machines to increase protection and maintain compliance integrity of servers and applications as virtual resources move from on-premises to public cloud environments.

By deploying this traditional line of defense to the virtual machine itself, you can enable critical applications and data to be moved to the cloud securely. To facilitate the centralized management of a server firewall policy, the security software loaded onto a virtual machine should include a bidirectional stateful firewall that enables virtual machine isolation and location awareness, thereby enabling a tightened policy and the flexibility to move the virtual machine from on-premises to cloud resources. Integrity monitoring and log inspection software must be applied at the virtual machine level.

This approach to virtual machine security, which connects the machine back to the mother ship, has some advantages in that the security software can be put into a single software agent that provides for consistent control and management throughout the cloud while integrating

seamlessly back into existing security infrastructure investments, providing economies of scale, deployment, and cost savings for both the service provider and the enterprise.

4.8.1 Privacy and Copyright Protection

Here are several security features desired in a secure cloud:

• Dynamic web services with full support from secure web technologies

• Established trust between users and providers through SLAs and reputation systems

• Effective user identity management and data-access management

• Single sign-on and single sign-off to reduce security enforcement overhead

• Auditing and copyright compliance through proactive enforcement

• Shifting of control of data operations from the client environment to cloud providers

• Protection of sensitive and regulated information in a shared environment

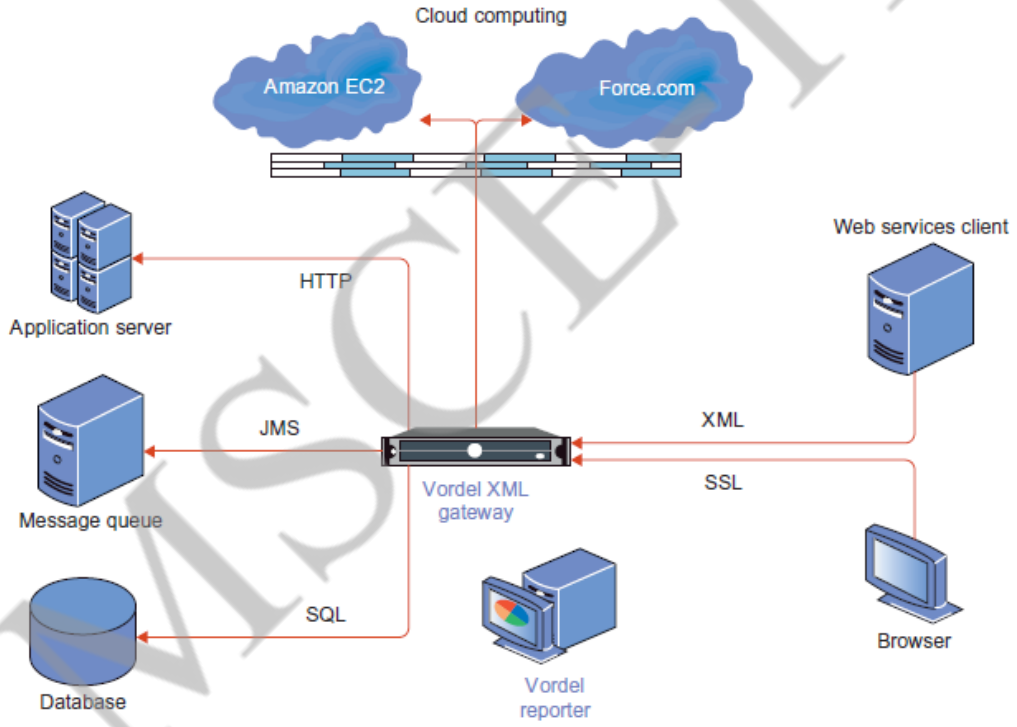


Fig: The typical security structure coordinated by a secured gateway plus external firewalls to safeguard the access of public or private clouds.

4.9. Identity Access Management (IAM)

Identity and access management is a critical function for every organization, and a fundamental expectation of SaaS customers is that the principle of least privilege is granted to their data.

The principle of least privilege states that only the minimum access necessary to perform an operation should be granted, and that access should be granted only for the minimum amount of time necessary.
However, business and IT groups will need and expect access to systems and applications. The advent of cloud services and services on demand is changing the identity management landscape. Most of the current identity management solutions are focused on the enterprise and typically are architected to work in a very controlled, static environment. User-centric identity management solutions such as federated identity management also make some assumptions about the parties involved and their related services.

Identity and Access Management can be broken down into three distinct capabilities

Identity Governance:
The ability in making sure the right people are granted the right access rights, making sure the wrong ones are not and managing the lifecycle through organization
structure, processes and enabling technology.

Directory Services:
The ability in enforcing access rights, within specified policy, when users attempt to access a desired application, system or platform.

Access Management:
The ability to provide ways to control storage of identity information about users and access rights.

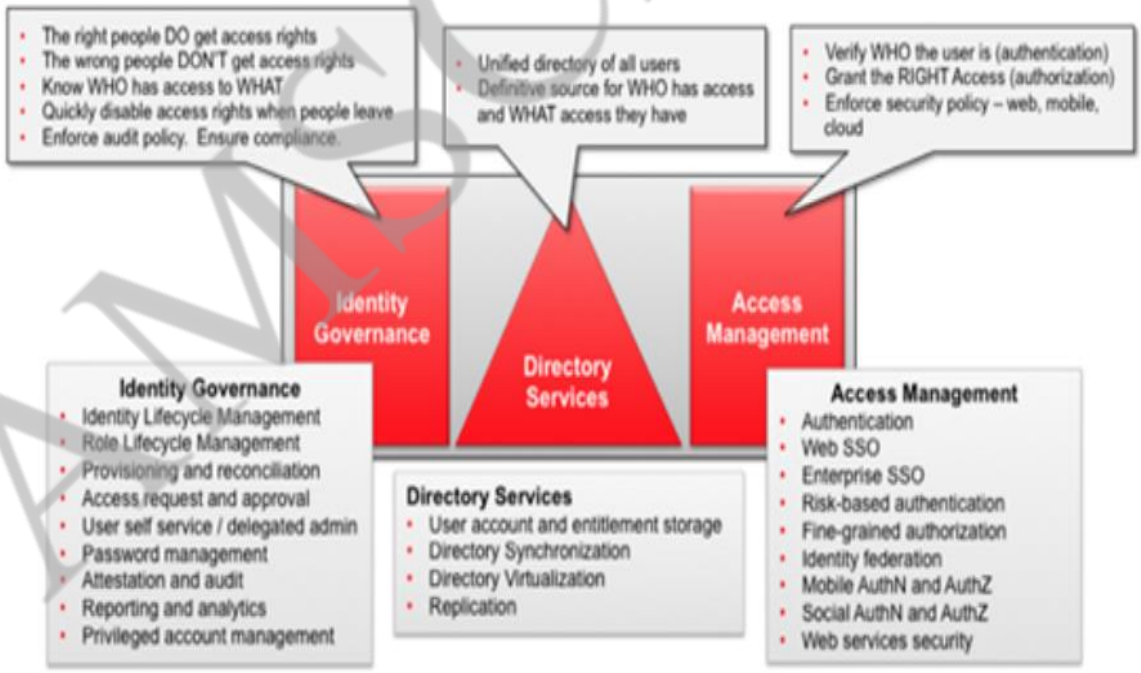


Fig: IAM Architecture

In the cloud environment, where services are offered on demand and they can continuously evolve, aspects of current models such as trust assumptions, privacy implications, and operational aspects of authentication and authorization, will be challenged. Meeting these challenges will require a balancing act for SaaS providers as they evaluate new models and management processes for IAM to provide end-to-end trust and identity throughout the cloud and the enterprise. Another issue will be finding the right balance between usability and security. If a good balance is not achieved, both business and IT groups may be affected by barriers to completing their support and maintenance activities efficiently.

4.10. 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.

4.10.1.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, version1.1). 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.

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 personidentified 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.

4.10.2.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.

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. It is important to understand that security and privacy are not guaranteed by the protocol.

In fact, OAuth by itself *provides no privacy at all* and depends on other protocols such as SSL to accomplish that. OAuth can be implemented in a secure manner, however.

In fact, the specification includes substantial security considerations that must be taken into account when working with sensitive data. With Oauth, sites use tokens coupled with shared secrets to access resources. Secrets, just like passwords, must be protected.

4.10.3.OpenID

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.

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.

After the OpenID identifier has been verified, OpenID authentication is considered successful and the user is considered logged in to the relying party web site. The web site typically then stores the OpenID identifier in the user's session. OpenID does not provide its own authentication methods, but if an identity provider uses strong authentication, OpenID can be used for secure transactions.

4.10.4.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. 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 oneway—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 of Certification Authorities (CAs).

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 as mutual (assured) authentication.

Mutual authentication requires the TLS clientside 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

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. Hadoop

Hadoop is an open source implementation of MapReduce coded and released in Java (rather than C) by Apache. The Hadoop implementation of MapReduce uses the Hadoop Distributed File System (HDFS) as its underlying layer rather than GFS. The Hadoop core is divided into two fundamental layers: the MapReduce engine and HDFS. The MapReduce engine is the computation engine running on top of HDFS as its data storage manager. The following two sections cover the details of these two fundamental layers.

HDFS: HDFS is a distributed file system inspired by GFS that organizes files and stores their data on a distributed computing system.

HDFS Architecture: HDFS has a master/slave architecture containing a single NameNode as the master and a number of DataNodes as workers (slaves). To store a file in this architecture, HDFS splits the file into fixed-size blocks (e.g., 64 MB) and stores them on workers (DataNodes). The mapping of blocks to DataNodes is determined by the NameNode. The NameNode (master) also manages the file system's metadata and namespace. In such systems, the namespace is the area maintaining the metadata, and metadata refers to all the information stored by a file system that is needed for overall management of all files.

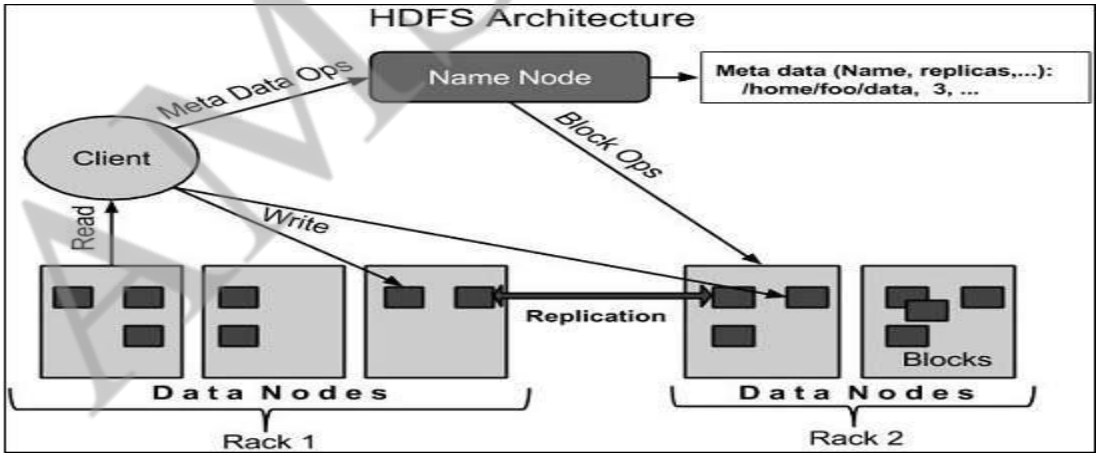


Fig: HDFS Architecture

HDFS follows the master-slave architecture and it has the following elements:

Namenode - The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks −

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

Datanode - The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

Block - Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

HDFS Features: Distributed file systems have special requirements, such as performance, scalability, concurrency control, fault tolerance, and security requirements, to operate efficiently. However, because HDFS is not a general-purpose file system, as it only executes specific types of applications, it does not need all the requirements of a general distributed file system. For example, security has never been supported for HDFS systems.

HDFS Fault Tolerance: One of the main aspects of HDFS is its fault tolerance characteristic. Since Hadoop is designed to be deployed on low-cost hardware by default, a hardware failure in this system is considered to be common rather than an exception. Therefore, Hadoop considers the following issues to fulfill reliability requirements of the file system:

- **Block replication** To reliably store data in HDFS, file blocks are replicated in this system. In other words, 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. Although storing replicas on different nodes (DataNodes) located in different racks across the whole cluster provides more reliability, it is sometimes ignored as the cost of communication between two nodes in different racks is relatively high in comparison with that of different nodes located in the same rack. Therefore, sometimes HDFS compromises its reliability to achieve lower communication costs. For example, for the default replication factor of three, HDFS stores one replica in the same node the original data is stored, one replica on a different node but in the same rack, and one replica on a different node in a different rack to provide three copies of the data.
- **Heartbeat and Blockreport messages** Heartbeats and Blockreports are periodic messages sent to the NameNode by each DataNode in a cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly, while each Blockreport contains a list of all blocks on a DataNode [65]. The NameNode receives such messages because it is the sole decision maker of all replicas in the system.

HDFS High-Throughput Access to Large Data Sets (Files): Because HDFS is primarily designed for batch processing rather than interactive processing, data access throughput in HDFS is more important than latency. Also, because applications run on HDFS typically have large data sets, individual files are broken into large blocks (e.g., 64MB) to allow HDFS to decrease the amount of metadata storage required per file. This provides two advantages: The list of blocks per file will shrink as the size of individual blocks increases, and by keeping large amounts of data sequentially within a block, HDFS provides fast streaming reads of data.

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

- **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. Upon receiving such information,

the user calls the read function to connect to the closest DataNode containing the first block of the file. After the first block is streamed from the respective DataNode to the user, the established connection is terminated and the same process is repeated for all blocks of the requested file until the whole file is streamed to the user.

- **Writing to a file** To write a file in HDFS, a user sends a "create" request to the NameNode to create a new file in the file system namespace. If the file does not exist, the NameNode notifies the user and allows him to start writing data to the file by calling the write function. The first block of the file is written to an internal queue termed the data queue while a data streamer monitors its writing into a DataNode. Since 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.

5.2. Map Reduce

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

5.2.1. Architecture of MapReduce in Hadoop

The topmost layer of Hadoop is the MapReduce engine that manages the data flow and control flow of MapReduce jobs over distributed computing systems. Figure shows the MapReduce

engine architecture cooperating with HDFS. Similar to HDFS, the MapReduce engine also has a master/slave architecture consisting of a single JobTracker as the master and a number of TaskTrackers as the slaves (workers). The JobTracker manages the MapReduce job over a cluster and is responsible for monitoring jobs and assigning tasks to TaskTrackers. The TaskTracker manages the execution of the map and/or reduce tasks on a single computation node in the cluster.

Each TaskTracker node has a number of simultaneous execution slots, each executing either a map or a reduce task. Slots are defined as the number of simultaneous threads supported by CPUs of the TaskTracker node.

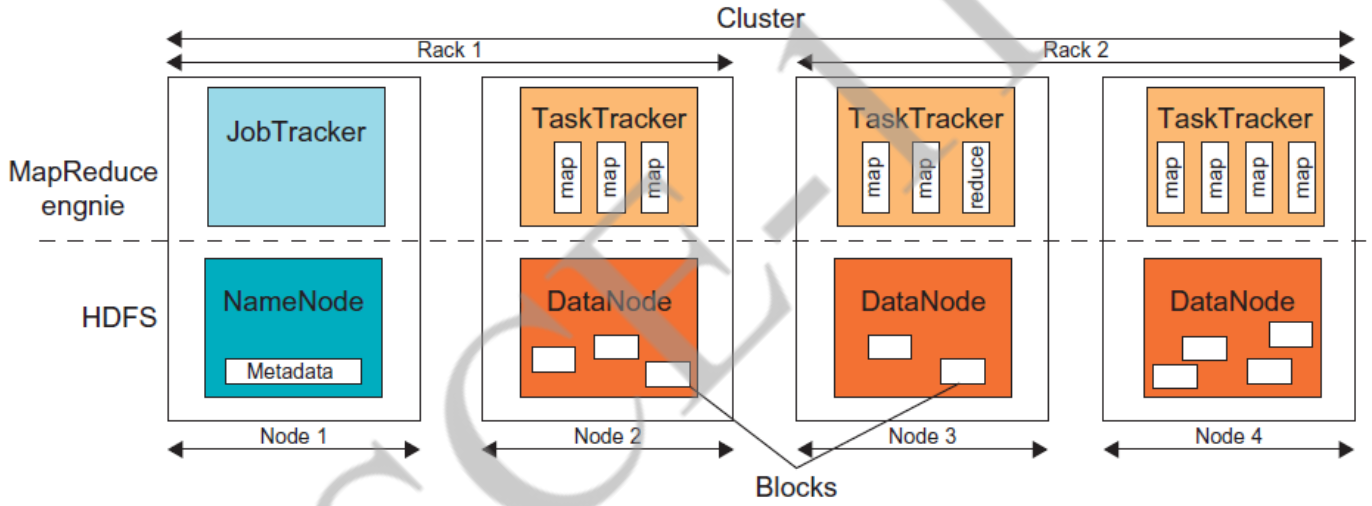


Fig: HDFS and MapReduce architecture in Hadoop where boxes with different shadings refer to different functional nodes applied to different blocks of data.

For example, a TaskTracker node with N CPUs, each supporting M threads, has M * N simultaneous execution slots. It is worth noting that each data block is processed by one map task running on a single slot. Therefore, there is a one-to-one correspondence between map tasks in a TaskTracker and data blocks in the respective DataNode.

5.2.2. Running a Job in Hadoop

Three components contribute in running a job in this system: a user node, a JobTracker, and several TaskTrackers. The data flow starts by calling the runJob(conf) function inside a user program running on the user node, in which conf is an object containing some tuning parameters

for the MapReduce framework and HDFS. The runJob(conf) function and conf are comparable to the MapReduce(Spec, &Results) function and Spec in the first implementation of MapReduce by Google. Figure depicts the data flow of running a MapReduce job in Hadoop.

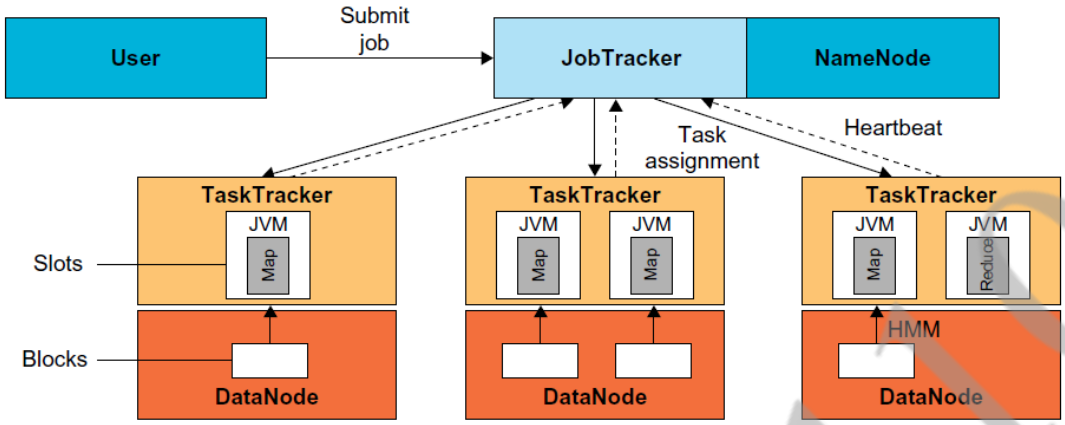


Fig: Data flow in running a MapReduce job at various task trackers using the Hadoop library.

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

• A user node asks for a new job ID from the JobTracker and computes input file splits.

• The user node copies some resources, such as the job's JAR file, configuration file, and computed input splits, to the JobTracker's file system.

• The user node submits the job to the JobTracker by calling the submitJob() function.

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

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

• Task running check A task running check is performed by receiving periodic heartbeat messages to the JobTracker from the TaskTrackers. Each heartbeat notifies the JobTracker that

the sending TaskTracker is alive, and whether the sending TaskTracker is ready to run a new task.

5.3. Virtual Box

Hardware virtualization solutions such as VMware Desktop, Virtual Box, and Parallels provide the ability to create a virtual computer with customized virtual hardware on top of which a new operating system can be installed.

Virtual Box is a general-purpose virtualization tool for x86 and x86-64 hardware, targeted at server, desktop, and embedded use.

It allows users and administrators to easily run multiple guest operating systems on a single host.

By default, the file system exposed by the virtual computer is completely separated from the one of the host machine. This becomes the perfect environment for running applications without affecting other users in the environment.

Users of Virtual Box can load multiple guest OSes under a single host operating-system (host OS). Each guest can be started, paused and stopped independently within its own (VM).

The user can independently configure each VM and run it under a choice of software-based virtualization or hardware assisted virtualization if the underlying host hardware supports this.

The host OS and guest OSs and applications can communicate with each other through a number of mechanisms including a common clipboard and a virtualized network facility.

Guest VMs can also directly communicate with each other if configured to do so. Virtual Box has supported Open Virtualization Format (OVF).

Features of Virtual Box :

- With Virtual Box, it’s also possible to share your clipboard between the virtualized and host operating system.
- Virtual Box truly has a lot of support because it's open-source and free
- Virtual Box is easy to install, takes a smaller amount of resources, and is many people’s first choice.
- Virtual Box 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.

- In Virtual Box, 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.

5.4. Google App Engine

App Engine is a fully managed, serverless platform for developing and hosting web applications at scale. You can choose from several popular languages, libraries, and frameworks to develop your apps, then let App Engine take care of provisioning servers and scaling your app instances based on demand.

Google App Engine (GAE) is a service for developing and hosting Web applications in Google's data centers, belonging to the platform as a service (PaaS) category of cloud computing. Web applications hosted on GAE are sandboxed and run across multiple servers for redundancy and allowing for scaling of resources according to the traffic requirements of the moment. App Engine automatically allocates additional resources to the servers to accommodate increased load.

Google App Engine is Google's platform as a service offering that allows developers and businesses to build and run applications using Google's advanced infrastructure. These applications are required to be written in one of a few supported languages, namely: Java, Python, PHP and Go. It also requires the use of Google query language and that the database used is Google Big Table. Applications must abide by these standards, so applications either must be developed with GAE in mind or else modified to meet the requirements.

GAE is a platform, so it provides all of the required elements to run and host Web applications, be it on mobile or Web. Without this all-in feature, developers would have to source their own servers, database software and the APIs that would make all of them work properly together, not to mention the entire configuration that must be done. GAE takes this burden off the developers so they can concentrate on the app front end and functionality, driving better user experience.

Advantages of GAE include:

- Readily available servers with no configuration requirement
- Power scaling function all the way down to "free" when resource usage is minimal
- Automated cloud computing tools

5.5. Programming Support of Google App Engine

5.5.1 Programming the Google App Engine

There are several powerful constructs for storing and accessing data. The data store is a NOSQL data management system for entities that can be, at most, 1 MB in size and are labeled by a set of schema-less properties. Queries can retrieve entities of a given kind filtered and sorted by the values of the properties.

Java offers Java Data Object (JDO) and Java Persistence API (JPA) interfaces implemented by the open source Data Nucleus Access platform, while Python has a SQL-like query language called GQL. The data store is strongly consistent and uses optimistic concurrency control.

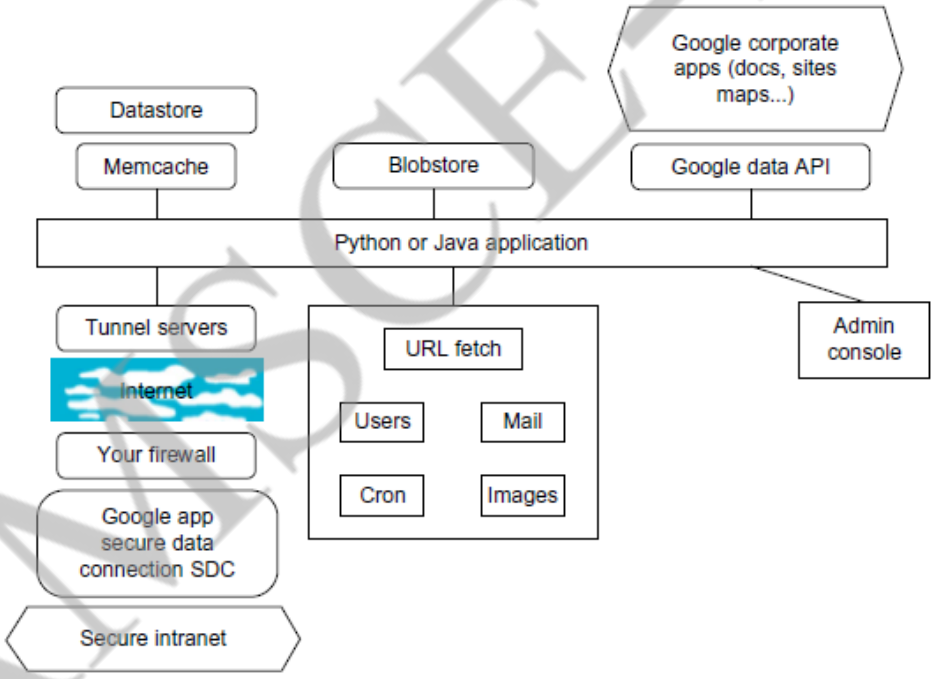


Fig: Programming environment for Google AppEngine

The data store implements transactions across its distributed network using “entity groups.” A transaction manipulates entities within a single group. Entities of the same group are stored together for efficient execution of transactions. Your GAE application can assign entities to

groups when the entities are created. The performance of the data store can be enhanced by in-memory caching using the memcache, which can also be used independently of the data store.

Alternatively, the application can perform tasks added to a queue by the application itself, such as a background task created while handling a request. 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.

5.5.1.1 Google File System (GFS)

GFS was designed for Google applications, and Google applications were built for GFS. In traditional file system design, such a philosophy is not attractive, as there should be a clear interface between applications and the file system, such as a POSIX interface.

There are several assumptions concerning GFS. One is related to the characteristic of the cloud computing hardware infrastructure (i.e., the high component failure 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. Another concerns the file size in GFS. GFS typically will hold a large number of huge files, each 100MB or larger, with files that are multiple GB in size quite common. Thus, Google has chosen its file data block size to be 64MB instead of the 4 KB in typical traditional file systems.

Fig. 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

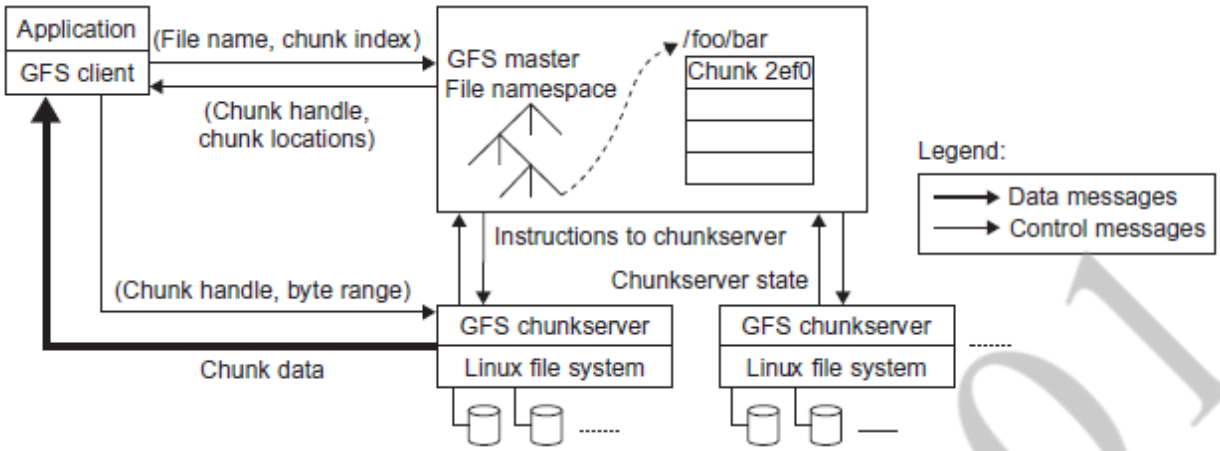


Fig: Architecture of Google File System (GFS)

The master has enough information to keep the whole cluster in a healthy state. With a single master, many complicated distributed algorithms can be avoided and the design of the system can be simplified.

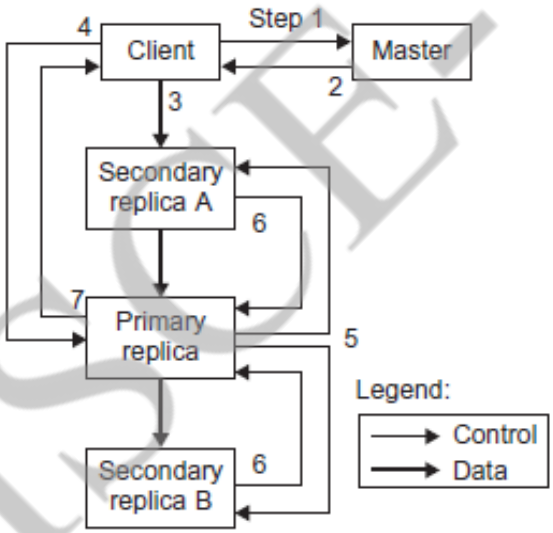


Fig: Data mutation sequence in GFS

Fig: shows the data mutation (write, append operations) in GFS. Data blocks must be created for all replicas. The goal is to minimize involvement of the master. 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. A client can do so in any order. 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. It will make a few attempts at steps 3 through 7 before falling back to a retry from the beginning of the write.

5.6. Open Stack

OpenStack was been introduced by Rackspace and NASA in July 2010. The project is building an open source community spanning technologists, developers, researchers, and industry to share resources and technologies with the goal of creating a massively scalable and secure cloud infrastructure. In the tradition of other open source projects, the software is open source and limited to just open source APIs such as Amazon.

Currently, OpenStack focuses on the development of two aspects of cloud computing to address compute and storage aspects with the *OpenStack Compute* and *OpenStack Storage solutions*. "OpenStack Compute is the internal fabric of the cloud creating and managing large groups of virtual private servers" and "OpenStack Object Storage is software for creating redundant,

scalable object storage using clusters of commodity servers to store terabytes or even petabytes of data."

5.6.1 OpenStack Compute

As part of its computing support efforts, OpenStack is developing a cloud computing fabric controller, a component of an IaaS system, known as Nova. The architecture for Nova is built on the concepts of shared-nothing and messaging-based information exchange. Hence, most communication in Nova is facilitated by message queues. To prevent blocking components while waiting for a response from others, deferred objects are introduced. Such objects include callbacks that get triggered when a response is received. This is very similar to established concepts from parallel computing, such as "futures," which have been used in the grid community by projects such as the CoG Kit.

Fig. shows the main architecture of Open Stack Compute. In this architecture, the API Server receives HTTP requests from boto, converts the commands to and from the API format, and forwards the requests to the cloud controller.

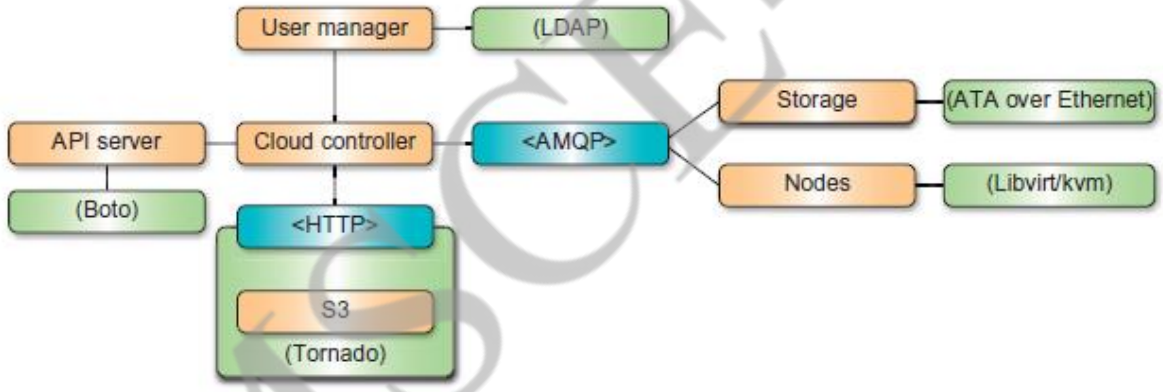


Fig: OpenStack Nova system architecture

The cloud controller maintains the global state of the system, ensures authorization while interacting with the User Manager via Lightweight Directory Access Protocol (LDAP), interacts with the S3 service, and manages nodes, as well as storage workers through a queue. Additionally, Nova integrates networking components to manage private networks, public IP addressing, virtual private network (VPN) connectivity, and firewall rules. It includes the following types:

• NetworkController manages address and virtual LAN (VLAN) allocations

• RoutingNode governs the NAT (network address translation) conversion of public IPs to private IPs, and enforces firewall rules

• AddressingNode runs Dynamic Host Configuration Protocol (DHCP) services for private networks

• TunnelingNode provides VPN connectivity

The network state (managed in the distributed object store) consists of the following:

• VLAN assignment to a project

• Private subnet assignment to a security group in a VLAN

• Private IP assignments to running instances

• Public IP allocations to a project

• Public IP associations to a private IP/running instance

5.6.2 OpenStack Storage

The OpenStack storage solution is built around a number of interacting components and concepts, including a proxy server, a ring, an object server, a container server, an account server, replication, updaters, and auditors. The role of the proxy server is to enable lookups to the accounts, containers, or objects in OpenStack storage rings and route the requests. Thus, any object is streamed to or from an object server directly through the proxy server to or from the user.

A ring represents a mapping between the names of entities stored on disk and their physical locations. Separate rings for accounts, containers, and objects exist. A ring includes the concept of using zones, devices, partitions, and replicas. Hence, it allows the system to deal with failures, and isolation of zones representing a drive, a server, a cabinet, a switch, or even a data center. Weights can be used to balance the distribution of partitions on drives across the cluster, allowing users to support heterogeneous storage resources.

Objects are stored as binary files with metadata stored in the file's extended attributes. This requires that the underlying file system is built around object servers, which is often not the case for standard Linux installations. To list objects, a container server can be utilized. Listing of containers is handled by the account server.

5.7. Federation in the Cloud

Federation in the Cloud One challenge in creating and managing a globally decentralized cloud computing environment is maintaining consistent connectivity between untrusted components while remaining fault-tolerant. A key opportunity ecosystem by connecting multiple cloud computing providers using a common standard.

A notable research project being conducted by Microsoft, called the Geneva Framework, focuses on issues involved in cloud federation. Geneva has been described as a claims-based access platform and is said to help simplify access to applications and other systems. The concept allows for multiple providers to interact seamlessly with others, and it enables developers to incorporate various authentication models that will work with any corporate identity system, including Active Directory, LDAPv3-based directories, application-specific databases, and new user-centric identity models such as LiveID, OpenID, and InfoCard systems. It also supports Microsoft's Card-Space and Novell's Digital Me.

The remainder of this section focuses on federation in the cloud through use of the Internet Engineering Task Force (IETF) standard Extensible Messaging and Presence Protocol (XMPP) and interdomain federation using the Jabber Extensible Communications Platform (Jabber XCP), because this protocol is currently used by a wide range of existing services offered by providers as diverse as Google Talk, Live Journal, Earthlink, Facebook, ooVoo, Meebo, Twitter, the U.S. Marines Corps, the Defense Information Systems Agency (DISA), the U.S. Joint Forces Command (USJFCOM), and the National Weather Service. We also look at federation with non-XMPP technologies such as the Session Initiation Protocol (SIP), which is the foundation of popular enterprise messaging systems such as IBM's Lotus Sametime and Microsoft's Live Communications Server (LCS) and Office Communications Server (OCS).

Jabber XCP is a highly scalable, extensible, available, and device-agnostic presence solution built on XMPP and supports multiple protocols such as Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) and Instant Messaging and Presence Service (IMPS). Jabber XCP is a highly programmable platform, which makes it ideal for adding presence and messaging to existing applications or services and for building next-generation, presence-based solutions.

Over the last few years there has been a controversy brewing in web services architectures. Cloud services are being talked up as a fundamental shift in web architecture that promises to move us from interconnected silos to a collaborative network of services whose sum is greater than its parts. The problem is that the protocols powering current cloud services, SOAP (Simple Object Access Protocol) and a few other assorted HTTP-based protocols, are all one-way information exchanges. Therefore cloud services aren't real-time, won't scale, and often can't clear the firewall. Many believe that those barriers can be overcome by XMPP (also called Jabber) as the protocol that will fuel the Software-as-a-Service (SaaS) models of tomorrow. Google, Apple, AOL, IBM, Livejournal, and Jive have all incorporated this protocol into their cloud-based solutions in the last few years.

Since the beginning of the Internet era, if you wanted to synchronize services between two servers, the most common solution was to have the client "ping" the host at regular intervals, which is known as polling. Polling is how most of us check our email. We ping our email server every few minutes to see if we have new mail. It's also how nearly all web services application programming interfaces (APIs) work.

XMPP's profile has been steadily gaining since its inception as the protocol behind the open source instant messenger (IM) server jabberd in 1998.

XMPP's advantages include:

- It is decentralized, meaning anyone may set up an XMPP server.
- It is based on open standards.
- It is mature—multiple implementations of clients and servers exist.
- Robust security is supported via Simple Authentication and Security
- Layer (SASL) and Transport Layer Security (TLS).
- It is flexible and designed to be extended.
- XMPP is a good fit for cloud computing because it allows for easy two way communication; it eliminates the need for polling; it has rich publish subscribe (pub-sub) functionality built in; it is XML-based and easily extensible, perfect for both new IM features and custom cloud services; it is efficient and has been proven to scale to millions of concurrent users on a single service (such as Google's GTalk); and it also has a built-in worldwide federation model.

Of course, XMPP is not the only pub-sub enabler getting a lot of interest from web application developers. An Amazon EC2-backed server can run Jetty and Cometd from Dojo. Unlike XMPP, Comet is based on HTTP and in conjunction with the Bayeux Protocol, uses JSON to exchange data.

Given the current market penetration and extensive use of XMPP and XCP for federation in the cloud and that it is the dominant open protocol in that space, we will focus on its use in our discussion of federation.

The ability to exchange data used for presence, messages, voice, video, files, notifications, etc., with people, devices, and applications gain more power when they can be shared across organizations and with other service providers. Federation differs from peering, which requires a prior agreement between parties before a server-to-server (S2S) link can be established. In the past, peering was more common among traditional telecommunications providers (because of the high cost of transferring voice traffic). In the brave new Internet world, federation has become a de facto standard for most email systems because they are federated dynamically through Domain Name System (DNS) settings and server configurations.

5.7.1. Four Levels of Federation

Technically speaking, federation is the ability for two XMPP servers in different domains to exchange XML stanzas. According to the XEP-0238:

XMPP Protocol Flows for Inter-Domain Federation, there are at least four basic types of federation

5.7.1.1. 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. With the release of the open source jabberd 1.2 server in October 2000, which included support for the Server Dialback protocol (fully supported in Jabber XCP), permissive federation met its demise on the XMPP network.

5.7.1.2. 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.

To make this work, 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.

5.7.1.3. Encrypted federation

In this mode, a 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.

If this is the case, both parties proceed to weakly verify identity using Server Dialback. XEP-0220 defines the Server Dialback protocol, 3.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 when 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.

Although Server Dialback does not provide strong authentication or trusted federation, and although it is subject to DNS poisoning attacks, it has effectively prevented most instances of address spoofing on the XMPP network since its release in 2000. 4.This results in an encrypted connection with weak identity verification.

5.7.1.4. Trusted federation

Here, 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.

In trusted federation, 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.

5.7.2. Federated Services and Applications

S2S federation is a good start toward building a real-time communications cloud. Clouds typically consist of all the users, devices, services, and applications connected to the network.

In order to fully utilize 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 (Extensible Messaging and Presence Protocol).It communicates with XMPP (Federated server) to provide better performance.

XMPP is an open XML technology for realtime communication which powers wide range of applications including instant messaging, presence and collaboration.

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.

In response to a service discovery query, 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.

Usually, these lists are comprised of other entities the users are interested in or interact with regularly.

Most XMPP deployments include custom directories so that internal users of those services can easily find what they are looking for.

5.7.3. The 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.

The power of a federated, presence-enabled communications infrastructure is that it enables software developers and service providers to build and deploy such applications without asking permission from a large, centralized communications operator.

The process of server-to-server federation for the purpose of interdomain communication has played a large role in the success of XMPP, which relies on a small set of simple but powerful mechanisms for domain checking and security to generate verified, encrypted, and trusted connections between any two deployed servers.

These mechanisms have provided a stable, secure foundation for growth of the XMPP network and similar realtime technologies.