CLOUD COMPUTING TECHNOLOGY

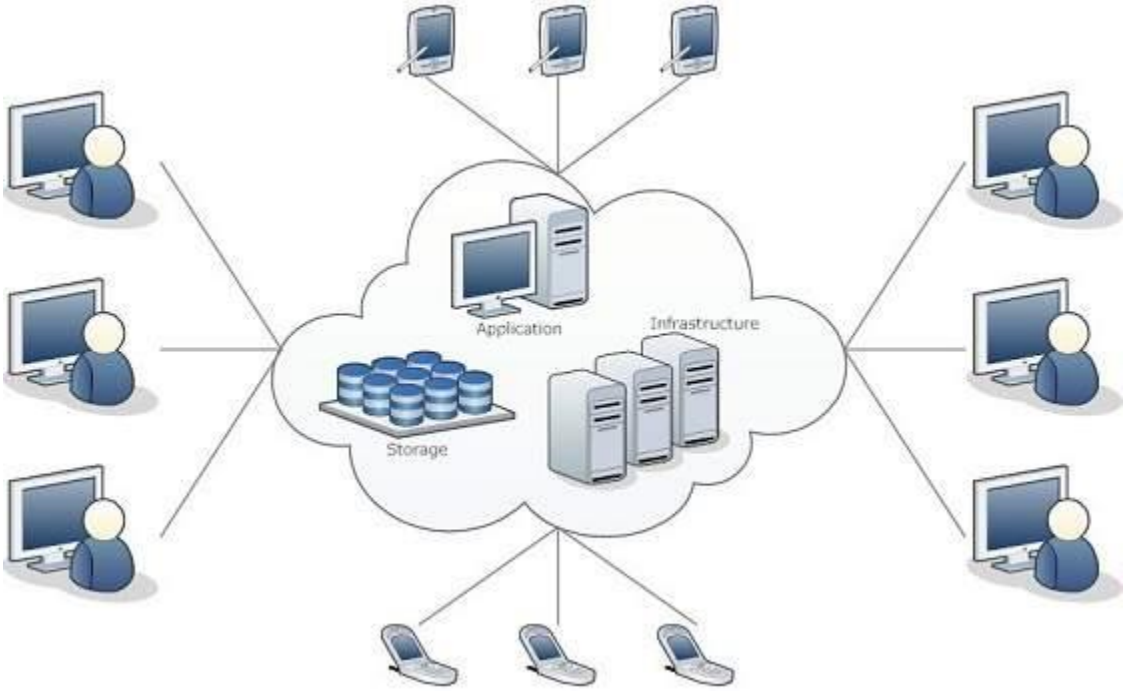
What is Cloud?

The term **Cloud** refers to a **Network** or **Internet.** In other words, we can say that Cloud is something, which is present at remote location. Cloud can provide services over public and private networks, i.e., WAN, LAN or VPN.

Applications such as e-mail, web conferencing, customer relationship management (CRM) execute on cloud.

What is Cloud Computing?

Cloud Computing refers to **manipulating, configuring,** and **accessing** the hardware and software resources remotely. It offers online data storage, infrastructure, and application.



Cloud computing offers **platform independency,** as the software is not required to be installed locally on the PC. Hence, the Cloud Computing is making our business applications **mobile** and **collaborative.**

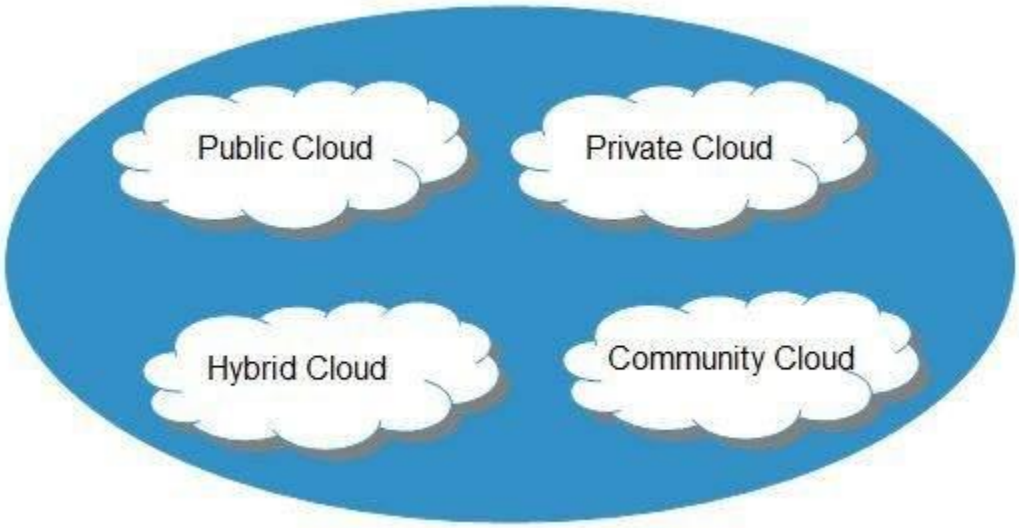
Basic Concepts

There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users. Following are the working models for cloud computing:

- Deployment Models
- Service Models

Deployment Models

Deployment models define the type of access to the cloud, i.e., how the cloud is located? Cloud can have any of the four types of access: Public, Private, Hybrid, and Community.



Public Cloud

The **public cloud** allows systems and services to be easily accessible to the general public. Public cloud may be less secure because of its openness.

Private Cloud

The **private cloud** allows systems and services to be accessible within an organization. It is more secured because of its private nature.

Community Cloud

The **community cloud** allows systems and services to be accessible by a group of organizations.

Hybrid Cloud

The **hybrid cloud** is a mixture of public and private cloud, in which the critical activities are performed using private cloud while the non-critical activities are performed using public cloud.

Service Models

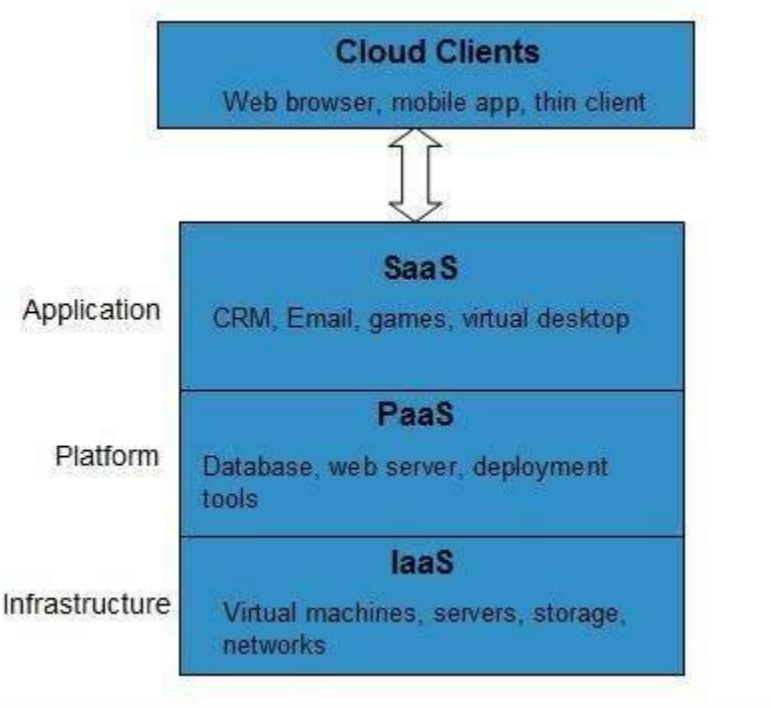
Cloud computing is based on service models. These are categorized into three basic service models which are -

- Infrastructure-as–a-Service (IaaS)

- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Anything-as-a-Service (XaaS) is yet another service model, which includes Network-as-a-Service, Business-as-a-Service, Identity-as-a-Service, Database-as-a-Service or Strategy-as-a-Service.

The **Infrastructure-as-a-Service (IaaS)** is the most basic level of service. Each of the service models inherit the security and management mechanism from the underlying model, as shown in the following diagram:



Infrastructure-as-a-Service (IaaS)

IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc.

Platform-as-a-Service (PaaS)

PaaS provides the runtime environment for applications, development and deployment tools, etc.

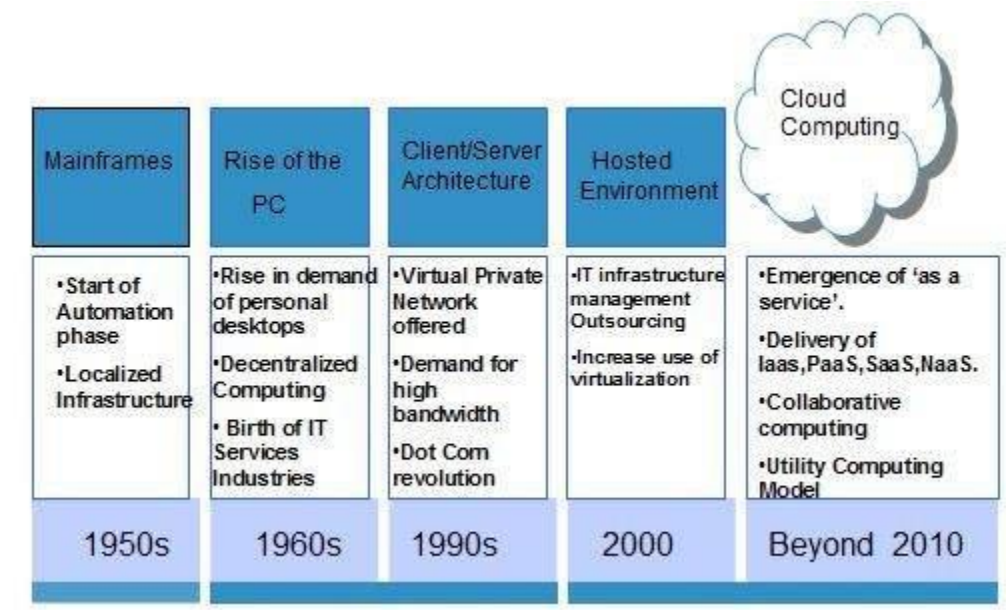
Software-as-a-Service (SaaS)

SaaS model allows to use software applications as a service to end-users.

History of Cloud Computing

The concept of **Cloud Computing** came into existence in the year 1950 with implementation of mainframe computers, accessible via **thin/static clients.** Since then, cloud computing has been

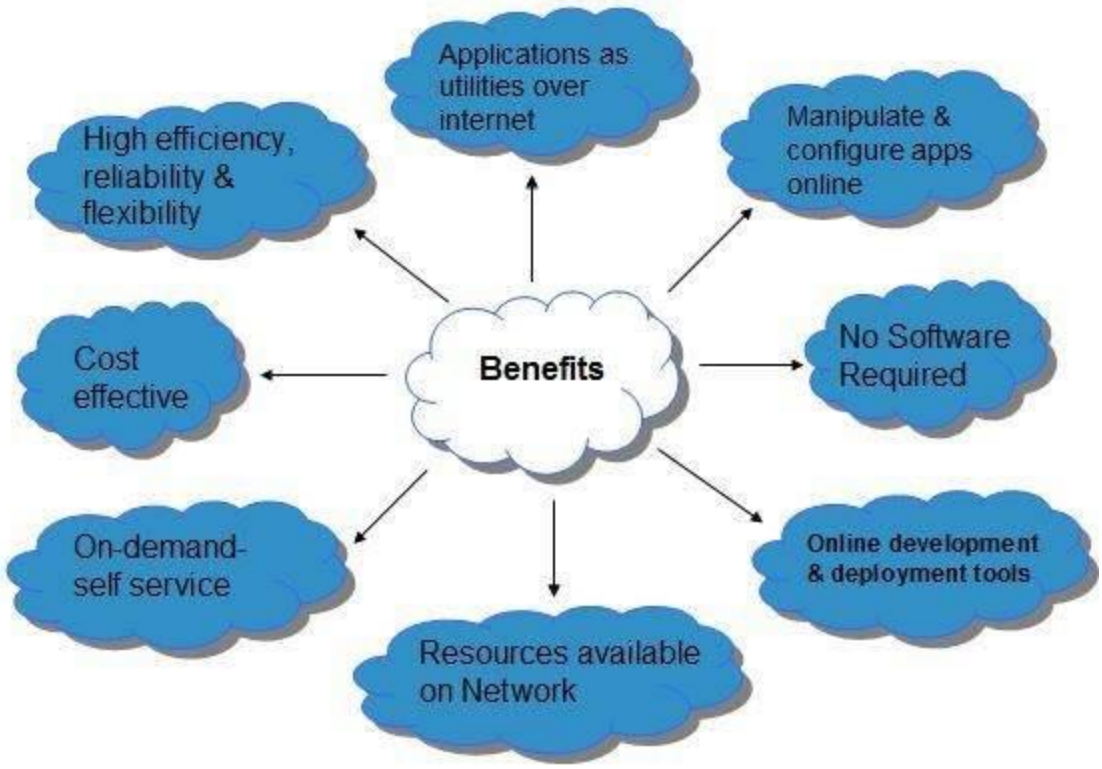
evolved from static clients to dynamic ones and from software to services. The following diagram explains the evolution of cloud computing:



Benefits

Cloud Computing has numerous advantages. Some of them are listed below -

- One can access applications as utilities, over the Internet.
- One can manipulate and configure the applications online at any time.
- It does not require to install a software to access or manipulate cloud application.
- Cloud Computing offers online development and deployment tools, programming runtime environment through **PaaS model.**
- Cloud resources are available over the network in a manner that provide platform independent access to any type of clients.
- Cloud Computing offers **on-demand self-service.** The resources can be used without interaction with cloud service provider.
- Cloud Computing is highly cost effective because it operates at high efficiency with optimum utilization. It just requires an Internet connection
- Cloud Computing offers load balancing that makes it more reliable.



Risks related to Cloud Computing

Although cloud Computing is a promising innovation with various benefits in the world of computing, it comes with risks. Some of them are discussed below:

Security and Privacy

It is the biggest concern about cloud computing. Since data management and infrastructure management in cloud is provided by third-party, it is always a risk to handover the sensitive information to cloud service providers.

Although the cloud computing vendors ensure highly secured password protected accounts, any sign of security breach may result in loss of customers and businesses.

Lock In

It is very difficult for the customers to switch from one **Cloud Service Provider (CSP)** to another. It results in dependency on a particular CSP for service.

Isolation Failure

This risk involves the failure of isolation mechanism that separates storage, memory, and routing between the different tenants.

Management Interface Compromise

In case of public cloud provider, the customer management interfaces are accessible through the Internet.

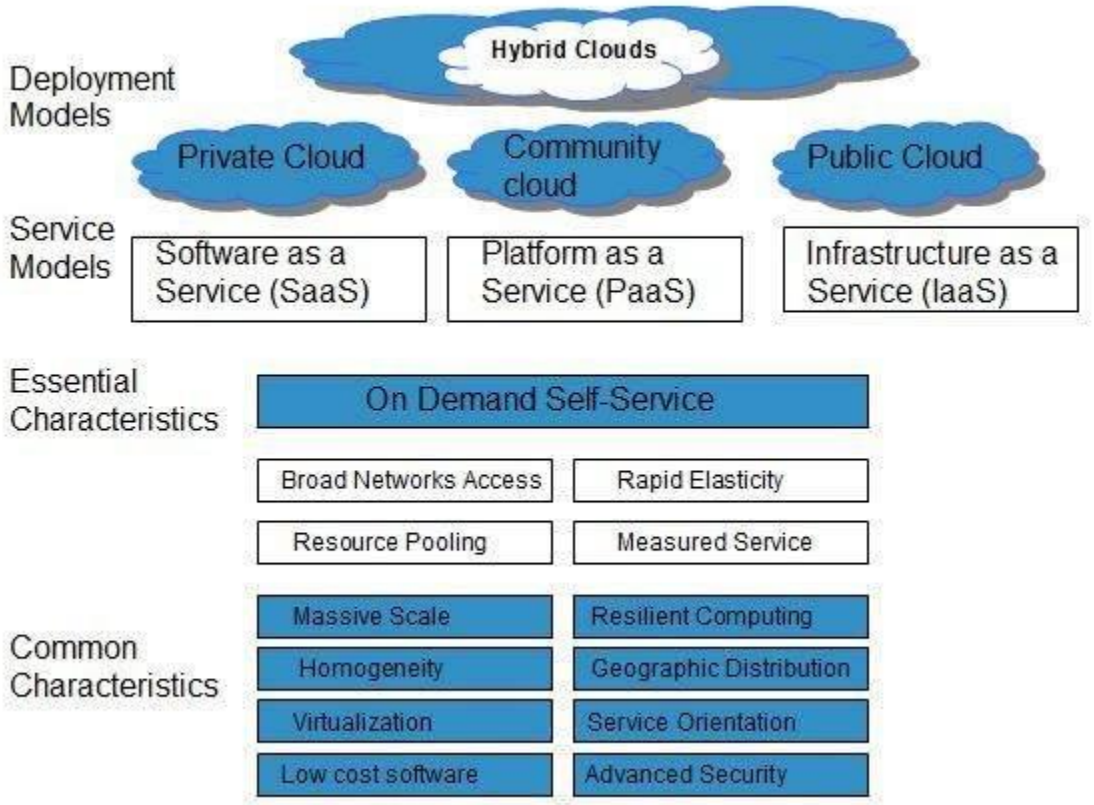
Insecure or Incomplete Data Deletion

It is possible that the data requested for deletion may not get deleted. It happens because either of the following reasons

- Extra copies of data are stored but are not available at the time of deletion
- Disk that stores data of multiple tenants is destroyed.

Characteristics of Cloud Computing

There are four key characteristics of cloud computing. They are shown in the following diagram:



On Demand Self Service

Cloud Computing allows the users to use web services and resources on demand. One can logon to a website at any time and use them.

Broad Network Access

Since cloud computing is completely web based, it can be accessed from anywhere and at any time.

Resource Pooling

Cloud computing allows multiple tenants to share a pool of resources. One can share single physical instance of hardware, database and basic infrastructure.

Rapid Elasticity

It is very easy to scale the resources vertically or horizontally at any time. Scaling of resources means the ability of resources to deal with increasing or decreasing demand.

The resources being used by customers at any given point of time are automatically monitored.

Measured Service

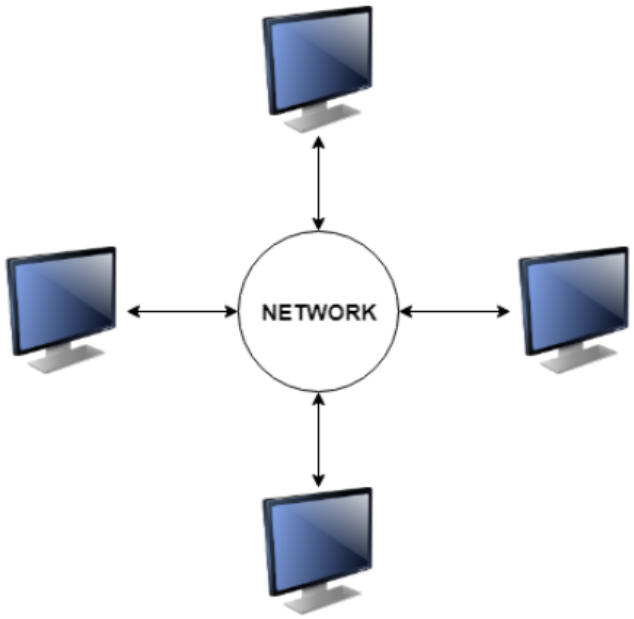
In this service cloud provider controls and monitors all the aspects of cloud service. Resource optimization, billing, and capacity planning etc. depend on it.

Unit I

DISTRIBUTED SYSTEM

A distributed system contains multiple nodes that are physically separate but linked together using the network. All the nodes in this system communicate with each other and handle processes in tandem. Each of these nodes contains a small part of the distributed operating system software.

A diagram to better explain the distributed system is −



DISTRIBUTED OPERATING SYSTEM

Types of Distributed Systems

The nodes in the distributed systems can be arranged in the form of client/server systems or peer to peer systems. Details about these are as follows −

Client/Server Systems

In client server systems, the client requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network and so they are a part of distributed systems.

Peer to Peer Systems

The peer to peer systems contains nodes that are equal participants in data sharing. All the tasks are equally divided between all the nodes. The nodes interact with each other as required as share resources. This is done with the help of a network.

Advantages of Distributed Systems

Some advantages of Distributed Systems are as follows −

- All the nodes in the distributed system are connected to each other. So nodes can easily share data with other nodes.
- More nodes can easily be added to the distributed system i.e. it can be scaled as required.
- Failure of one node does not lead to the failure of the entire distributed system. Other nodes can still communicate with each other.
- Resources like printers can be shared with multiple nodes rather than being restricted to just one.

Disadvantages of Distributed Systems

Some disadvantages of Distributed Systems are as follows −

- It is difficult to provide adequate security in distributed systems because the nodes as well as the connections need to be secured.
- Some messages and data can be lost in the network while moving from one node to another.
- The database connected to the distributed systems is quite complicated and difficult to handle as compared to a single user system.
- Overloading may occur in the network if all the nodes of the distributed system try to send data at once.

Key Characteristics of Distributed Systems

A *distributed system* is a system in which components are located on different *networked computers*, which can *communicate* and *coordinate* their actions by passing messages to one another. The components interact with one another in order to achieve a common goal.

Key characteristics of distributed systems are

- ✓ **Resource Sharing**

Resource sharing means that the existing *resources* in a distributed system can be accessed or remotely accessed across multiple computers in the system. Computers in distributed systems shares resources like *hardware* (disks and printers), *software* (files, windows and data objects) and *data*. Hardware resources are shared for reductions in cost and convenience. Data is shared for consistency and exchange of information.

Resources are managed by a software module known as a resource manager. Every resource has its own management policies and methods.

- ✓ **Heterogeneity**

In distributed systems components can have variety and differences in Networks, Computer hardware, Operating systems, Programming languages and implementations by different developers.

- ✓ **Openness**

Openness is concerned with extensions and improvements of distributed systems. The distributed system must be open in terms of *Hardware* and *Softwares*. In order to make a distributed system open,

1. A detailed and well-defined interface of components must be published.

2. Should standardize the interfaces of components

3. The new component must be easily integrated with existing components

- ✓ **Concurrency**

Concurrency is a property of a system representing the fact that multiple activities are executed at the same time. The concurrent execution of activities takes place in different components running on multiple machines as part of a distributed system. In addition, these activities may perform

some kind of interactions among them. Concurrency *reduces the latency* and *increases the throughput* of the distributed system.

- ✓ **Scalability**

Scalability is mainly concerned about how the distributed system handles the *growth* as the number of users for the system increases. Mostly we scale the distributed system by adding more computers in the network. Components should not need to be changed when we scale the system. Components should be designed in such a way that it is scalable.

- ✓ **Fault Tolerance**

In a distributed system hardware, software, network anything can fail. The system must be designed in such a way that it is available all the time even after something has failed.

- ✓ **Transparency**

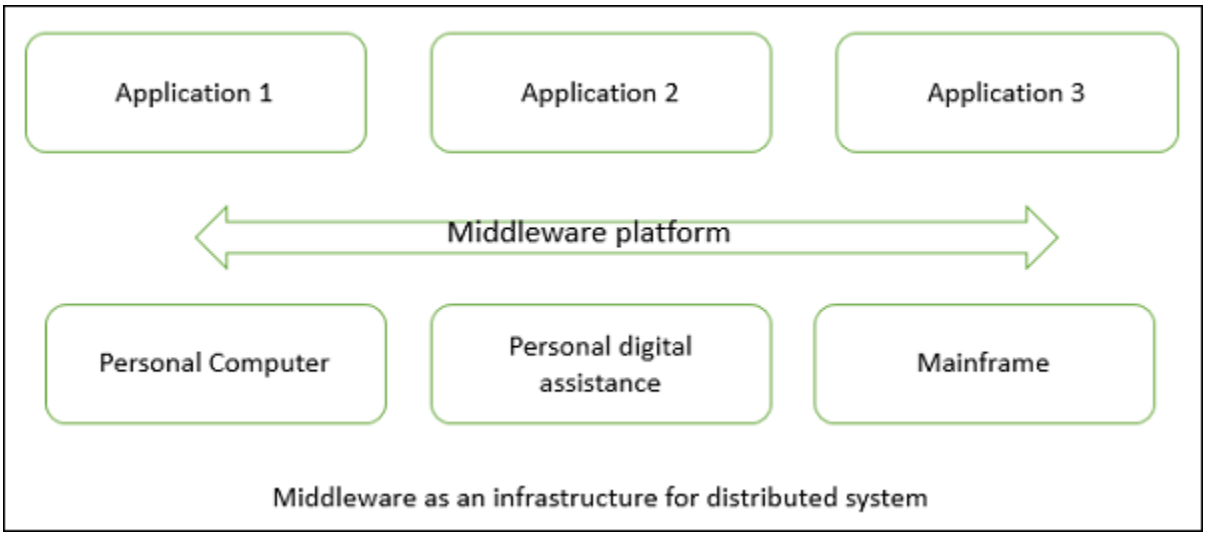
Distributed systems should be perceived by users and application programmers as a whole rather than as a collection of cooperating components. Transparency can be of various types like access, location, concurrency, replication, etc.

DISTRIBUTED ARCHITECTURE MODEL:

In distributed architecture, components are presented on different platforms and several components can cooperate with one another over a communication network in order to achieve a specific objective or goal.

- In this architecture, information processing is not confined to a single machine rather it is distributed over several independent computers.
- A distributed system can be demonstrated by the client-server architecture which forms the base for multi-tier architectures; alternatives are the broker architecture such as CORBA, and the Service-Oriented Architecture (SOA).
- There are several technology frameworks to support distributed architectures, including .NET, J2EE, CORBA, .NET Web services, AXIS Java Web services, and Globus Grid services.
- Middleware is an infrastructure that appropriately supports the development and execution of distributed applications. It provides a buffer between the applications and the network.
- It sits in the middle of system and manages or supports the different components of a distributed system. Examples are transaction processing monitors, data convertors and communication controllers etc.

Middleware as an infrastructure for distributed system



Middleware as an infrastructure for distributed system

The basis of a distributed architecture is its transparency, reliability, and availability.

The following table lists the different forms of transparency in a distributed system −

Sr.No.	Transparency & Description
1	**Access** Hides the way in which resources are accessed and the differences in data

	platform.
2	**Location** Hides where resources are located.
3	**Technology** Hides different technologies such as programming language and OS from user.
4	**Migration / Relocation** Hide resources that may be moved to another location which are in use.
5	**Replication** Hide resources that may be copied at several location.
6	**Concurrency** Hide resources that may be shared with other users.
7	**Failure** Hides failure and recovery of resources from user.
8	**Persistence** Hides whether a resource (software) is in memory or disk.

Advantages

- **Resource sharing** − Sharing of hardware and software resources.
- **Openness** − Flexibility of using hardware and software of different vendors.
- **Concurrency** − Concurrent processing to enhance performance.
- **Scalability** − Increased throughput by adding new resources.
- **Fault tolerance** − The ability to continue in operation after a fault has occurred.

Disadvantages

- **Complexity** − They are more complex than centralized systems.
- **Security** − More susceptible to external attack.
- **Manageability** − More effort required for system management.
- **Unpredictability** − Unpredictable responses depending on the system organization and network load.

Centralized System vs. Distributed System

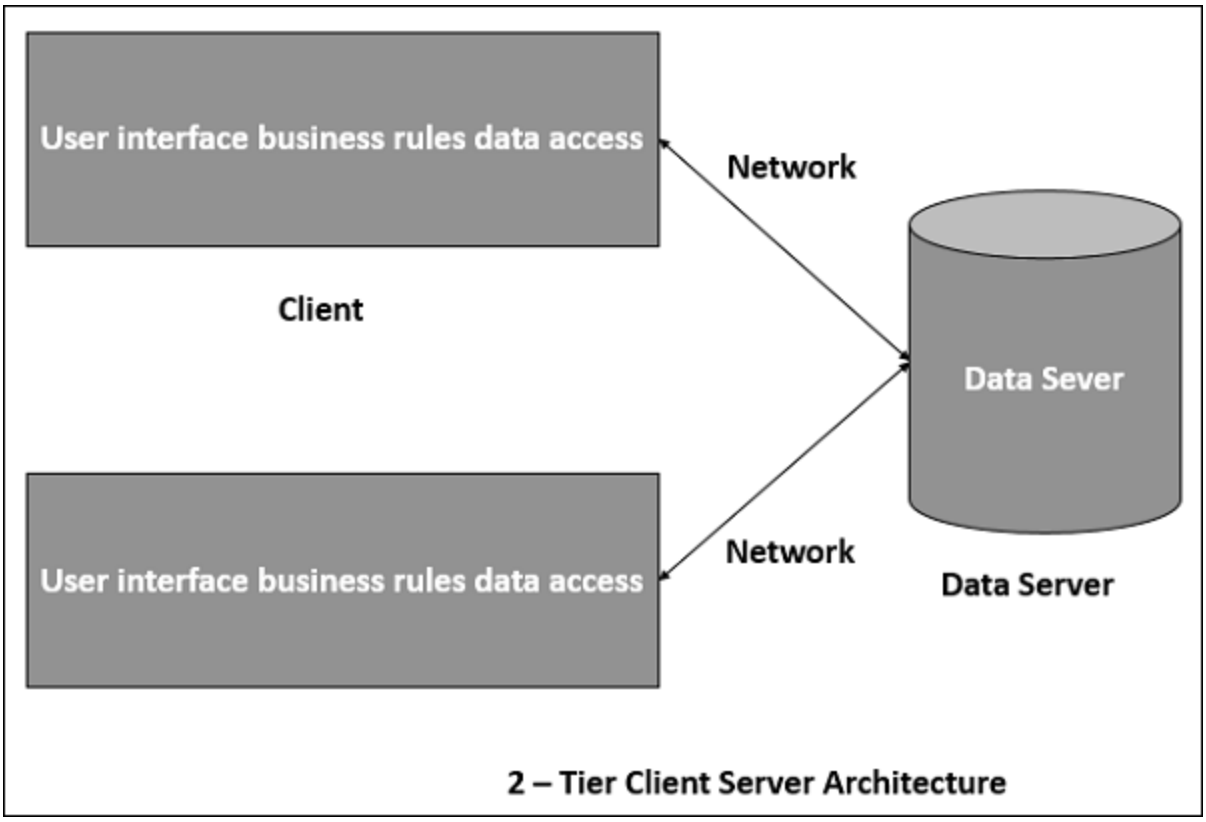
Criteria	Centralized system	Distributed System
Economics	Low	High
Availability	Low	High
Complexity	Low	High
Consistency	Simple	High
Scalability	Poor	Good
Technology	Homogeneous	Heterogeneous
Security	High	Low

Client-Server Architecture

The client-server architecture is the most common distributed system architecture which decomposes the system into two major subsystems or logical processes −

- **Client** − This is the first process that issues a request to the second process i.e. the server.
- **Server** − This is the second process that receives the request, carries it out, and sends a reply to the client.

In this architecture, the application is modelled as a set of services that are provided by servers and a set of clients that use these services. The servers need not know about clients, but the clients must know the identity of servers, and the mapping of processors to processes is not necessarily 1 : 1



Client-server Architecture can be classified into two models based on the functionality of the client −

Thin-client model

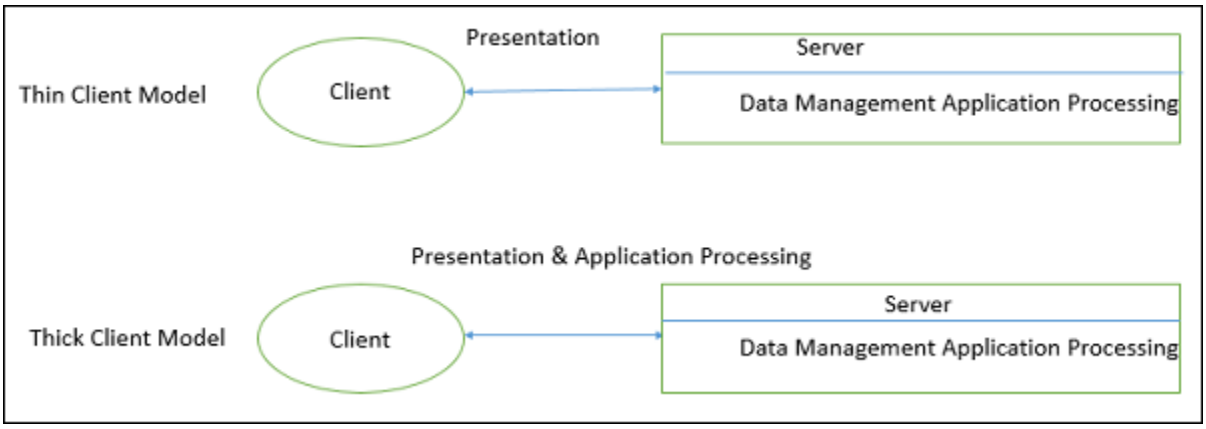
In thin-client model, all the application processing and data management is carried by the server. The client is simply responsible for running the presentation software.

- Used when legacy systems are migrated to client server architectures in which legacy system acts as a server in its own right with a graphical interface implemented on a client
- A major disadvantage is that it places a heavy processing load on both the server and the network.

Thick/Fat-client model

In thick-client model, the server is only in charge for data management. The software on the client implements the application logic and the interactions with the system user.

- Most appropriate for new C/S systems where the capabilities of the client system are known in advance
- More complex than a thin client model especially for management. New versions of the application have to be installed on all clients.



Advantages

- Separation of responsibilities such as user interface presentation and business logic processing.
- Reusability of server components and potential for concurrency
- Simplifies the design and the development of distributed applications
- It makes it easy to migrate or integrate existing applications into a distributed environment.
- It also makes effective use of resources when a large number of clients are accessing a high-performance server.

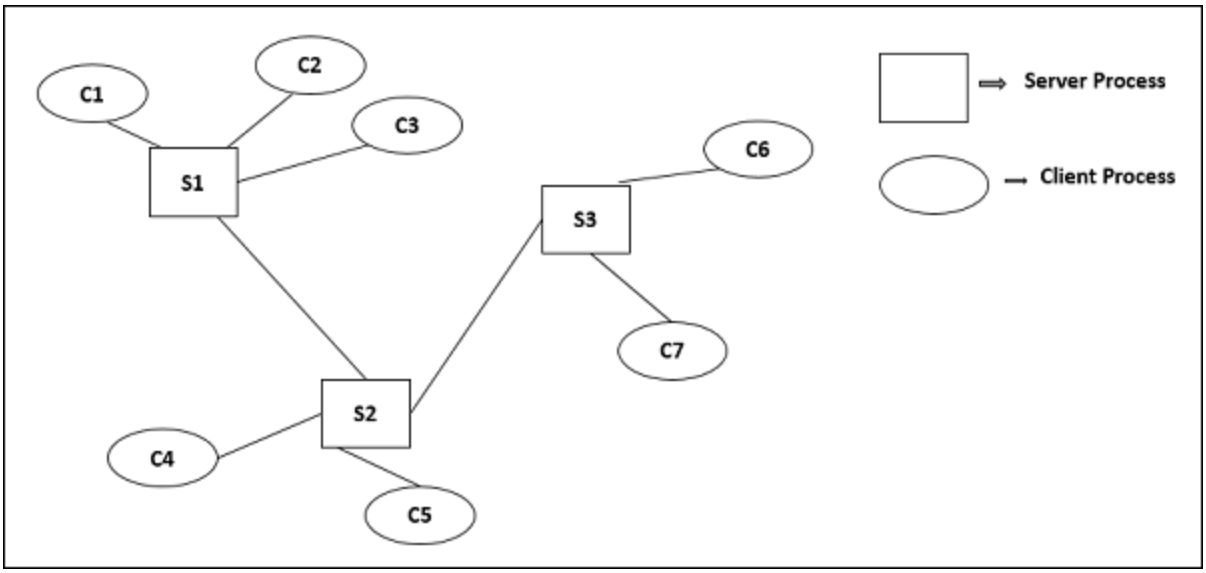
Disadvantages

- Lack of heterogeneous infrastructure to deal with the requirement changes.
- Security complications.
- Limited server availability and reliability.
- Limited testability and scalability.
- Fat clients with presentation and business logic together.

Multi-Tier Architecture (n-tier Architecture)

Multi-tier architecture is a client–server architecture in which the functions such as presentation, application processing, and data management are physically separated. By separating an application into tiers, developers obtain the option of changing or adding a specific layer, instead of reworking the entire application. It provides a model

by which developers can create flexible and reusable applications.



The most general use of multi-tier architecture is the three-tier architecture. A three-tier architecture is typically composed of a presentation tier, an application tier, and a data storage tier and may execute on a separate processor.

Presentation Tier

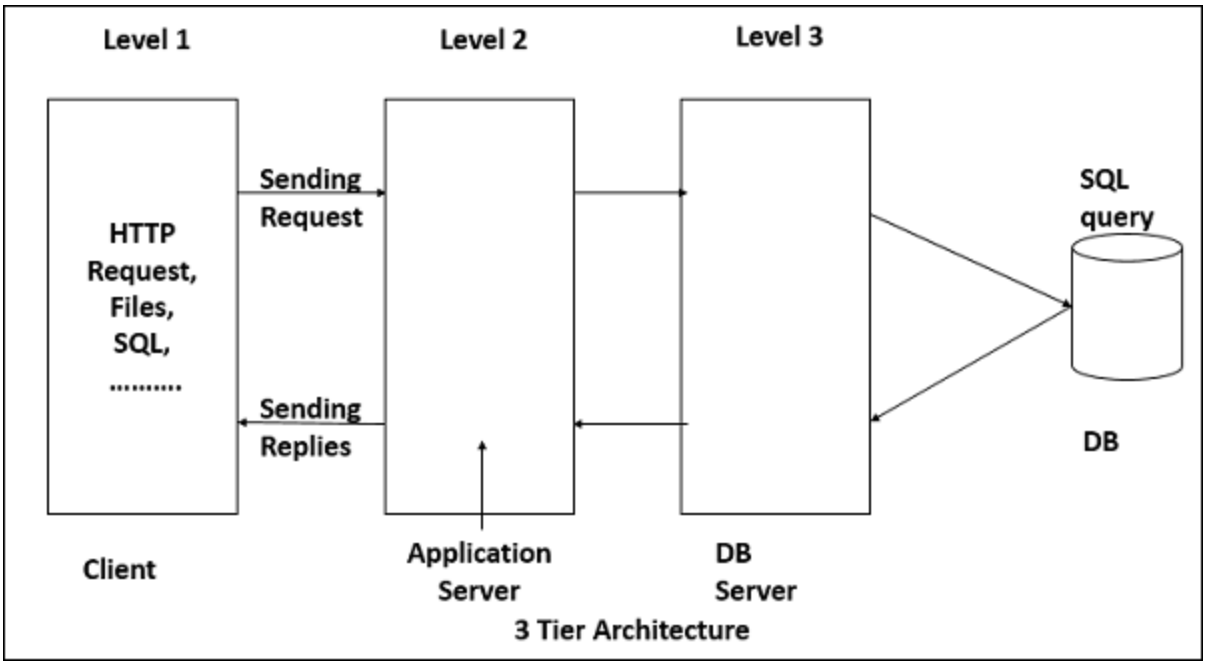
Presentation layer is the topmost level of the application by which users can access directly such as webpage or Operating System GUI (Graphical User interface). The primary function of this layer is to translate the tasks and results to something that user can understand. It communicates with other tiers so that it places the results to the browser/client tier and all other tiers in the network.

Application Tier (Business Logic, Logic Tier, or Middle Tier)

Application tier coordinates the application, processes the commands, makes logical decisions, evaluation, and performs calculations. It controls an application's functionality by performing detailed processing. It also moves and processes data between the two surrounding layers.

Data Tier

In this layer, information is stored and retrieved from the database or file system. The information is then passed back for processing and then back to the user. It includes the data persistence mechanisms (database servers, file shares, etc.) and provides API (Application Programming Interface) to the application tier which provides methods of managing the stored data.



Advantages

- Better performance than a thin-client approach and is simpler to manage than a thick-client approach.
- Enhances the reusability and scalability − as demands increase, extra servers can be added.
- Provides multi-threading support and also reduces network traffic.
- Provides maintainability and flexibility

Disadvantages

- Unsatisfactory Testability due to lack of testing tools.
- More critical server reliability and availability.

Broker Architectural Style

Broker Architectural Style is a middleware architecture used in distributed computing to coordinate and enable the communication between registered servers and clients. Here, object communication takes place through a middleware system called an object request broker (software bus).

- Client and the server do not interact with each other directly. Client and server have a direct connection to its proxy which communicates with the mediator-broker.
- A server provides services by registering and publishing their interfaces with the broker and clients can request the services from the broker statically or dynamically by look-up.

- CORBA (Common Object Request Broker Architecture) is a good implementation example of the broker architecture.

Components of Broker Architectural Style

The components of broker architectural style are discussed through following heads −

Broker

Broker is responsible for coordinating communication, such as forwarding and dispatching the results and exceptions. It can be either an invocation-oriented service, a document or message - oriented broker to which clients send a message.

- It is responsible for brokering the service requests, locating a proper server, transmitting requests, and sending responses back to clients.
- It retains the servers' registration information including their functionality and services as well as location information.
- It provides APIs for clients to request, servers to respond, registering or unregistering server components, transferring messages, and locating servers.

Stub

Stubs are generated at the static compilation time and then deployed to the client side which is used as a proxy for the client. Client-side proxy acts as a mediator between the client and the broker and provides additional transparency between them and the client; a remote object appears like a local one.

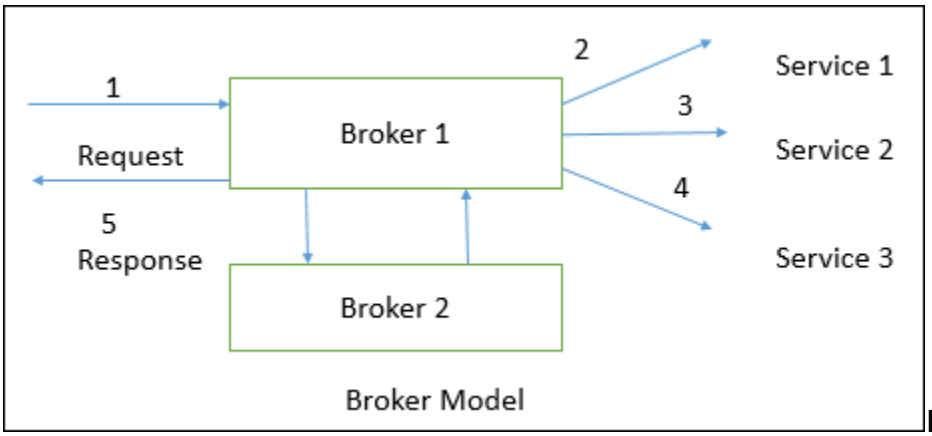
The proxy hides the IPC (inter-process communication) at protocol level and performs marshaling of parameter values and un-marshaling of results from the server.

Skeleton

Skeleton is generated by the service interface compilation and then deployed to the server side, which is used as a proxy for the server. Server-side proxy encapsulates low-level system-specific networking functions and provides high-level APIs to mediate between the server and the broker.

It receives the requests, unpacks the requests, unmarshals the method arguments, calls the suitable service, and also marshals the result before sending it back to the client.

Bridge



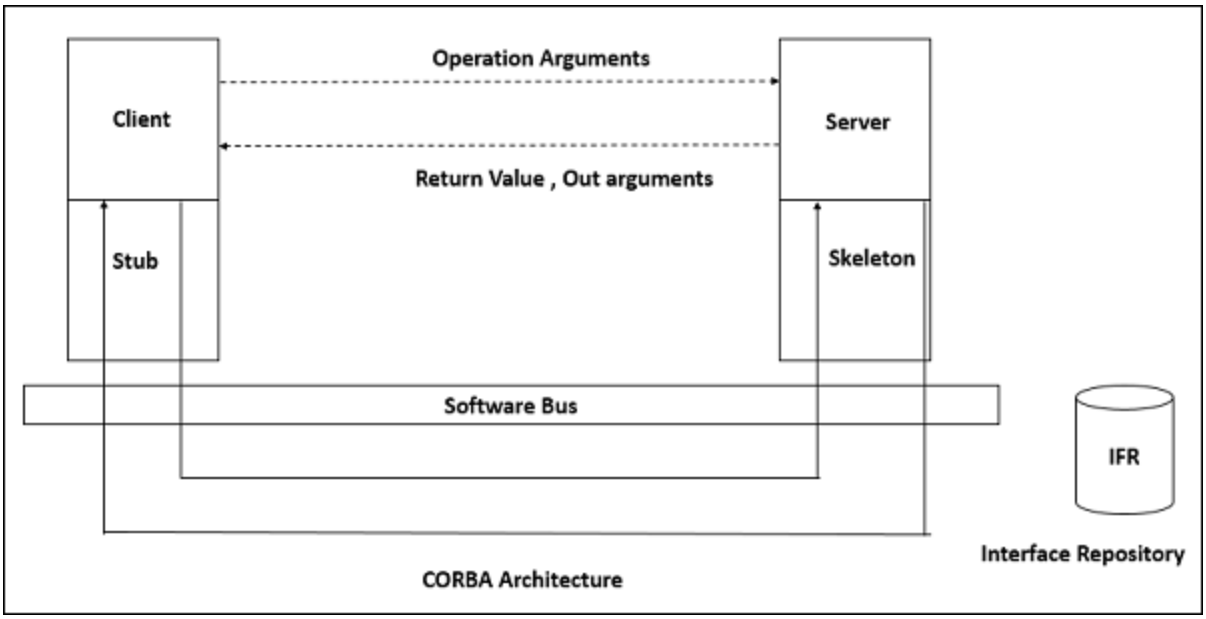
Bridge

A bridge can connect two different networks based on different communication protocols. It mediates different brokers including DCOM, .NET remote, and Java CORBA brokers.

Bridges are optional component, which hides the implementation details when two brokers interoperate and take requests and parameters in one format and translate them to another format.

Broker implementation in CORBA

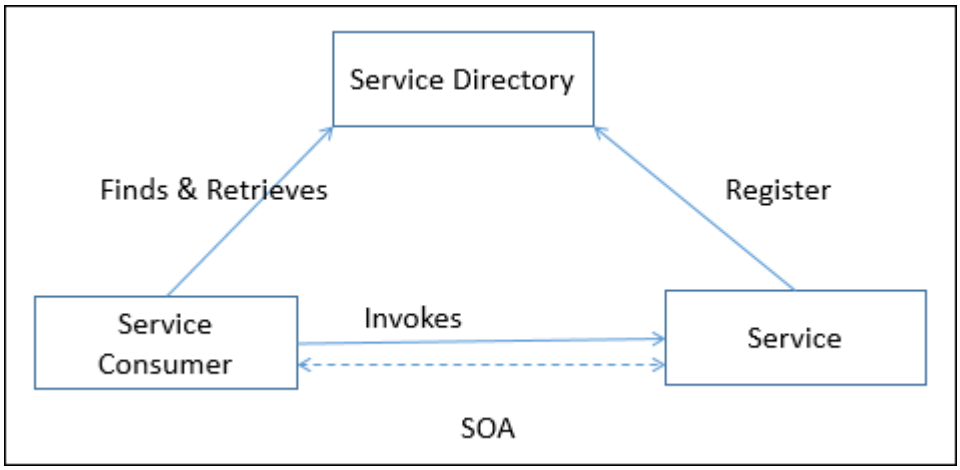
CORBA is an international standard for an Object Request Broker – a middleware to manage communications among distributed objects defined by OMG (object management group).



Service-Oriented Architecture (SOA)

A service is a component of business functionality that is well-defined, self-contained, independent, published, and available to be used via a standard programming interface. The connections between services are conducted by common and universal message-oriented protocols such as the SOAP Web service protocol, which can deliver requests and responses between services loosely.

Service-oriented architecture is a client/server design which support business-driven IT approach in which an application consists of software services and software service consumers (also known as clients or service requesters).



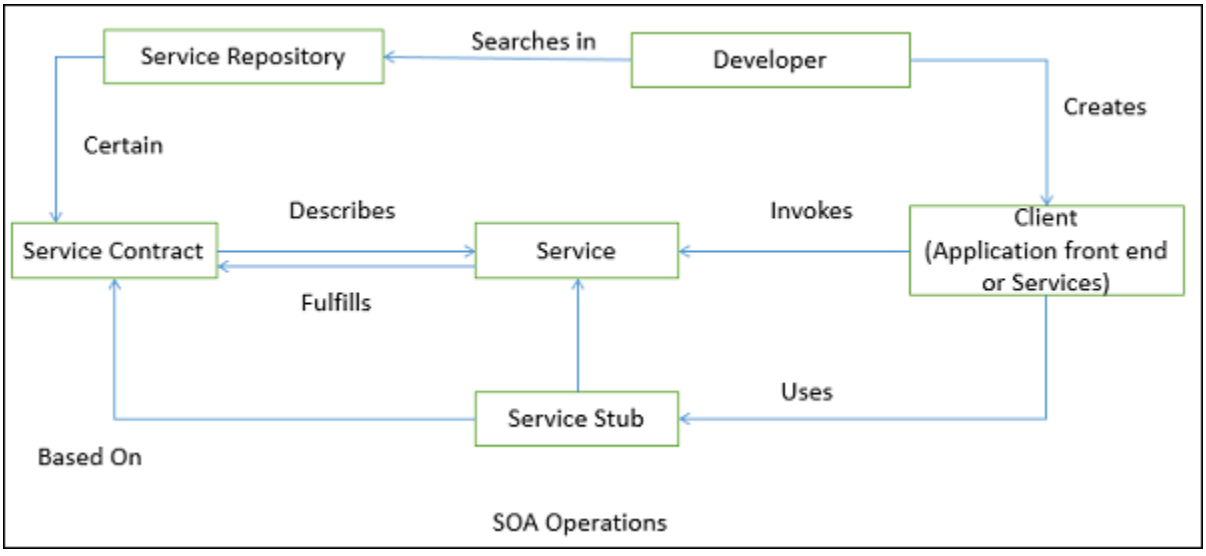
Features of SOA

A service-oriented architecture provides the following features −

- **Distributed Deployment** − Expose enterprise data and business logic as loosely, coupled, discoverable, structured, standard-based, coarse-grained, stateless units of functionality called services.
- **Composability** − Assemble new processes from existing services that are exposed at a desired granularity through well defined, published, and standard complaint interfaces.
- **Interoperability** − Share capabilities and reuse shared services across a network irrespective of underlying protocols or implementation technology.
- **Reusability** − Choose a service provider and access to existing resources exposed as services.

SOA Operation

The following figure illustrates how does SOA operate–



Advantages

- Loose coupling of service–orientation provides great flexibility for enterprises to make use of all available service recourses irrespective of platform and technology restrictions.
- Each service component is independent from other services due to the stateless service feature.
- The implementation of a service will not affect the application of the service as long as the exposed interface is not changed.
- A client or any service can access other services regardless of their platform, technology, vendors, or language implementations.
- Reusability of assets and services since clients of a service only need to know its public interfaces, service composition.
- SOA based business application development are much more efficient in terms of time and cost.
- Enhances the scalability and provide standard connection between systems.
- Efficient and effective usage of 'Business Services'.
- Integration becomes much easier and improved intrinsic interoperability.
- Abstract complexity for developers and energize business processes closer to end users.

RMI (Remote Method Invocation)

The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.

Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.

A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),

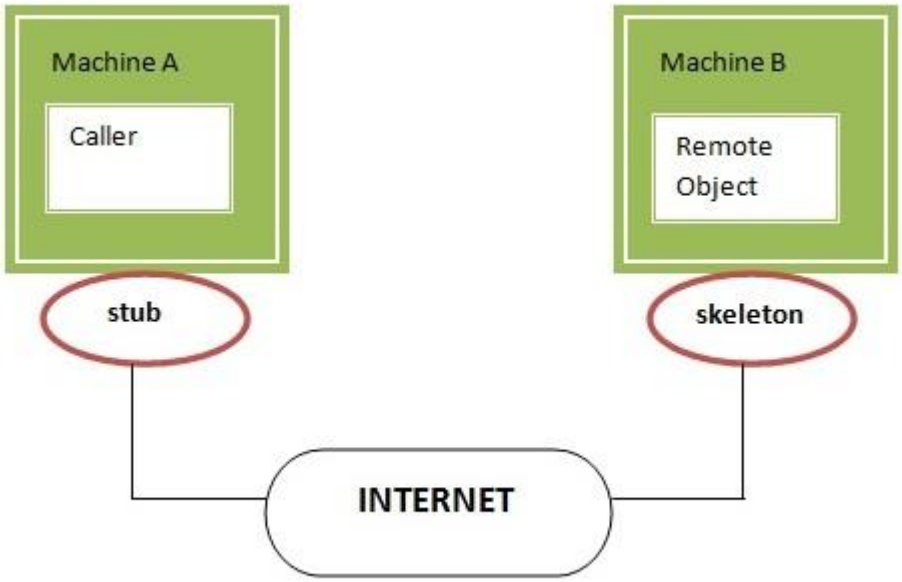
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

In the Java 2 SDK, an stub protocol was introduced that eliminates the need for



skeletons.

Understanding requirements for the distributed applications

If any application performs these tasks, it can be distributed application.

1. The application need to locate the remote method
2. It need to provide the communication with the remote objects, and

3. The application need to load the class definitions for the objects.

The RMI application have all these features, so it is called the distributed application.

What is RPC?

Remote Procedure Call (RPC) is an interprocess communication technique. The Full form of RPC is Remote Procedure Call. It is used for client-server applications. RPC mechanisms are used when a computer program causes a procedure or subroutine to execute in a different address space, which is coded as a normal procedure call without the programmer specifically coding the details for the remote interaction.

This procedure call also manages low-level transport protocol, such as User Datagram Protocol, Transmission Control Protocol/Internet Protocol etc. It is used for carrying the message data between programs.

Types of RPC

Three types of RPC are:

- Callback RPC
- Broadcast RPC
- Batch-mode RPC

Callback RPC

This type of RPC enables a P2P paradigm between participating processes. It helps a process to be both client and server services.

Functions of Callback RPC:

- Remotely processed interactive application problems
- Offers server with clients handle
- Callback makes the client process wait
- Manage callback deadlocks
- It facilitates a peer-to-Peer paradigm among participating processes.

Broadcast RPC

Broadcast RPC is a client's request, that is broadcast on the network, processed by all servers which have the method for processing that request.

Functions of Broadcast RPC:

- Allows you to specify that the client's request message has to be broadcasted.
- You can declare broadcast ports.
- It helps to reduce the load on the physical network

Batch-mode RPC

Batch-mode RPC helps to queue, separate RPC requests, in a transmission buffer, on the client-side, and then send them on a network in one batch to the server.

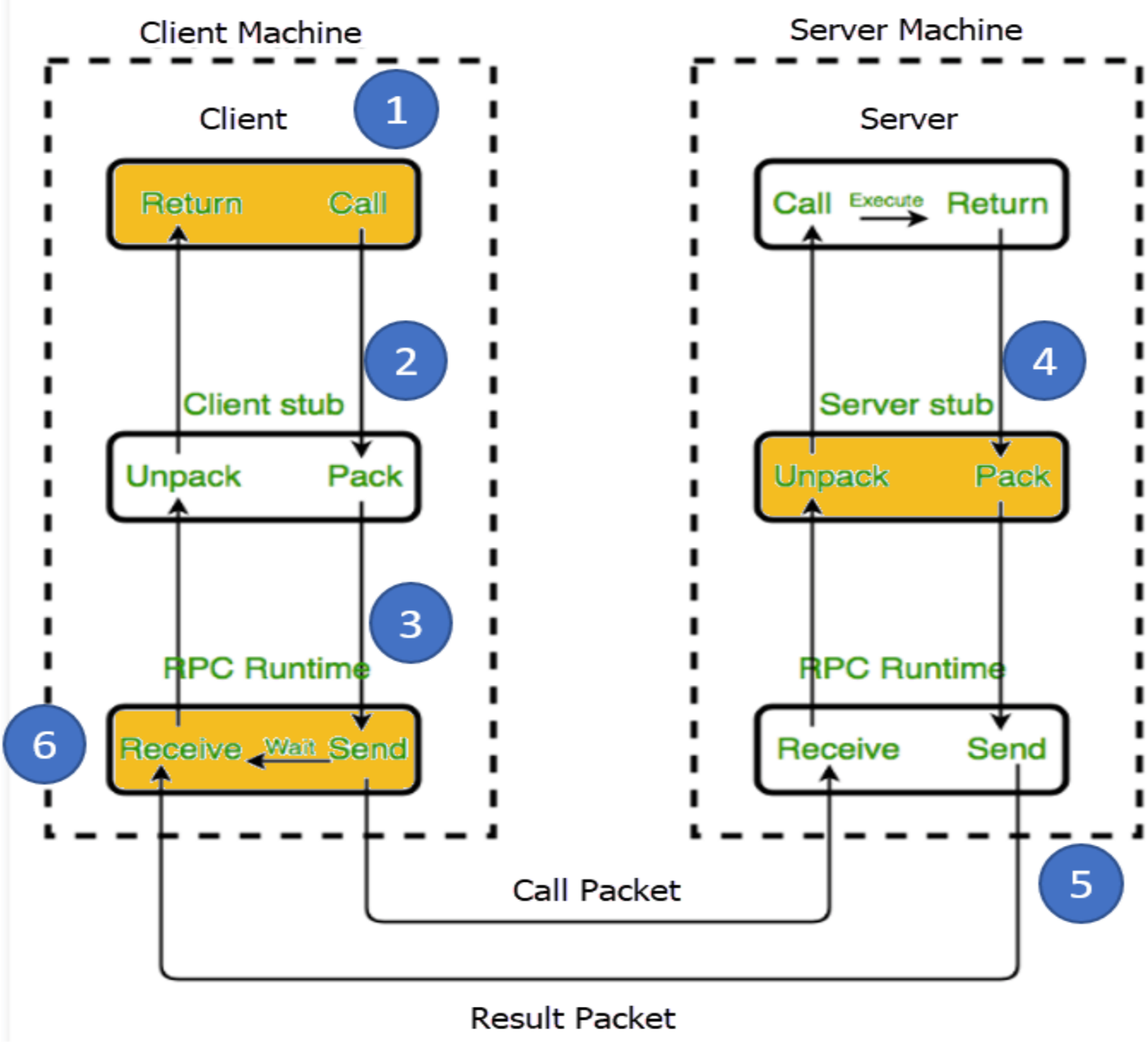
Functions of Batch-mode RPC:

- It minimizes overhead involved in sending a request as it sends them over the network in one batch to the server.
- This type of RPC protocol is only efficient for the application that needs lower call rates.
- It needs a reliable transmission protocol.

RPC Architecture

RPC architecture has mainly five components of the program:

1. **Client**
2. **Client Stub**
3. **RPC Runtime**
4. **Server Stub**
5. **Server**



RPC Architecture

How RPC Works?

Following steps take place during the RPC process:

Step 1) The client, the client stub, and one instance of RPC run time execute on the client machine.

Step 2) A client starts a client stub process by passing parameters in the usual way. The client stub stores within the client's own address space. It also asks the local RPC Runtime to send back to the server stub.

Step 3) In this stage, RPC accessed by the user by making regular Local Procedural Cal. RPC Runtime manages the transmission of messages between the network across client and server. It also performs the job of retransmission, acknowledgment, routing, and encryption.

Step 4) After completing the server procedure, it returns to the server stub, which packs (marshalls) the return values into a message. The server stub then sends a message back to the transport layer.

Step 5) In this step, the transport layer sends back the result message to the client transport layer, which returns back a message to the client stub.

Step 6) In this stage, the client stub demarshalls (unpack) the return parameters, in the resulting packet, and the execution process returns to the caller.

Characteristics of RPC

Here are the essential characteristics of RPC:

- The called procedure is in another process, which is likely to reside in another machine.
- The processes do not share address space.
- Parameters are passed only by values.
- RPC executes within the environment of the server process.
- It doesn't offer access to the calling procedure's environment.

Features of RPC

Here are the important features of RPC:

- Simple call syntax
- Offers known semantics
- Provide a well-defined interface
- It can communicate between processes on the same or different machines

Advantages of RPC

Here are Pros/benefits of RPC:

- RPC method helps clients to communicate with servers by the conventional use of procedure calls in high-level languages.

- RPC method is modeled on the local procedure call, but the called procedure is most likely to be executed in a different process and usually a different computer.
- RPC supports process and thread-oriented models.
- RPC makes the internal message passing mechanism hidden from the user.
- The effort needs to re-write and re-develop the code is minimum.
- Remote procedure calls can be used for the purpose of distributed and the local environment.
- It commits many of the protocol layers to improve performance.
- RPC provides abstraction. For example, the message-passing nature of network communication remains hidden from the user.
- RPC allows the usage of the applications in a distributed environment that is not only in the local environment.
- With RPC code, re-writing and re-developing effort is minimized.
- Process-oriented and thread-oriented models support by RPC.

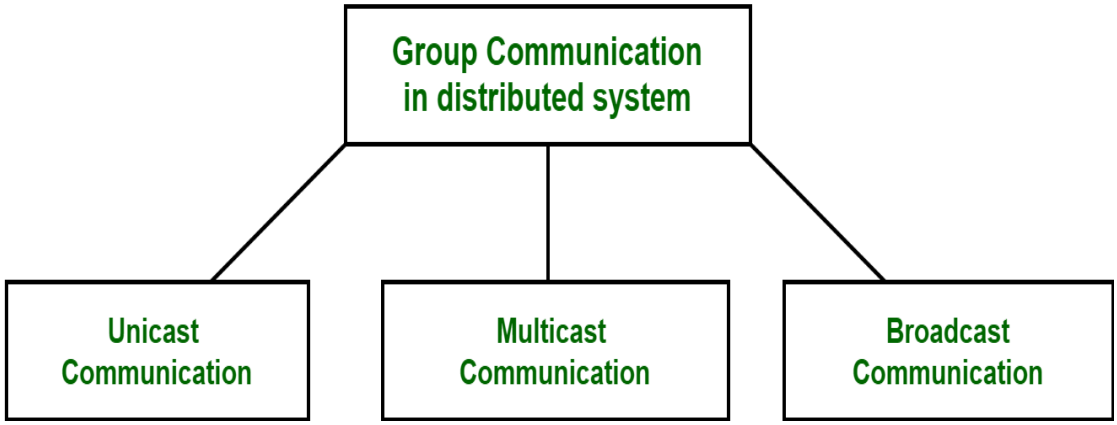
Disadvantages of RPC

Here are the cons/drawbacks of using RPC:

- Remote Procedure Call Passes Parameters by values only and pointer values, which is not allowed.
- Remote procedure calling (and return) time (i.e., overheads) can be significantly lower than that for a local procedure.
- This mechanism is highly vulnerable to failure as it involves a communication system, another machine, and another process.
- RPC concept can be implemented in different ways, which is can't standard.
- Not offers any flexibility in RPC for hardware architecture as It is mostly interaction-based.
- The cost of the process is increased because of a remote procedure call.

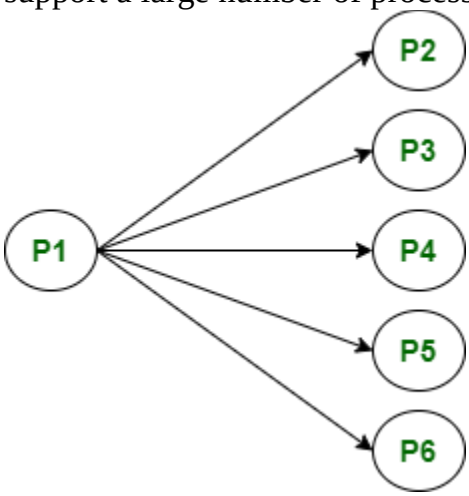
GROUP COMMUNICATION

Communication between two processes in a distributed system is required to exchange various data, such as code or a file, between the processes. When one source process tries to communicate with multiple processes at once, it is called **Group Communication**. A group is a collection of interconnected processes with abstraction. This abstraction is to hide the message passing so that the communication looks like a normal procedure call. Group communication also helps the processes from different hosts to work together and perform operations in a synchronized manner, therefore increases the overall performance of the system.



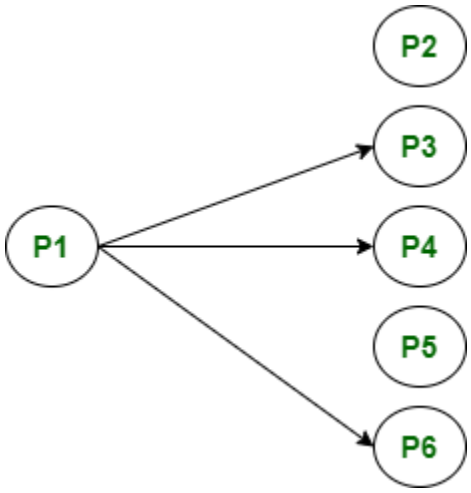
Types of Group Communication in a Distributed System :

- **Broadcast Communication :**
  When the host process tries to communicate with every process in a distributed system at same time. Broadcast communication comes in handy when a common stream of information is to be delivered to each and every process in most efficient manner possible. Since it does not require any processing whatsoever, communication is very fast in comparison to other modes of communication. However, it does not support a large number of processes and cannot treat a specific process individually.



A broadcast Communication: P1 process communicating with every process in the system

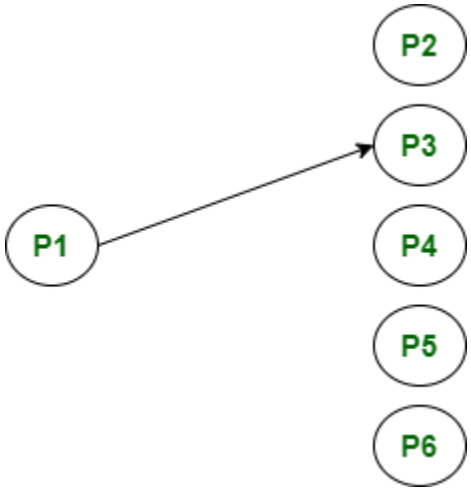
- **Multicast Communication :**
  When the host process tries to communicate with a designated group of processes in a distributed system at the same time. This technique is mainly used to find a way to address problem of a high workload on host system and redundant information from process in system. Multitasking can significantly decrease time taken for message handling.



A multicast Communication: P1 process communicating with only a group of the process in the system

Unicast Communication :

When the host process tries to communicate with a single process in a distributed system at the same time. Although, same information may be passed to multiple processes. This works best for two processes communicating as only it has to treat a specific process only. However, it leads to overheads as it has to find exact process and then exchange information/data.



Features of Group Communication are:

- Goals.
- Members.
- Interaction.
- Interdependence.
- Working.

PHYSICAL CLOCK SYNCHRONIZATION

Synchronization in distributed systems is achieved via clocks. **The physical clocks are used to adjust the time of nodes**. Each node in the system can share its local time with other nodes in the system. The time is set based on UTC (Universal Time Coordination).

Synchronization in Distributed Systems

Distributed system is a collection of computers connected via the high speed communication network. In the distributed system, the hardware and software components communicate and coordinate their actions by message passing. Each node in distributed systems can share their resources with other nodes. So, there is need of proper allocation of resources to preserve the state of resources and help coordinate between the several processes. To resolve such conflicts, synchronization is used. Synchronization in distributed systems is achieved via clocks.

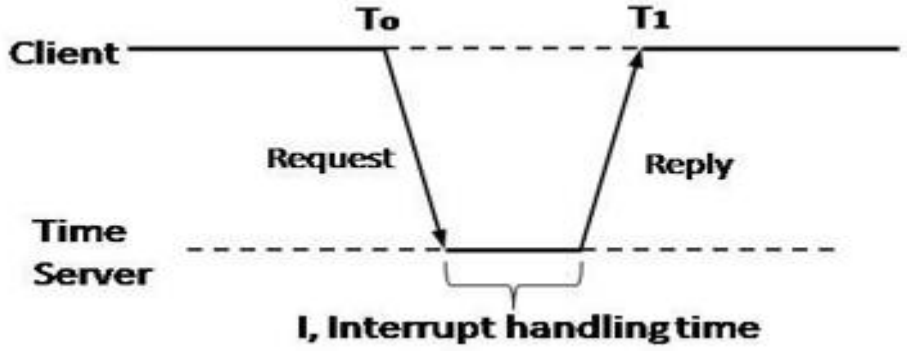
The physical clocks are used to adjust the time of nodes.Each node in the system can share its local time with other nodes in the system. The time is set based on UTC (Universal Time Coordination). UTC is used as a reference time clock for the nodes in the system.

The clock synchronization can be achieved by 2 ways: External and Internal Clock Synchronization.

1. **External clock synchronization** is the one in which an external reference clock is present. It is used as a reference and the nodes in the system can set and adjust their time accordingly.
2. **Internal clock synchronization** is the one in which each node shares its time with other nodes and all the nodes set and adjust their times accordingly.

There are 2 types of clock synchronization algorithms: Centralized and Distributed.

1. **Centralized** is the one in which a time server is used as a reference. The single time server propagates its time to the nodes and all the nodes adjust the time accordingly. It is dependent on single time server so if that node fails, the whole system will lose synchronization. Examples of centralized are- Berkeley Algorithm, Passive Time Server, Active Time Server etc.
2. **Distributed** is the one in which there is no centralized time server present. Instead the nodes adjust their time by using their local time and then, taking the average of the differences of time with other nodes. Distributed algorithms overcome the issue of centralized algorithms like the scalability and single point failure. Examples of Distributed algorithms are – Global Averaging Algorithm, Localized Averaging Algorithm, NTP (Network time protocol) etc.



Logical Clock in Distributed System

Logical Clocks refer to implementing a protocol on all machines within your distributed system, so that the machines are able to maintain consistent ordering of events within some virtual timespan. A logical clock is a mechanism for capturing chronological and causal relationships in a distributed system. Distributed systems may have no physically synchronous global clock, so a logical clock allows global ordering on events from different processes in such systems.

Example :

If we go outside then we have made a full plan that at which place we have to go first, second and so on. We don't go to second place at first and then the first place. We always maintain the procedure or an organization that is planned before. In a similar way, we should do the operations on our PCs one by one in an organized way.

Suppose, we have more than 10 PCs in a distributed system and every PC is doing it's own work but then how we make them work together. There comes a solution to this i.e. LOGICAL CLOCK.

Method-1:

To order events across process, try to sync clocks in one approach.

This means that if one PC has a time 2:00 pm then every PC should have the same time which is quite not possible. Not every clock can sync at one time. Then we can't follow this method.

Method-2:
Another approach is to assign Timestamps to events.
Taking the example into consideration, this means if we assign the first place as 1, second place as 2, third place as 3 and so on. Then we always know that the first place will always come first and then so on. Similarly, If we give each PC their individual number than it will be organized in a way that 1st PC will complete its process first and then second and so on.

BUT, Timestamps will only work as long as they obey causality.

What is causality ?
Causality is fully based on HAPPEN BEFORE RELATIONSHIP.

- Taking single PC only if 2 events A and B are occurring one by one then TS(A) < TS(B). If A has timestamp of 1, then B should have timestamp more than 1, then only happen before relationship occurs.
- Taking 2 PCs and event A in P1 (PC.1) and event B in P2 (PC.2) then also the condition will be TS(A) < TS(B). Taking example- suppose you are sending message to someone at 2:00:00 pm, and the other person is receiving it at 2:00:02 pm.Then it's obvious that TS(sender) < TS(receiver).

Properties Derived from Happen Before Relationship –

- **Transitive Relation –**
  If, TS(A) <TS(B) and TS(B) <TS(C), then TS(A) < TS(C)
- **Causally Ordered Relation –**
  a->b, this means that a is occurring before b and if there is any changes in a it will surely reflect on b.
- **Concurrent Event –**
  This means that not every process occurs one by one, some processes are made to happen simultaneously i.e., A || B.

Unit II

Basics of Cloud Computing

Features of Cloud Computing

Cloud computing is becoming popular day by day. Continuous business expansion and growth requires huge computational power and large-scale data storage systems. Cloud computing can help organizations expand and securely move data from physical locations to the 'cloud' that can be accessed anywhere.

Cloud computing has many features that make it one of the fastest growing industries at present. The flexibility offered by cloud services in the form of their growing set of tools and technologies has accelerated its deployment across industries. This blog will tell you about the essential features of cloud computing.

1. Resources Pooling

Resource pooling is one of the essential features of cloud computing. Resource pooling means that a cloud service provider can share resources among multiple clients, each providing a different set of services according to their needs. It is a multi-client strategy that can be applied to data storage, processing and bandwidth-delivered services. The administration process of allocating resources in real-time does not conflict with the client's experience.

2. On-Demand Self-Service

It is one of the important and essential features of cloud computing. This enables the client to continuously monitor server uptime, capabilities and allocated network storage. This is a fundamental feature of cloud computing, and a customer can also control the computing capabilities according to their needs.

3. Easy Maintenance

This is one of the best cloud features. Servers are easily maintained, and downtime is minimal or sometimes zero. Cloud computing powered resources often undergo several updates to optimize their capabilities and potential. Updates are more viable with devices and perform faster than previous versions.

4. Scalability And Rapid Elasticity

A key feature and advantage of cloud computing is its rapid scalability. This cloud feature enables cost-effective handling of workloads that require a large number of servers but only for a short period. Many customers have workloads that can be run very cost-effectively due to the rapid scalability of cloud computing.

5. Economical

This cloud feature helps in reducing the IT expenditure of the organizations. In cloud computing, clients need to pay the administration for the space used by them. There is no cover-up or additional charges that need to be paid. Administration is economical, and more often than not, some space is allocated for free.

6. Measured And Reporting Service

Reporting Services is one of the many cloud features that make it the best choice for organizations. The measurement and reporting service is helpful for both cloud providers and their customers. This enables both the provider and the customer to monitor and report which services have been used and for what purposes. It helps in monitoring billing and ensuring optimum utilization of resources.

7. Security

Data security is one of the best features of cloud computing. Cloud services make a copy of the stored data to prevent any kind of data loss. If one server loses data by any chance, the copied version is restored from the other server. This feature comes in handy when multiple users are working on a particular file in real-time, and one file suddenly gets corrupted.

8. Automation

Automation is an essential feature of cloud computing. The ability of cloud computing to automatically install, configure and maintain a cloud service is known as automation in cloud computing. In simple words, it is the process of

making the most of the technology and minimizing the manual effort. However, achieving automation in a cloud ecosystem is not that easy. This requires the installation and deployment of virtual machines, servers, and large storage. On successful deployment, these resources also require constant maintenance.

9. Resilience

Resilience in cloud computing means the ability of a service to quickly recover from any disruption. The resilience of a cloud is measured by how fast its servers, databases and network systems restart and recover from any loss or damage. Availability is another key feature of cloud computing. Since cloud services can be accessed remotely, there are no geographic restrictions or limits on the use of cloud resources.

10. Large Network Access

A big part of the cloud's characteristics is its ubiquity. The client can access cloud data or transfer data to the cloud from any location with a device and internet connection. These capabilities are available everywhere in the organization and are achieved with the help of internet. Cloud providers deliver that large network access by monitoring and guaranteeing measurements that reflect how clients access cloud resources and data: latency, access times, data throughput, and more.

Rapid Elasticity in Cloud Computing

Elasticity is a 'rename' of scalability, a known non-functional requirement in IT architecture for many years already. Scalability is the ability to add or remove capacity, mostly processing, memory, or both, from an IT environment.

Ability to dynamically scale the services provided directly to customers' need for space and other services. It is one of the five fundamental aspects of cloud computing.

It is usually done in two ways:

- **Horizontal Scalability**: Adding or removing nodes, servers, or instances to or from a pool, such as a cluster or a farm.
- **Vertical Scalability**: Adding or removing resources to an existing node, server, or instance to increase the capacity of a node, server, or instance.

Most implementations of scalability are implemented using the horizontal method, as it is the easiest to implement, especially in the current web-based world we live in. Vertical Scaling is less dynamic because this requires reboots of systems, sometimes adding physical components to servers.

A well-known example is adding a load balancer in front of a farm of web servers that distributes the requests.

Why call it Elasticity?

Traditional IT environments have scalability built into their architecture, but scaling up or down isn't done very often. It has to do with Scaling and the amount of time, effort, and cost.

Servers have to be purchased, operations need to be screwed into server racks, installed and configured, and then the test team needs to verify functioning, and only after that's done can you get the big There are. And you don't just buy a server for a few months - typically, it's three to five years. So it is a long-term investment that you make.

The latch is doing the same, but more like a rubber band. You 'stretch' the ability when you need it and 'release' it when you don't have it. And this is possible because of some of the other features of cloud computing, such as "resource pooling" and "on-demand self-service". Combining these features with advanced image management capabilities allows you to scale more efficiently.

What is on-demand computing?

On-demand computing (ODC) is a delivery model in which computing resources are made available to the user as needed. The resources may be maintained within the user's enterprise or made available by a cloud service provider. The term cloud computing is often used as a synonym for on-demand computing when the services are provided by a third party -- such as a cloud hosting organization.

The on-demand business computing model was developed to overcome the challenge of enterprises meeting fluctuating demands efficiently. Because an enterprise's demand for computing resources can be unpredictable at times, maintaining sufficient resources to meet peak requirements can be costly. And cutting costs by only maintaining minimal resources means there are likely

insufficient resources to meet peak loads. The on-demand model provides an enterprise with the ability to scale computing resources up or down whenever needed, with the click of a button.

The model is characterized by three attributes: scalability, pay-per-use and self-service. Whether the resource is an application program that helps team members collaborate or provides additional storage, the computing resources are elastic, metered and easy to obtain.

When an organization pairs with a third party to provide on-demand computing, it either subscribes to the service or uses a pay-per-use model. The third party then provides computing resources whenever needed, including when the organization is working on temporary projects, has expected or unexpected workloads or has long-term computing requirements. For example, a retail organization could use on-demand computing to scale up their online services, providing additional computing resources during a high-volume time, such as Black Friday.

On-demand computing normally provides computing resources such as storage capacity, or hardware and software applications. The service itself is provided with methods including virtualization, computer clusters and distributed computing.

How does cloud computing provide on-demand functionality?

Cloud computing is a general term for anything that involves delivering hosted services over the internet. These services are divided into different types of cloud computing resources and applications.

For example, on-demand computing often involves cloud computing methods, such as infrastructure as a service (IaaS), software as a service (SaaS), desktop as a service (DaaS), platform as a service (PaaS), managed hosting services, as well as cloud storage and backup services. These methods offer the following:

- **IaaS** provides virtualized computing resources over the internet.
- **SaaS** is a software distribution model where a cloud provider hosts applications and makes them available to users over the internet.

- **DaaS** is a form of cloud computing where a third party hosts the back end of a virtual desktop infrastructure
- **PaaS** is a model in which a third-party provider hosts customer applications on their infrastructure. Hardware and software tools are delivered to users over the internet.
- **Managed hosting services** are an IT provisioning and cloud server hosting model where a service provider leases dedicated servers and associated hardware to a single customer and manages those systems on the customer's behalf.
- **Cloud storage** is a service model where data is transmitted and stored securely on remote storage systems, where it is maintained, managed, backed up and made available to users over a network.
- **Cloud backup** is a strategy for sending a copy of a file or database to a secondary location for preservation in case of equipment failure.

These cloud-based services are typically made on-demand and in real time for users. Computing resources are delivered using a shared pool of servers, storage devices, networks and applications.

Cloud hosting providers may provide an enterprise-level control panel where they can quickly view and scale up or down their cloud services. An organization could use this to scale their storage space, speed, software applications, servers or networks.

Benefits of on-demand computing

On-demand computing offers the following benefits:

  - **Flexibility to meet fluctuating demands.** Users can quickly increase or decrease their computing resources as needed -- either short-term or long-term.
  - **Removes the need to purchase, maintain and upgrade hardware.** The cloud service organization managing the on-demand services handles resources such as servers and hardware, system updates and maintenance.
- **User friendly.** Many on-demand computing services in the cloud are user friendly enabling most users to easily acquire additional computing

resources without any help from their IT department. This can help to improve business agility.

- **Cut costs.** Saves money because organizations don't have to purchase hardware or software to meet peaks in demand. Organizations also don't have to worry about updating or maintaining those resources.

But organizations must also be concerned about the unauthorized use of added resources via on-demand computing, as shadow IT can pose security risks. For this reason, many IT departments perform periodic cloud audits to identify unauthorized use of on-demand applications and other rogue IT.

The future of on-demand computing

According to a report from Gartner, cloud-based platform services may increase to $109.6 billion in corporate spending in 2022 -- which is up from $86.9 billion in 2021. Likewise, Gartner predicts a 26% increase in spending on cloud-based platform services. Spending on IaaS is also forecasted by Gartner to show one of the highest growth rates at 31% in 2022 compared to other cloud categories.

Large vendors such as Amazon Web Services, HPE, IBM and Microsoft offer on-demand computing products. Microsoft, for example, provides Azure SaaS and AWS offers pay-as-you go pricing with its IaaS offerings. As more of these services become available, there is a greater chance that enterprises will look to on-demand computing as a way to facilitate the challenges of fluctuating computing resource needs.

Components of Cloud Computing

The basic components of cloud computing in a simple topology are divided into 3 (three) parts, namely clients, datacenter, and distributed servers. The three basic components have specific goals and roles in running cloud computing operations. The concept of the three components can be described as follows:

- **Clients** on cloud computing architecture are said to be the exact same things that are plain, old, everyday local area networks (LANs). They are, typically, the computers that just sit on your desk. But they might also be laptops, tablet computers, mobile phones, or PDAs - all big drivers for cloud computing because of their mobility. Clients are interacting with to manage their information on the cloud.

- **Datacenter** is collection of servers where the application to which you subscribe is housed. It could be a large room in the basement of your building full of servers on the other side of the world that you access via the Internet. A growing trend in the IT world is virtualizing servers. That is, software can be installed allowing multiple instances of virtual servers to be used. In this way, you can have half a dozen virtual servers running on one physical server.
- **Distributed** Servers is a server placement in a different location. But the servers don't have to be housed in the same location. Often, servers are in geographically disparate locations. But to you, the cloud subscribers, these servers act as if they're humming away right next to each other.

Another component of cloud computing is Cloud Applications cloud computing in terms of software architecture. So that the user does not need to install and run applications using a computer. Cloud Platform is a service in the form of a computing platform that contains hardware infrastructure and software. Usually have certain business applications and use services PaaS as its business application infrastructure. Cloud Storage involves processes delivering data storage as a service. Cloud Infrastructure is the delivery of computing infrastructure as a service.

Cloud Computing services have several components required, namely:

a. Cloud Clients, a computer or software specifically designed for the use of cloud computing based services.
Example :

- Mobile - Windows Mobile, Symbian
- Thin Client - Windows Terminal Service, CherryPal
- Thick Client - Internet Explorer, FireFox, Chrome

b. Cloud Services, products, services and solutions that are used and delivered real-time via internet media.
Example :

- Identity - OpenID, OAuth, etc.
- Integration - Amazon Simple Queue Service.
- Payments - PayPal, Google Checkout.
- Mapping - Google Maps, Yahoo! Maps.

c. Cloud Applications, applications that use Cloud Computing in software architecture so that users don't need to install but they can use the application using a computer.
Example :

- Peer-to-peer - BitTorrent, SETI, and others.
- Web Application - Facebook.

- SaaS - Google Apps, SalesForce.com, and others

d. Cloud Platform, a service in the form of a computing platform consisting of hardware and infrastructure software. This service is a service in the form of a computing platform which contains infrastructure hardware and software. Usually has an application certain businesses and use PaaS services as application infrastructure his business
Example :

- Web Application Frameworks - Python Django, Rubyon Rails, .NET
- Web Hosting
- Propietary - Force.com

e. Cloud Storage, involves the process of storing data as a service.
Example :

- Database - Google Big Table, Amazon SimpleDB.
- Network Attached Storage - Nirvanix CloudNAS, MobileMe iDisk.

f. Cloud Infrastructure, delivery of computing infrastructure as a service.
Example:

- Grid Computing - Sun Grid.
- Full Virtualization - GoGrid, Skytap.
- Compute - Amazon Elastic Compute Cloud

Underlying Principles of Parallel and Distributed Computing System - CS8791 Cloud Computing - Unit 1

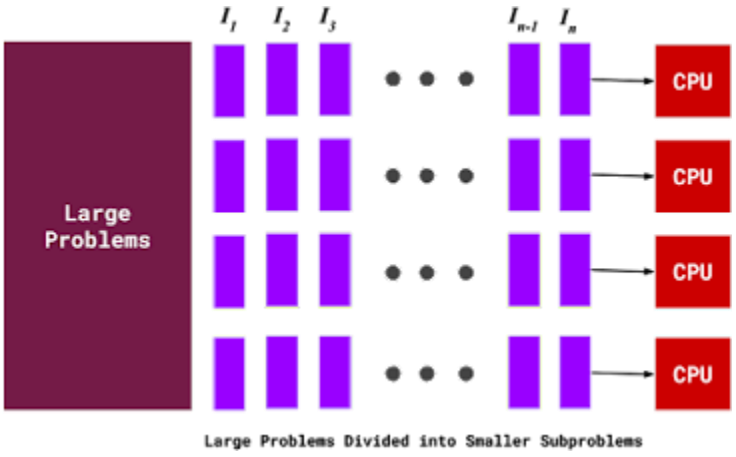
by**Admin**-October 03, 20201 Comments

Underlying Principles of Parallel and Distributed Computing System

- The terms parallel computing and distributed computing are used interchangeably.
- It implies a tightly coupled system.
- It is characterised by homogeneity of components (Uniform Structure).
- Multiple Processors share the same physical memory.

Parallel Processing

- Processing multiple tasks simultaneously in multiple processors is called parallel processing.
- Parallel program consists of multiple processes (tasks) simultaneously solving a given problem.
- Divide-and-Conquer technique is used.



Applications for Parallel Processing

- Science and Engineering
  - Atmospheric Analysis
  - Earth Sciences
  - Electrical Circuit Design
- Industrial and Commercial
  - Data Mining
  - Web Search Engine
  - Graphics Processing

Why to use parallel processing

- **Save time and money:** More resources at a task will shorten its time for completion, with potential cost savings.
- **Provide concurrency:** Single computing resources can only do one task at a time.
- **Serial computing limits:** Transmission speeds depend directly upon hardware.

Parallel computing memory architecture types

1. Shared memory architecture
2. Distributed memory architecture
3. Hybrid distributed shared memory architecture

1. Shared Memory Architecture

- All processes access all memory as a global address space.
- Multiple processors can operate independently but they share same memory resources available.
- Change in my location - affected by processor.

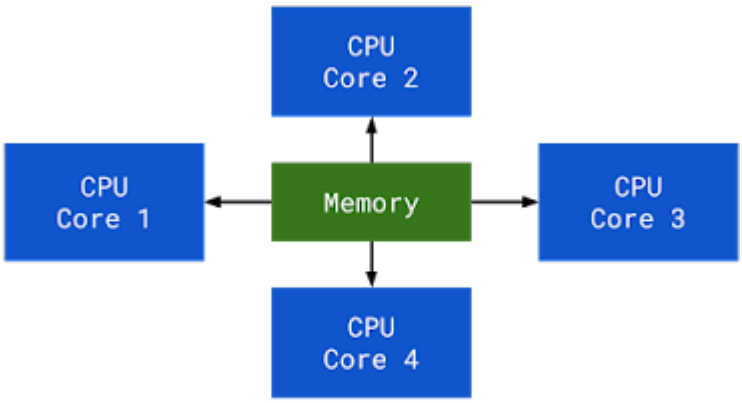


Figure: Memory Shared on a Quad Core Processor

https://cse-r17.blogspot.com

- Shared memory is classified into two types,
  - Uniform Memory Access (UMA)
  - Non-Uniform Memory Access (NUMA)

Uniform Memory Access (UMA)

- It is represented by symmetric multiprocessor machines
- Identical processes equal access and access time to memory

Non-Uniform Memory Access (NUMA)

- It can directly access the memory of another SMP.
- Equal access time to all memories.
- Memory access across the link is slower.

Advantages

- It is a global address space.
- User friendly programming prospective to memory.
- Data sharing between tasks is both fast and uniform due to the memory CPU.

Disadvantages

- Scalability between memory and CPU
    - Increases traffic associated with cache or memory.
- Synchronisation constructs the correct access to global memory.

2. Distributed Memory Architecture

- A distributed system is a collection of large amount of independent computers that appear to its users as a single coherent system.
- Distributed Systems have their own local memory

Advantages

- Memory is scalable with the number of processors
- Processes can be accessed rapidly with its own memory without any interference
- Cost effective use of commodity processors and networking
- Disadvantages
- The programmer is responsible for many of these details associated with data communication between processes
- Non-uniform memory access time data residing in a remote node to access local data

3. Hybrid distributed shared memory

- The shared memory component can be a shared memory machine or Graphics Processing Unit (GPU).
- Distributed memory components used in networking of multiple shared memory on GPU machines

Advantages

- Increase scalability.
- Whatever is common to both shared and distributed memory architecture.

Disadvantages

- Increased programmer complexity is an important disadvantage.

INTRODUCTION

Cloud computing stages is an on-demand openness of PC system resources, especially data stockpiling and figuring power, without a direct dynamic organization by the customer. The thought has advanced by a wide edge all through the long haul and has isolated into iaas, paas, and much more continue being discovered each passing season ahead. Distributed computing stage is ideal for the current IT culture, and it isn't going wherever, anytime sooner rather than later.

OPEN SOURCE CLOUD SOFTWARE AND SOLUTIONS LIST:

1. **Open Stack**
2. **Cloud Stack**
3. **Apache Mesos**
4. **Eucalyptus**
5. **Open Nebula**
6. **AppScale**

1. OPEN STACK

Open stack is a lot of open source cloud software programming contraptions for regulating distributed computing stages for public and private clouds. This programming stage is contained interrelated parts that control grouped, multi-dealer hardware pools of taking care of, amassing, and frameworks organization resources all through a server farm. Open Stack could be regulated through an electronic dashboard, through request line instruments, or through peaceful web organizations.

Key Highlights of Open Stack:

• Limitless accumulating: Tremendous and level namespace, incredibly versatile read/make access, prepared to serve substance clearly from the limit structure.

• Multi-dimensional adaptability: cale-out designing: It scales vertically and equitably scattered limit. It can back up and chronicles a ton of data with straight execution.

• Record/holder: No settling, not a standard record system. It scales to various petabytes just as billions of articles.

2. CLOUD STACK

Cloud stack is an open source cloud software platform expected to pass on and administer immense associations of the virtual machine, as a significantly available, especially adaptable establishment as an assistance distributed computing. It's a java-based undertaking that gives an organization labourer, and trained professionals (if essential) for hypervisor has so you can likewise run an iaas cloud. Cloud stack as of now reinforces the most well-known hypervisors: VMware, kvm, citric XenServer, xen cloud Stage (XCP), Prophet VM specialist and MS Hyper-v.

Key Highlights of Cloud stack:

• Works with have running xen worker, kvm, Hyper-v, just as VMware esxi with vsphere .

• Gives an agreeable Electronic UI to managing the cloud.

• Gives a nearby Programming interface. Customers can bargain their cloud with an easy to use Web interface, request line mechanical assemblies, or conceivably a full-included Serene Programming interface.

3. APACHE MESOS

Apache mesos is a complete open-source solution that handles occupations capably in a passed on the environment through ground-breaking resource sharing just as disconnection. It dynamic PC processor, memory, storing, and other register resources from machines, enabling issue liberal and adaptable appropriated structures to be helpfully built and run sufficiently.

Features of Mesos

• Mesos is a cross-stage: It runs on Linux, osx and Windows. It is a Cloud provider freethinker all the while.

• Local assistance for dispatching compartments with Docker and appc pictures.

• Accomplishes staggering levels of High Availability: Issue tolerant repeated master and experts using Creature controller. Non-hazardous updates.

4. EUCALYPTUS

Eucalyptus is an open source cloud software storage for building aws-feasible private and hybrid clouds. It is a Linux based programming designing that executes versatile private and cross variety cloud inside your present IT establishment. As and on-premise System as a Help cloud game plan, it licenses you to use your own collections of resources (hardware, storing, and association) using a self-organization interface subordinate upon the circumstance.

Key Highlights of Eucalyptus:

• Design of Eucalyptus is awes Practical and appropriately has five key parts, Cloud controller, Walrus, Pack controller, Accumulating controller, Center controller and Euca2ool.

• Clients can likewise run Amazon or Eucalyptus machine pictures as events on both the cloud.

• Since it is aws suitable, there is 100% AWS Programming interface similitude ans maintain.

5. OPEN NEBULA

Open nebula is clear yet mind boggling and versatile turnkey open source answer for manufacture Private Cloud and regulate Worker ranch virtualization. It completes IaaS. The chief open-source variation of Open nebula was conveyed in Walk 2008.

Key Highlights of Open cloud:

• Fine-grained upper leg ligaments for resource task.

• Asset Offer the chiefs to follow and confine figuring, amassing and frameworks organization resource utilization

• Dynamic creation of Bundles as pool of hosts that shares information stores and virtual associations for load changing, high availability, and prevalent enrolling.

6. APPSCALE

Appscale is an open source cloud software distributed computing stage that thus passes on and scales unmodified Google Application Engine applications over open and private fog systems. It's a circled programming system that completes a cloud stage as assistance (PaaS). In light of everything, AppScale is an easy to-regulate worker less stage for building and for running flexible web and versatile applications on any establishment. The objective of Appscale is to give designers a quick, programming interface driven improvement stages that can run application on any cloud foundation.

Highlights of AppScale:

• Snappy prototyping

• AppScale isn't hard to use hence making associations favour it.

CONCLUSION

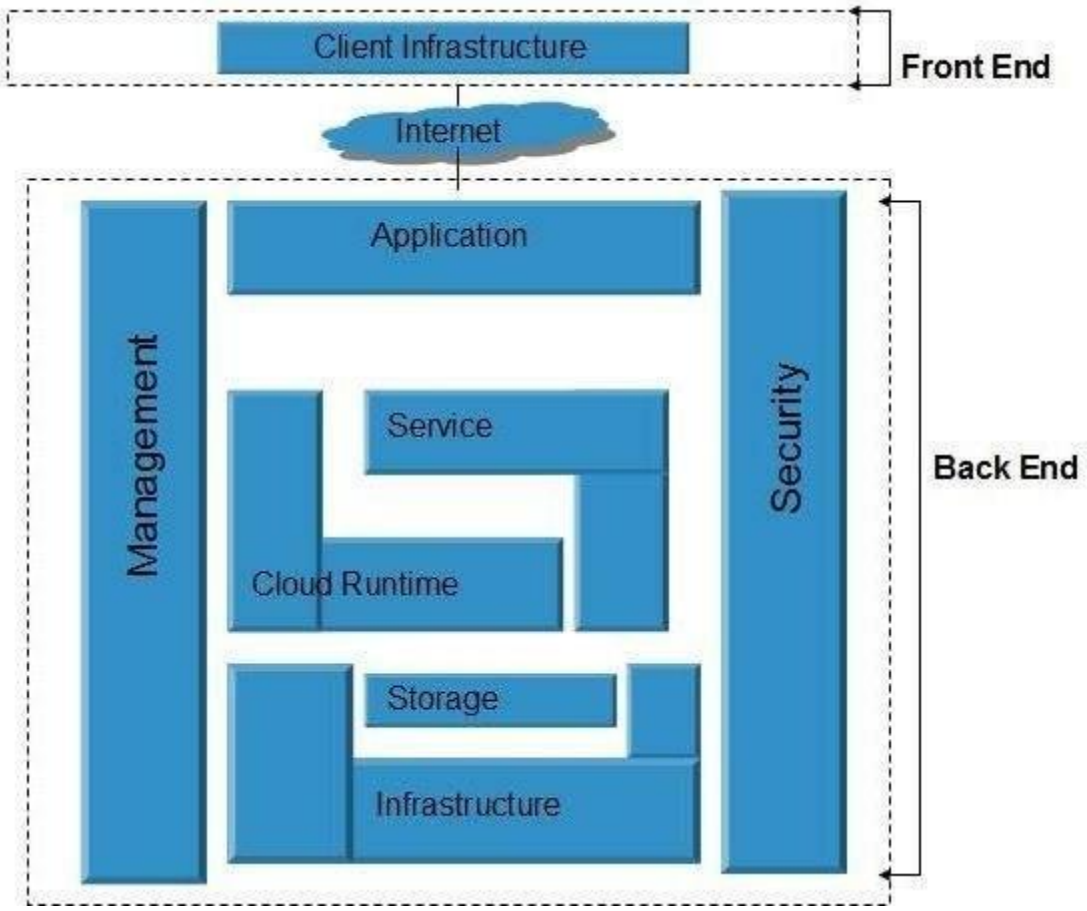
The result shows that with respect to costing is more intelligent to use open source procedure. In future also if we need to make a model that used open source cloud software in three layers of organization in cloud computing tools like SaaS, Paas, and IaaS . To be full of control by customer and measure execution and nature of organizations.

Cloud Computing Architecture

Cloud Computing architecture comprises of many cloud components, which are loosely coupled. We can broadly divide the cloud architecture into two parts:

- Front End
- Back End

Each of the ends is connected through a network, usually Internet. The following diagram shows the graphical view of cloud computing architecture:



Front End

The **front end** refers to the client part of cloud computing system. It consists of interfaces and applications that are required to access the cloud computing platforms, Example - Web Browser.

Back End

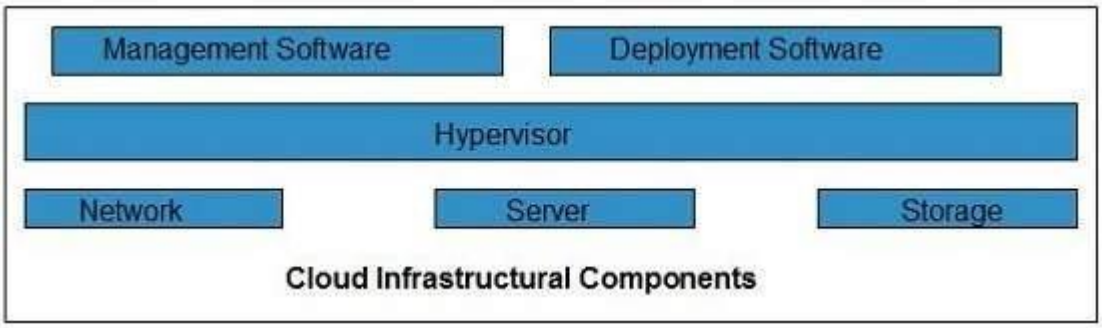
The **back End** refers to the cloud itself. It consists of all the resources required to provide cloud computing services. It comprises of huge data storage, virtual machines, security mechanism, services, deployment models, servers, etc.

Note

- It is the responsibility of the back end to provide built-in security mechanism, traffic control and protocols.
- The server employs certain protocols known as middleware, which help the connected devices to communicate with each other.

Cloud Computing Infrastructure

Cloud infrastructure consists of servers, storage devices, network, cloud management software, deployment software, and platform virtualization.



Hypervisor

Hypervisor is a **firmware** or **low-level program** that acts as a Virtual Machine Manager. It allows to share the single physical instance of cloud resources between several tenants.

Management Software

It helps to maintain and configure the infrastructure.

Deployment Software

It helps to deploy and integrate the application on the cloud.

Network

It is the key component of cloud infrastructure. It allows to connect cloud services over the Internet. It is also possible to deliver network as a utility over the Internet, which means, the customer can customize the network route and protocol.

Server

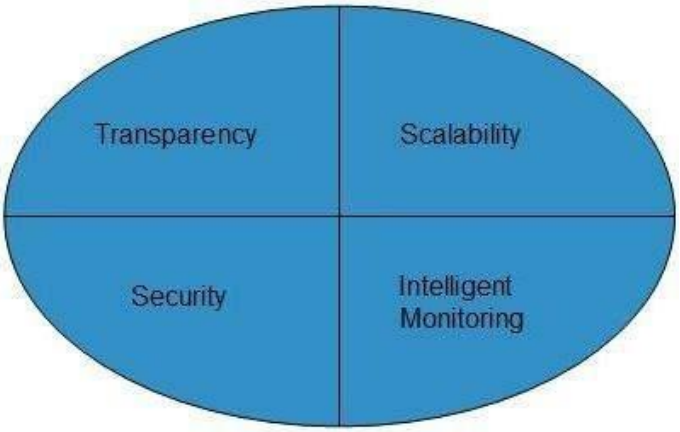
The **server** helps to compute the resource sharing and offers other services such as resource allocation and de-allocation, monitoring the resources, providing security etc.

Storage

Cloud keeps multiple replicas of storage. If one of the storage resources fails, then it can be extracted from another one, which makes cloud computing more reliable.

Infrastructural Constraints

Fundamental constraints that cloud infrastructure should implement are shown in the following diagram:



Transparency

Virtualization is the key to share resources in cloud environment. But it is not possible to satisfy the demand with single resource or server. Therefore, there must be transparency in resources, load balancing and application, so that we can scale them on demand.

Scalability

Scaling up an application delivery solution is not that easy as scaling up an application because it involves configuration overhead or even re-architecting the network. So, application delivery solution is need to be scalable which will require the virtual infrastructure such that resource can be provisioned and de-provisioned easily.

Intelligent Monitoring

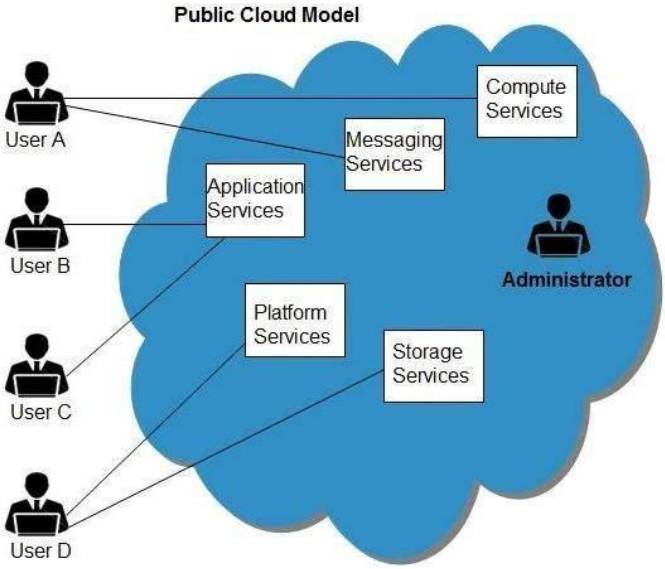
To achieve transparency and scalability, application solution delivery will need to be capable of intelligent monitoring.

Security

The mega data center in the cloud should be securely architected. Also the control node, an entry point in mega data center, also needs to be secure.

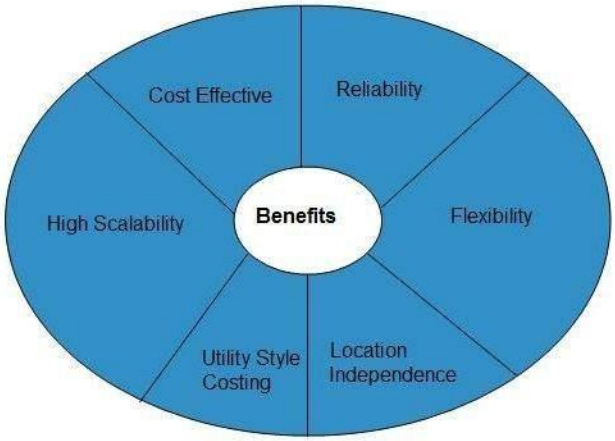
Public Cloud Model

Public Cloud allows systems and services to be easily accessible to general public. The IT giants such as **Google, Amazon** and **Microsoft** offer cloud services via Internet. The Public Cloud Model is shown in the diagram below.



Benefits

There are many benefits of deploying cloud as public cloud model. The following diagram shows some of those benefits:



Cost Effective

Since **public cloud** shares same resources with large number of customers it turns out inexpensive.

Reliability

The **public cloud** employs large number of resources from different locations. If any of the resources fails, public cloud can employ another one.

Flexibility

The public cloud can smoothly integrate with private cloud, which gives customers a flexible approach.

Location Independence

Public cloud services are delivered through Internet, ensuring location independence.

Utility Style Costing

Public cloud is also based on **pay-per-use** model and resources are accessible whenever customer needs them.

High Scalability

Cloud resources are made available on demand from a pool of resources, i.e., they can be scaled up or down according the requirement.

Disadvantages

Here are some disadvantages of public cloud model:

Low Security

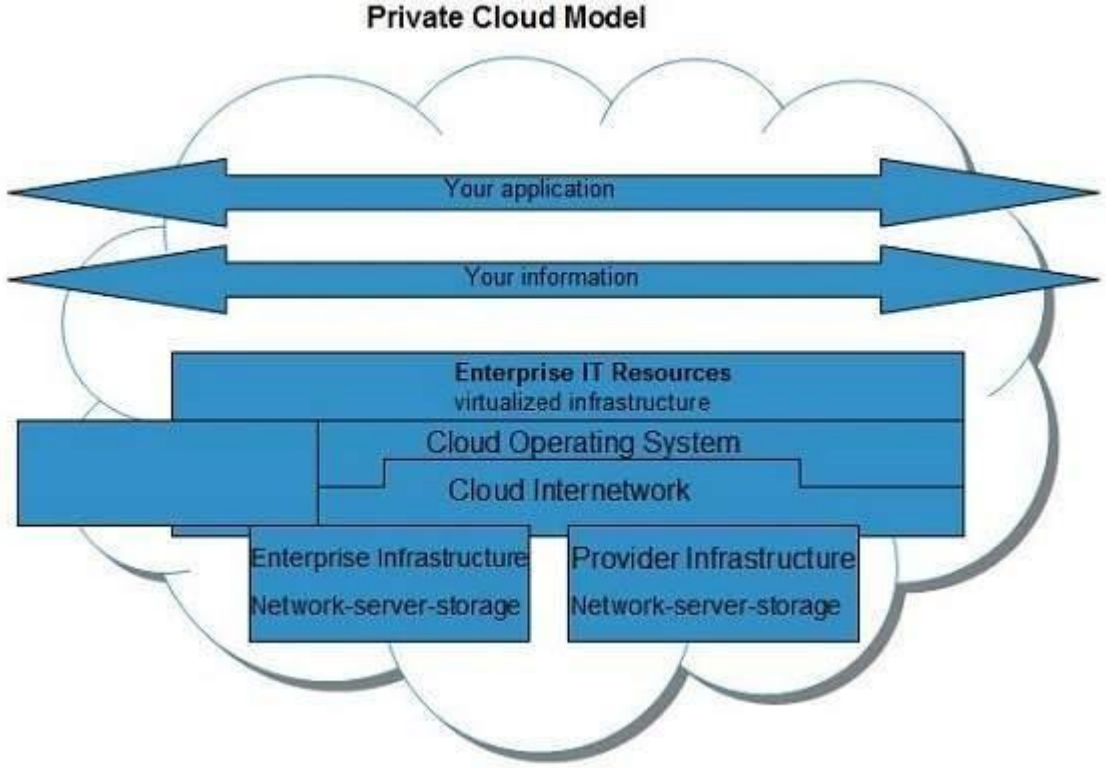
In **public cloud model,** data is hosted off-site and resources are shared publicly, therefore does not ensure higher level of security.

Less Customizable

It is comparatively less customizable than private cloud.

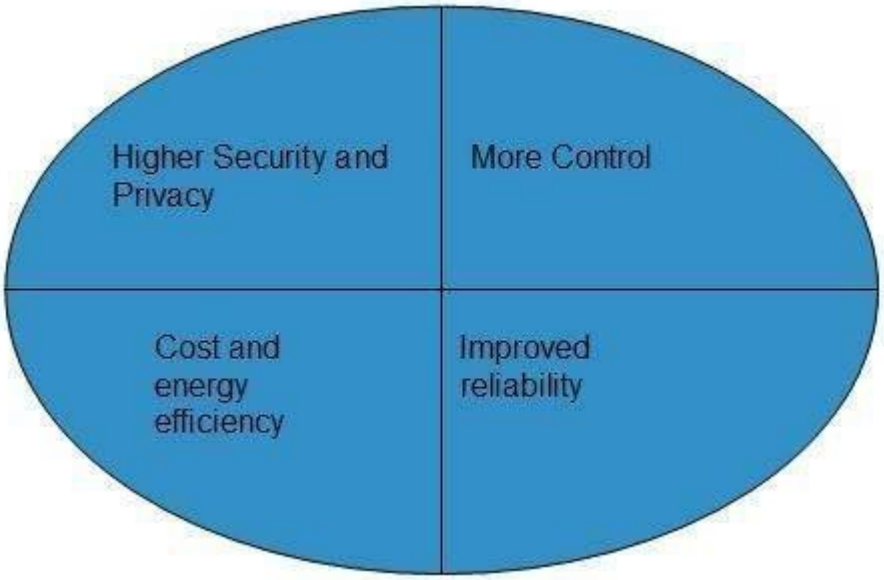
Private Cloud Model

Private Cloud allows systems and services to be accessible within an organization. The Private Cloud is operated only within a single organization. However, it may be managed internally by the organization itself or by third-party. The private cloud model is shown in the diagram below.



Benefits

There are many benefits of deploying cloud as private cloud model. The following diagram shows some of those benefits:



High Security and Privacy

Private cloud operations are not available to general public and resources are shared from distinct pool of resources. Therefore, it ensures high **security** and **privacy.**

More Control

The **private cloud** has more control on its resources and hardware than public cloud because it is accessed only within an organization.

Cost and Energy Efficiency

The **private cloud** resources are not as cost effective as resources in public clouds but they offer more efficiency than public cloud resources.

Disadvantages

Here are the disadvantages of using private cloud model:

Restricted Area of Operation

The private cloud is only accessible locally and is very difficult to deploy globally.

High Priced

Purchasing new hardware in order to fulfill the demand is a costly transaction.

Limited Scalability

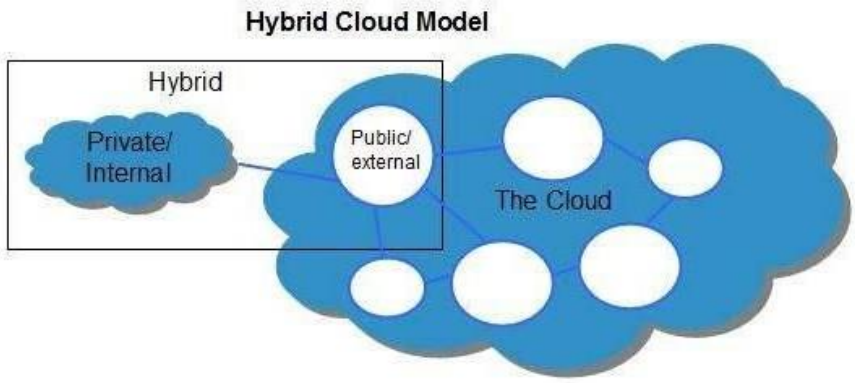
The private cloud can be scaled only within capacity of internal hosted resources.

Additional Skills

In order to maintain cloud deployment, organization requires skilled expertise.

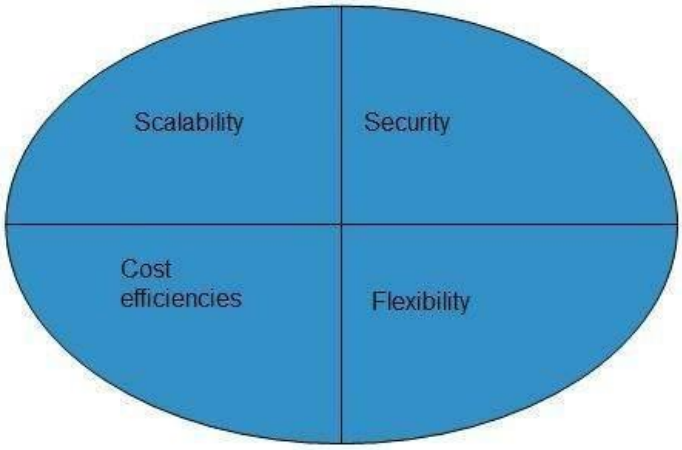
Hybrid Cloud Model

Hybrid Cloud is a mixture of public and private cloud. Non-critical activities are performed using public cloud while the critical activities are performed using private cloud. The Hybrid Cloud Model is shown in the diagram below.



Benefits

There are many benefits of deploying cloud as hybrid cloud model. The following diagram shows some of those benefits:



Scalability

It offers features of both, the public cloud scalability and the private cloud scalability.

Flexibility

It offers secure resources and scalable public resources.

Cost Efficiency

Public clouds are more cost effective than private ones. Therefore, hybrid clouds can be cost saving.

Security

The private cloud in hybrid cloud ensures higher degree of security.

Disadvantages

Networking Issues

Networking becomes complex due to presence of private and public cloud.

Security Compliance

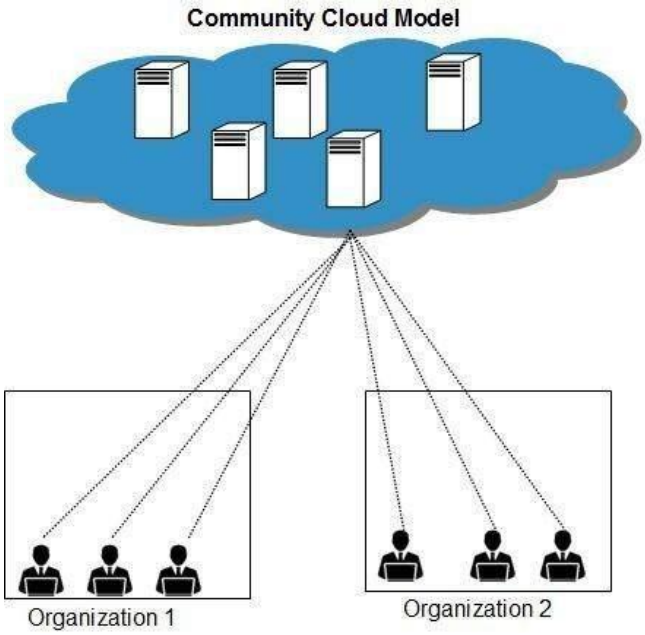
It is necessary to ensure that cloud services are compliant with security policies of the organization.

Infrastructure Dependency

The **hybrid cloud model** is dependent on internal IT infrastructure, therefore it is necessary to ensure redundancy across data centers.

Community Cloud Model

Community Cloud allows system and services to be accessible by group of organizations. It shares the infrastructure between several organizations from a specific community. It may be managed internally by organizations or by the third-party. The Community Cloud Model is shown in the diagram below.



Benefits

There are many benefits of deploying cloud as **community cloud model.**

Cost Effective

Community cloud offers same advantages as that of private cloud at low cost.

Sharing Among Organizations

Community cloud provides an infrastructure to share cloud resources and capabilities among several organizations.

Security

The community cloud is comparatively more secure than the public cloud but less secured than the private cloud.

Issues

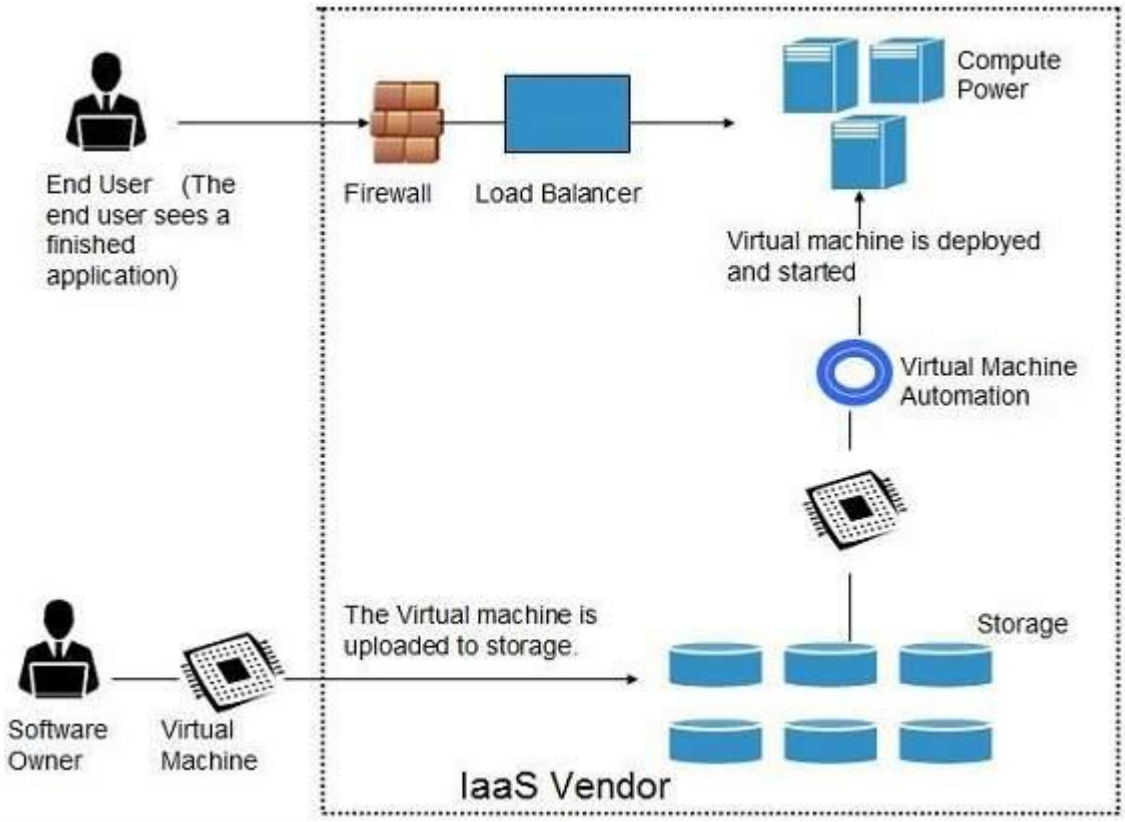
- Since all data is located at one place, one must be careful in storing data in community cloud because it might be accessible to others.
- It is also challenging to allocate responsibilities of governance, security and cost among organizations.

Cloud Computing Infrastructure as a Service (IaaS)

Infrastructure-as-a-Service provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc. Apart from these resources, the IaaS also offers:

- Virtual machine disk storage
- Virtual local area network (VLANs)
- Load balancers
- IP addresses
- Software bundles

All of the above resources are made available to end user via **server virtualization.** Moreover, these resources are accessed by the customers as if they own them.



Benefits

IaaS allows the cloud provider to freely locate the infrastructure over the Internet in a cost-effective manner. Some of the key benefits of IaaS are listed below:

- Full control of the computing resources through administrative access to VMs.
- Flexible and efficient renting of computer hardware.

- Portability, interoperability with legacy applications.

Full control over computing resources through administrative access to VMs

IaaS allows the customer to access computing resources through administrative access to virtual machines in the following manner:

- Customer issues administrative command to cloud provider to run the virtual machine or to save data on cloud server.
- Customer issues administrative command to virtual machines they owned to start web server or to install new applications.

Flexible and efficient renting of computer hardware

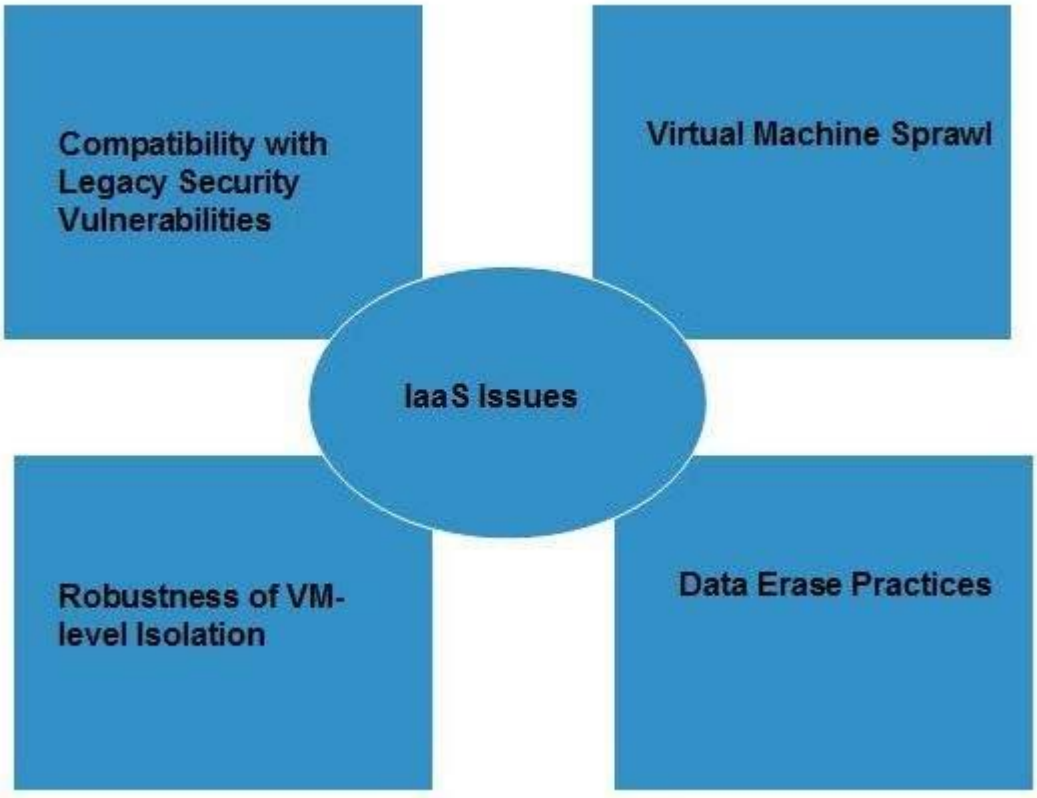
IaaS resources such as virtual machines, storage devices, bandwidth, IP addresses, monitoring services, firewalls, etc. are made available to the customers on rent. The payment is based upon the amount of time the customer retains a resource. Also with administrative access to virtual machines, the customer can run any software, even a custom operating system.

Portability, interoperability with legacy applications

It is possible to maintain legacy between applications and workloads between IaaS clouds. For example, network applications such as web server or e-mail server that normally runs on customer-owned server hardware can also run from VMs in IaaS cloud.

Issues

IaaS shares issues with PaaS and SaaS, such as Network dependence and browser based risks. It also has some specific issues, which are mentioned in the following diagram:



Compatibility with legacy security vulnerabilities

Because IaaS offers the customer to run legacy software in provider's infrastructure, it exposes customers to all of the security vulnerabilities of such legacy software.

Virtual Machine sprawl

The VM can become out-of-date with respect to security updates because IaaS allows the customer to operate the virtual machines in running, suspended and off state. However, the provider can automatically update such VMs, but this mechanism is hard and complex.

Robustness of VM-level isolation

IaaS offers an isolated environment to individual customers through hypervisor. Hypervisor is a software layer that includes hardware support for virtualization to split a physical computer into multiple virtual machines.

Data erase practices

The customer uses virtual machines that in turn use the common disk resources provided by the cloud provider. When the customer releases the resource, the cloud provider must ensure that next customer to rent the resource does not observe data residue from previous customer.

Characteristics

Here are the characteristics of IaaS service model:

- Virtual machines with pre-installed software.

- Virtual machines with pre-installed operating systems such as Windows, Linux, and Solaris.
- On-demand availability of resources.
- Allows to store copies of particular data at different locations.
- The computing resources can be easily scaled up and down.

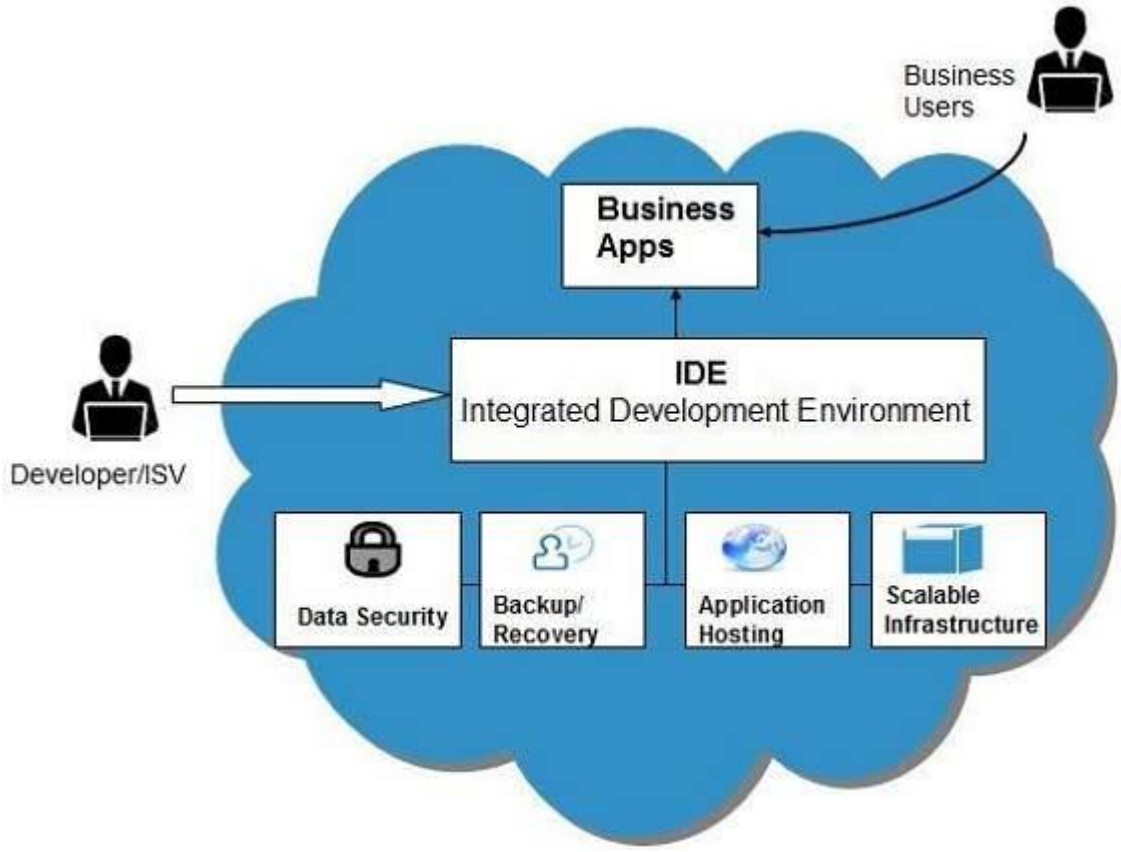
Cloud Computing Platform as a Service (PaaS)

Platform-as-a-Service offers the runtime environment for applications. It also offers development and deployment tools required to develop applications. PaaS has a feature of **point-and-click** tools that enables non-developers to create web applications.

App Engine of Google and **Force.com** are examples of PaaS offering vendors. Developer may log on to these websites and use the **built-in API** to create web-based applications.

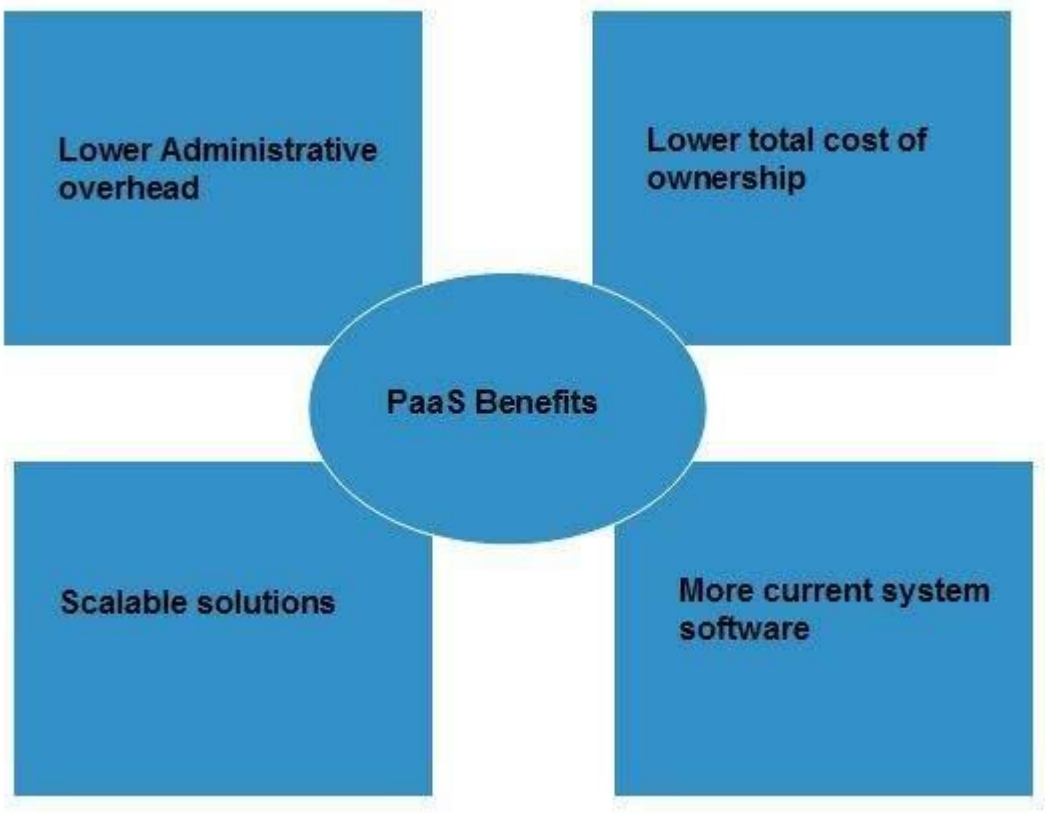
But the disadvantage of using PaaS is that, the developer **locks-in** with a particular vendor. For example, an application written in Python against API of Google, and using App Engine of Google is likely to work only in that environment.

The following diagram shows how PaaS offers an API and development tools to the developers and how it helps the end user to access business applications.



Benefits

Following are the benefits of PaaS model:



Lower administrative overhead

Customer need not bother about the administration because it is the responsibility of cloud provider.

Lower total cost of ownership

Customer need not purchase expensive hardware, servers, power, and data storage.

Scalable solutions

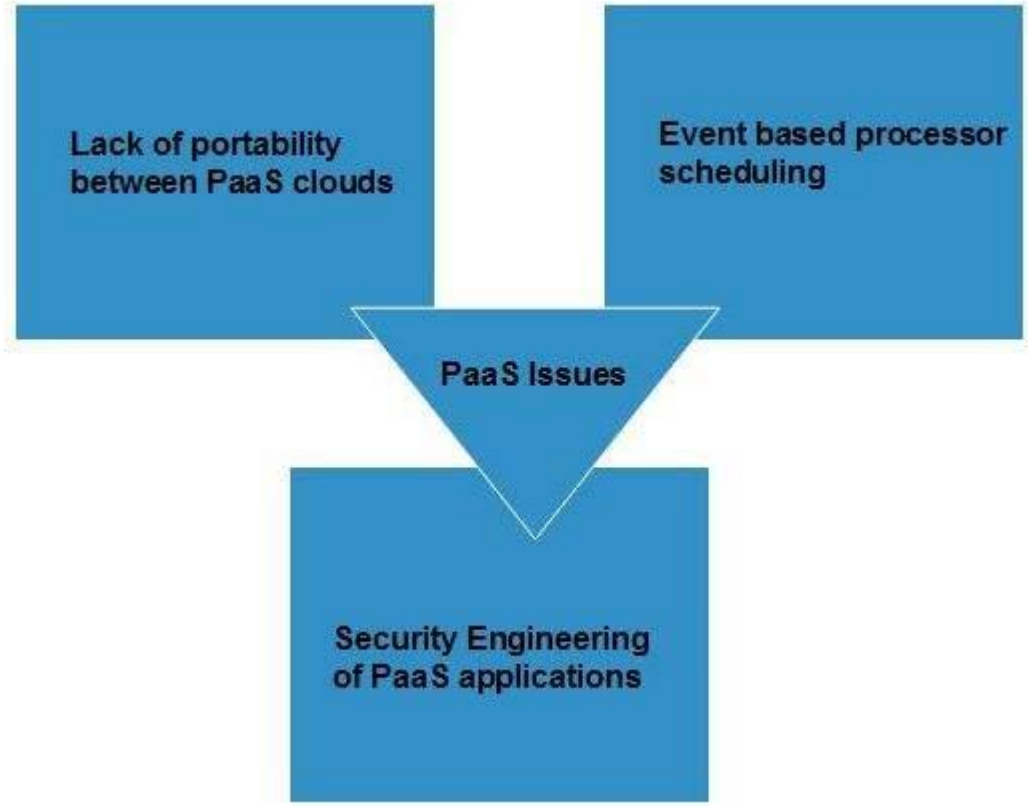
It is very easy to scale the resources up or down automatically, based on their demand.

More current system software

It is the responsibility of the cloud provider to maintain software versions and patch installations.

Issues

Like **SaaS, PaaS** also places significant burdens on customer's browsers to maintain reliable and secure connections to the provider’s systems. Therefore, PaaS shares many of the issues of SaaS. However, there are some specific issues associated with PaaS as shown in the following diagram:



Lack of portability between PaaS clouds

Although standard languages are used, yet the implementations of platform services may vary. For example, file, queue, or hash table interfaces of one platform may differ from another, making it difficult to transfer the workloads from one platform to another.

Event based processor scheduling

The PaaS applications are event-oriented which poses resource constraints on applications, i.e., they have to answer a request in a given interval of time.

Security engineering of PaaS applications

Since PaaS applications are dependent on network, they must explicitly use cryptography and manage security exposures.

Characteristics

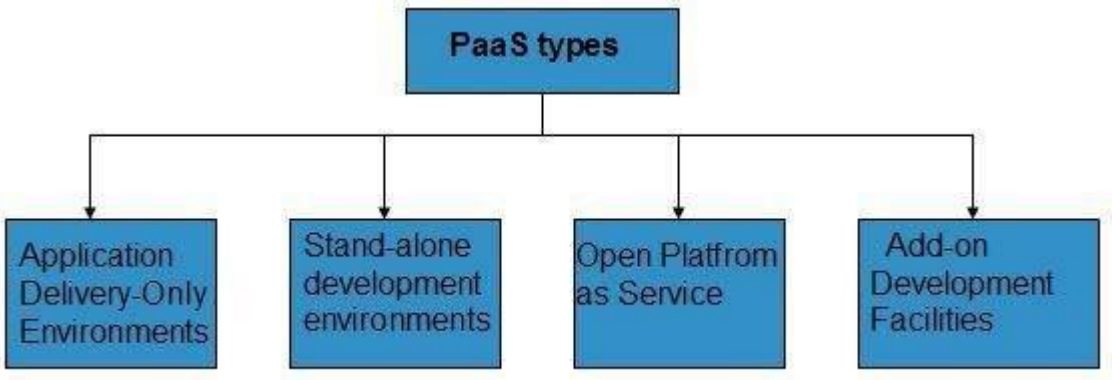
Here are the characteristics of PaaS service model:

- PaaS offers **browser based development environment.** It allows the developer to create database and edit the application code either via Application Programming Interface or point-and-click tools.
- PaaS provides **built-in security, scalability,** and **web service interfaces.**
- PaaS provides built-in tools for defining **workflow, approval processes,** and business rules.
- It is easy to integrate PaaS with other applications on the same platform.

- PaaS also provides web services interfaces that allow us to connect the applications outside the platform.

PaaS Types

Based on the functions, PaaS can be classified into four types as shown in the following diagram:



Stand-alone development environments

The **stand-alone PaaS** works as an independent entity for a specific function. It does not include licensing or technical dependencies on specific SaaS applications.

Application delivery-only environments

The **application delivery PaaS** includes **on-demand scaling** and **application security.**

Open platform as a service

Open PaaS offers an **open source software** that helps a PaaS provider to run applications.

Add-on development facilities

The **add-on PaaS** allows to customize the existing SaaS platform.

Cloud Computing Software as a Service (SaaS)

Software-as–a-Service (SaaS) model allows to provide software application as a service to the end users. It refers to a software that is deployed on a host service and is accessible via Internet. There are several SaaS applications listed below:

- Billing and invoicing system
- Customer Relationship Management (CRM) applications
- Help desk applications
- Human Resource (HR) solutions

Some of the SaaS applications are not customizable such as **Microsoft Office**

Suite. But SaaS provides us **Application Programming Interface (API),** which allows the developer to develop a customized application.

Characteristics

Here are the characteristics of SaaS service model:

- SaaS makes the software available over the Internet.
- The software applications are maintained by the vendor.
- The license to the software may be subscription based or usage based. And it is billed on recurring basis.
- SaaS applications are cost-effective since they do not require any maintenance at end user side.
- They are available on demand.
- They can be scaled up or down on demand.
- They are automatically upgraded and updated.
- SaaS offers shared data model. Therefore, multiple users can share single instance of infrastructure. It is not required to hard code the functionality for individual users.
- All users run the same version of the software.

Benefits

Using SaaS has proved to be beneficial in terms of scalability, efficiency and performance. Some of the benefits are listed below:

- Modest software tools
- Efficient use of software licenses
- Centralized management and data
- Platform responsibilities managed by provider
- Multitenant solutions

Modest software tools

The SaaS application deployment requires a little or no client side software installation, which results in the following benefits:

- No requirement for complex software packages at client side
- Little or no risk of configuration at client side
- Low distribution cost

Efficient use of software licenses

The customer can have single license for multiple computers running at different locations which reduces the licensing cost. Also, there is no requirement for license servers because the software runs in the provider's infrastructure.

Centralized management and data

The cloud provider stores data centrally. However, the cloud providers may store

data in a decentralized manner for the sake of redundancy and reliability.

Platform responsibilities managed by providers

All platform responsibilities such as backups, system maintenance, security, hardware refresh, power management, etc. are performed by the cloud provider. The customer does not need to bother about them.

Multitenant solutions

Multitenant solutions allow multiple users to share single instance of different resources in virtual isolation. Customers can customize their application without affecting the core functionality.

Issues

There are several issues associated with SaaS, some of them are listed below:

- Browser based risks
- Network dependence
- Lack of portability between SaaS clouds

Browser based risks

If the customer visits malicious website and browser becomes infected, the subsequent access to SaaS application might compromise the customer's data.

To avoid such risks, the customer can use multiple browsers and dedicate a specific browser to access SaaS applications or can use virtual desktop while accessing the SaaS applications.

Network dependence

The SaaS application can be delivered only when network is continuously available. Also network should be reliable but the network reliability cannot be guaranteed either by cloud provider or by the customer.

Lack of portability between SaaS clouds

Transferring workloads from one SaaS cloud to another is not so easy because work flow, business logics, user interfaces, support scripts can be provider specific.

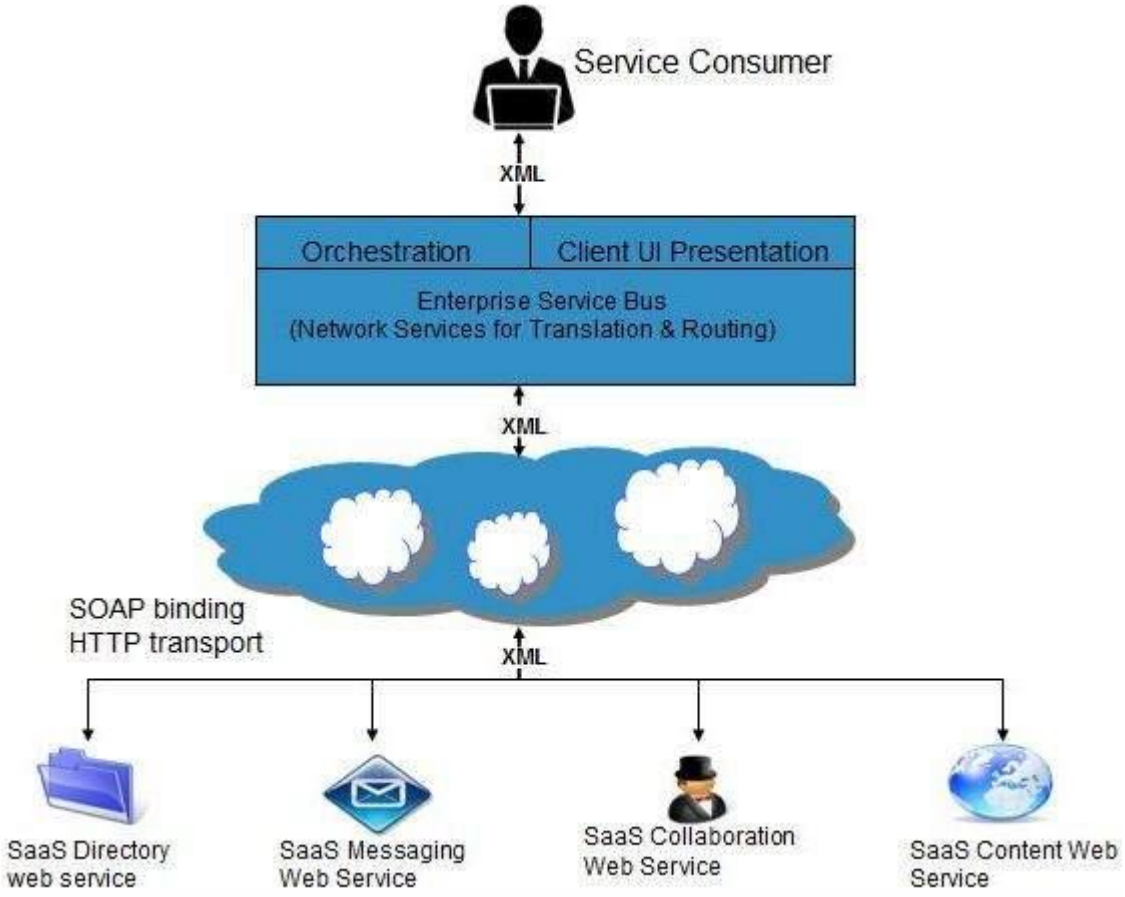
Open SaaS and SOA

Open SaaS uses those SaaS applications, which are developed using open source programming language. These SaaS applications can run on any open source operating system and database. Open SaaS has several benefits listed below:

- No License Required
- Low Deployment Cost
- Less Vendor Lock-in

- More portable applications
- More Robust Solution

The following diagram shows the SaaS implementation based on SOA:



Cloud Computing
Identity as a Service
(IDaaS)

Employees in a company require to login to system to perform various tasks. These systems may be based on local server or cloud based. Following are the problems that an employee might face:

- Remembering different username and password combinations for accessing multiple servers.
- If an employee leaves the company, it is required to ensure that each account of that user is disabled. This increases workload on IT staff.

To solve above problems, a new technique emerged which is known as **Identity-as–a-Service (IDaaS).**

IDaaS offers management of identity information as a digital entity. This identity can be used during electronic transactions.

Identity

Identity refers to set of attributes associated with something to make it recognizable. All objects may have same attributes, but their identities cannot be the same. A unique identity is assigned through unique identification attribute.

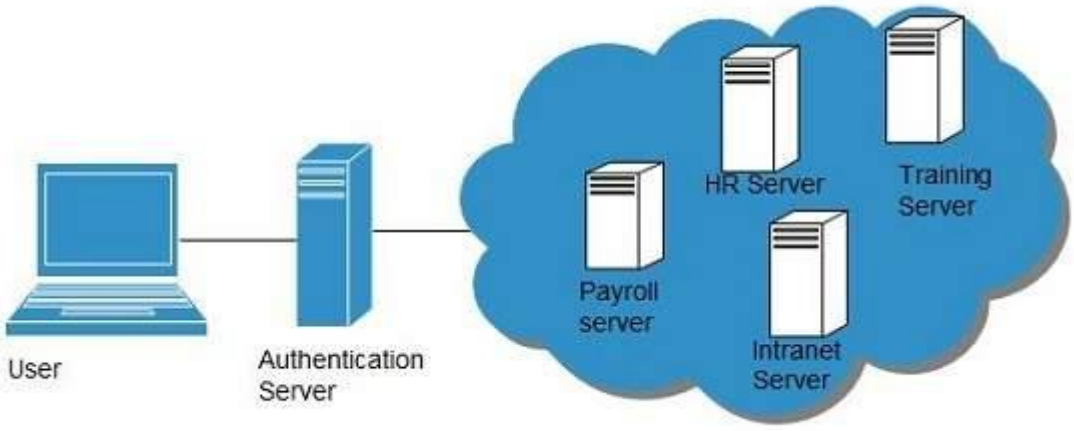
There are several **identity services** that are deployed to validate services such as validating web sites, transactions, transaction participants, client, etc. Identity-as-a-Service may include the following:

- Directory services
- Federated services
- Registration
- Authentication services
- Risk and event monitoring
- Single sign-on services
- Identity and profile management

Single Sign-On (SSO)

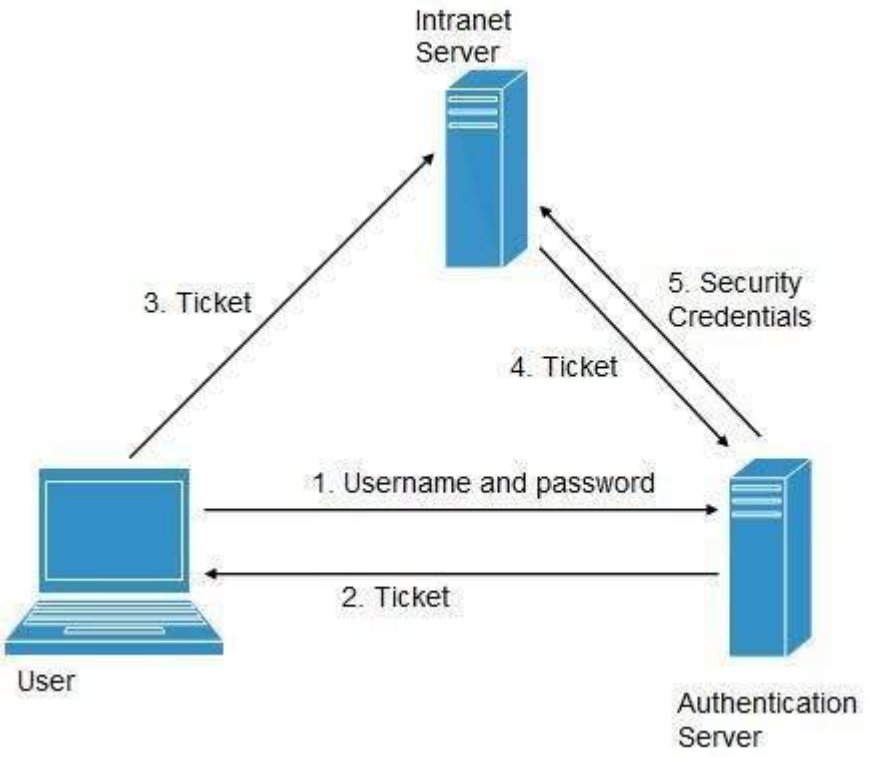
To solve the problem of using different username and password combinations for different servers, companies now employ Single Sign-On software, which allows the user to login only one time and manage the access to other systems.

SSO has single authentication server, managing multiple accesses to other systems, as shown in the following diagram:



SSO Working

There are several implementations of SSO. Here, we discuss the common ones:



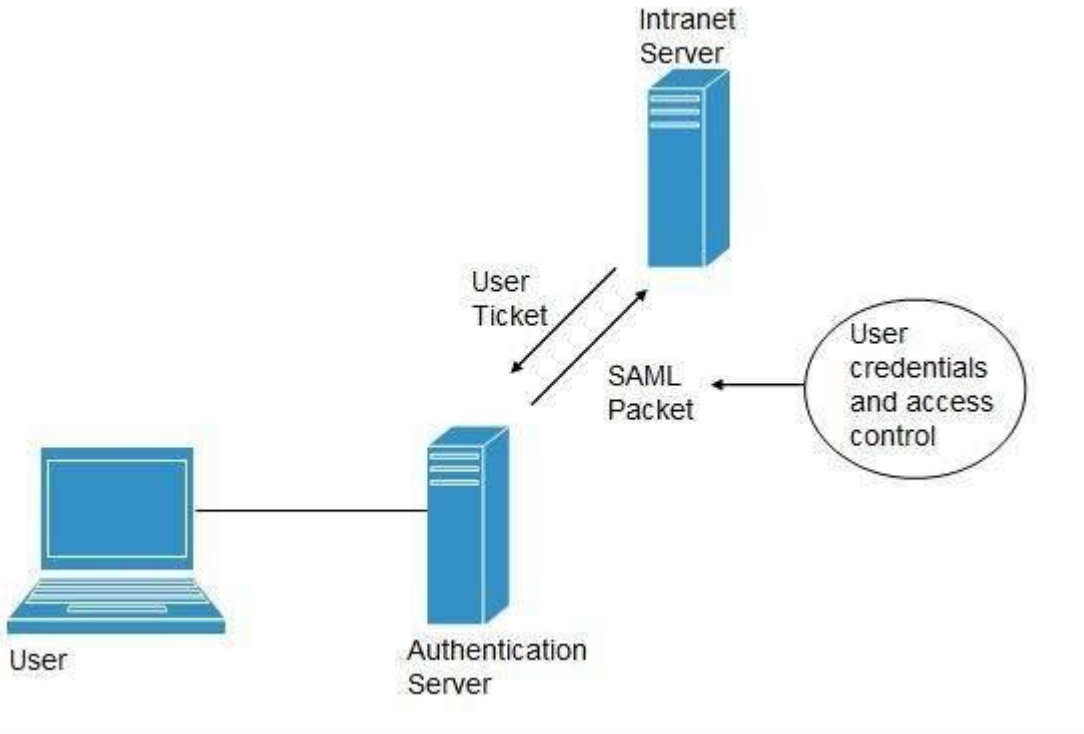
Following steps explain the working of Single Sign-On software:

- User logs into the authentication server using a username and password.
- The authentication server returns the user's ticket.
- User sends the ticket to intranet server.
- Intranet server sends the ticket to the authentication server.
- Authentication server sends the user's security credentials for that server back to the intranet server.

If an employee leaves the company, then disabling the user account at the authentication server prohibits the user's access to all the systems.

Federated Identity Management (FIDM)

FIDM describes the technologies and protocols that enable a user to package security credentials across security domains. It uses **Security Markup Language (SAML)** to package a user's security credentials as shown in the following diagram:



OpenID

It offers users to login into multiple websites with single account. Google, Yahoo!, Flickr, MySpace, WordPress.com are some of the companies that support OpenID.

Benefits

- Increased site conversation rates
- Access to greater user profile content
- Fewer problems with lost passwords
- Ease of content integration into social networking sites

Cloud Computing
Network as a Service
(NaaS)

Network-as-a-Service allows us to access to network infrastructure directly and securely. NaaS makes it possible to deploy **custom routing protocols.**

NaaS uses **virtualized network infrastructure** to provide network services to the customer. It is the responsibility of NaaS provider to maintain and manage the network resources. Having a provider working for a customer decreases the workload of the customer. Moreover, NaaS offers **network as a utility.** NaaS is also based on **pay-per-use model.**

How NaaS is delivered?

To use NaaS model, the customer is required to logon to the web portal, where he

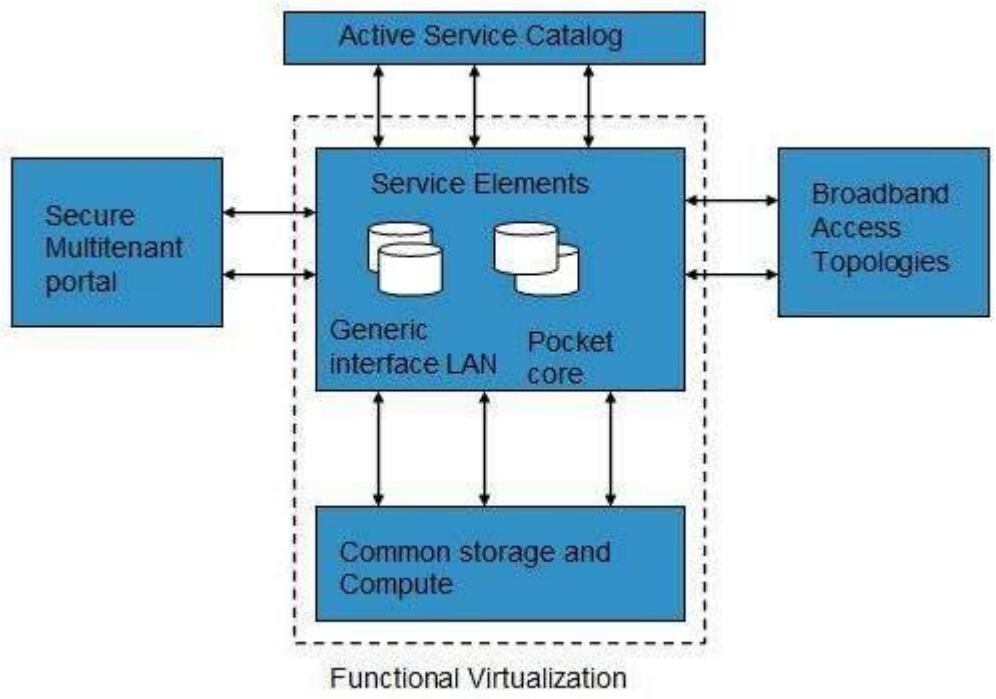
can get online API. Here, the customer can customize the route.

In turn, customer has to pay for the capacity used. It is also possible to turn off the capacity at any time.

Mobile NaaS

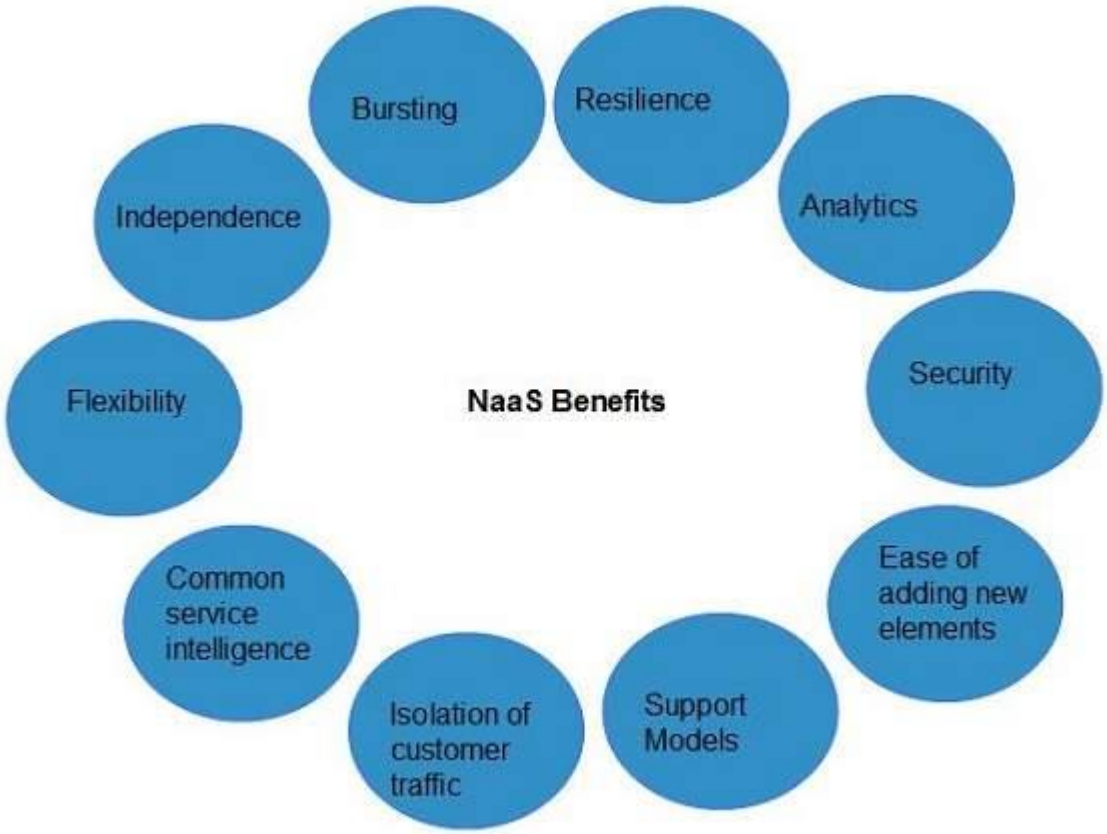
Mobile NaaS offers more efficient and flexible control over mobile devices. It uses virtualization to simplify the architecture thereby creating more efficient processes.

Following diagram shows the Mobile NaaS service elements:



NaaS Benefits

NaaS offers a number of benefits as discussed below:



Independence

Each customer is independent and can segregate the network.

Bursting

The customer pays for high-capacity network only on requirement.

Resilience

The reliability treatments are available, which can be applied for critical applications.

Analytics

The data protection solutions are available, which can be applied for highly sensitive applications.

Ease of Adding New Service Elements

It is very easy to integrate new service elements to the network.

Support Models

A number of support models are available to reduce operation cost.

Isolation of Customer Traffic

The customer traffic is logically isolated.

Cloud storage provider:

A cloud storage provider, also known as a managed service provider (MSP), is **a company that offers organizations and individuals the ability to place and retain data in an off-site**

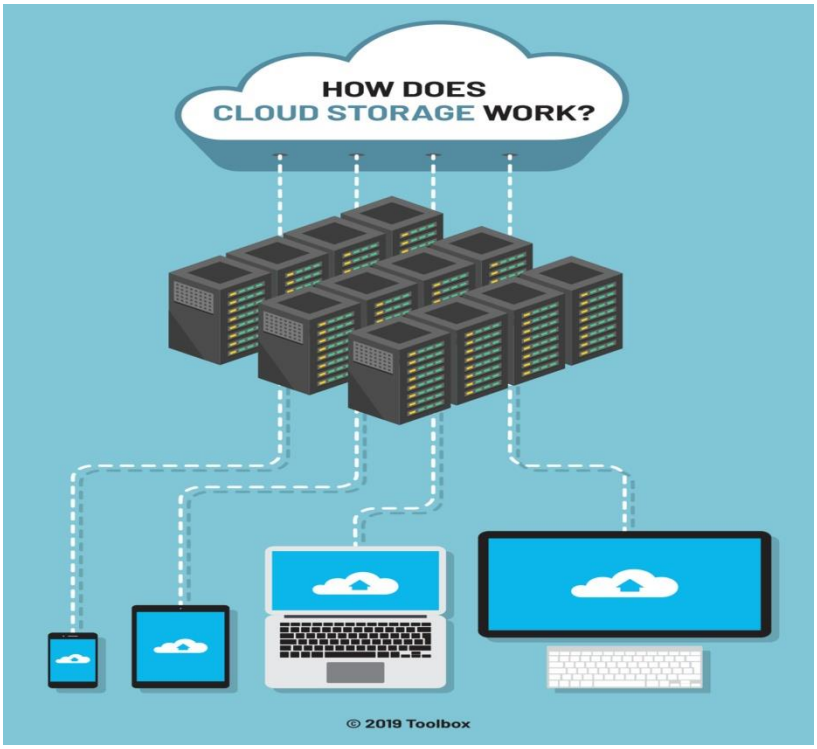
storage system. Customers can lease cloud storage capacity per month or on demand.
What Is Cloud Storage?

Cloud storage is a data deposit model in which digital information such as documents, photos, videos and other forms of media are stored on virtual or cloud servers hosted by third parties. It allows you to transfer data on an offsite storage system and access them whenever needed.

Cloud storage is a cloud computing model that allows users to save important data or media files on remote, third-party servers. Users can access these servers at any time over the internet. Also known as utility storage, cloud storage is maintained and operated by a cloud-based service provider.

From greater accessibility to data backup, cloud storage offers a host of benefits. The most notable being large storage capacity and minimal costs. Cloud storage delivers on-demand and eliminates the need to purchase and manage your own data storage infrastructure. With "anytime, anywhere" data access, this gives you agility, global scale and durability.

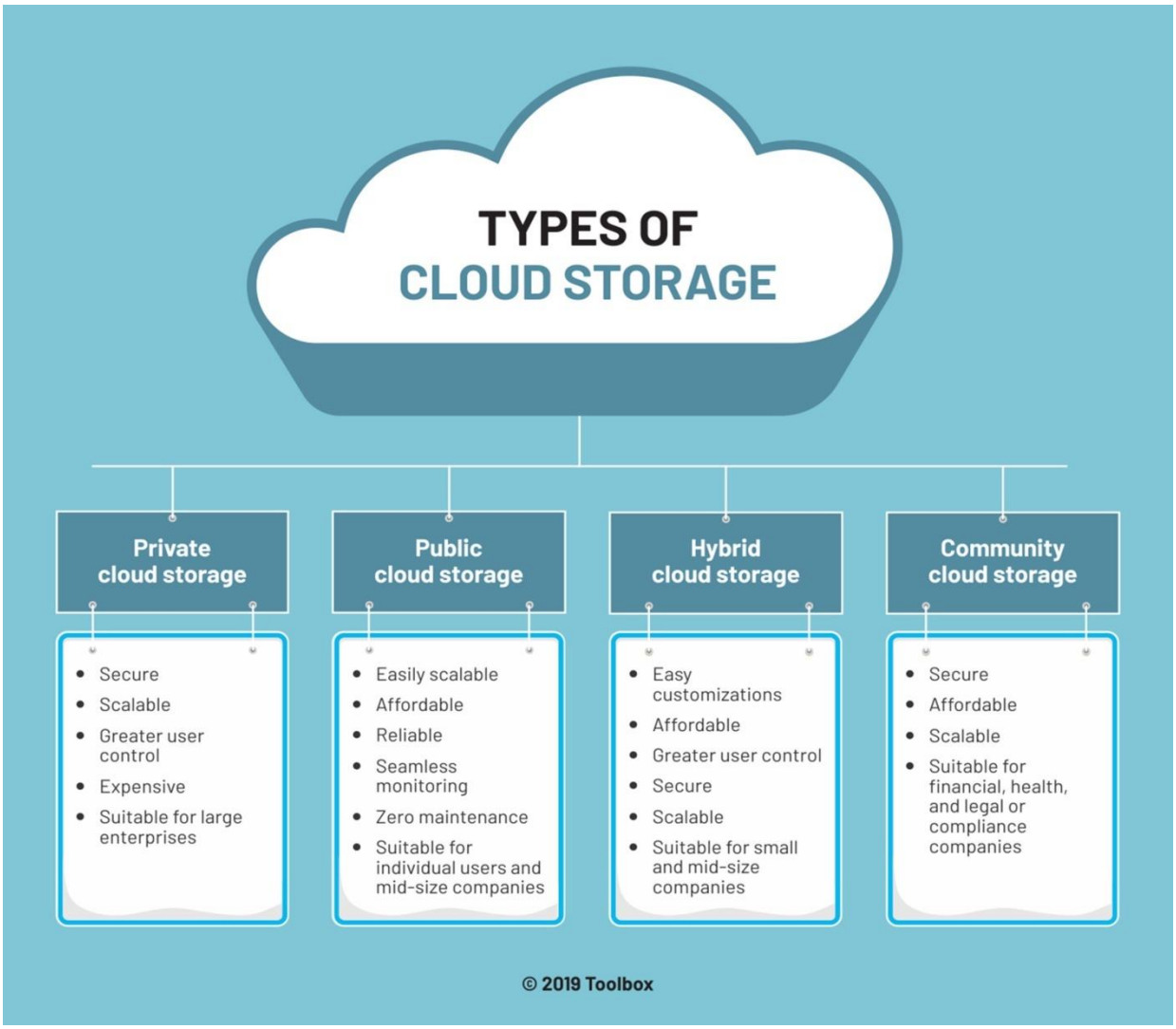
HOW DOES CLOUD STORAGE WORK:



Cloud storage works as a virtual data center. It offers end users and applications virtual storage infrastructure that can be scaled to the application's requirements. It generally operates via a web-based API implemented remotely through its interaction with in-house cloud storage infrastructure.

Cloud storage includes at least one data server to which a user can connect via the internet. The user sends files to the data server, which forwards the message to multiple servers, manually or in an automated manner, over the internet. The stored data can then be accessed via a web-based interface.

To ensure the constant availability of data, cloud storage systems involve large numbers of data servers. Therefore, if a server requires maintenance or fails, the user can be assured that the data has been moved elsewhere to ensure availability.



Benefits of cloud storage

- **Flexibility and ease of access**: Cloud storage means that your data is not tied down to any one location. Various stakeholders can access assets stored on the cloud from a location and device of their choice without any download or installation hassles.
- **Remote management support**: Cloud storage also paves the way for remote management either by internal IT teams or by managed service providers (MSPs). They can troubleshoot without being present on-site, speeding up issue resolution.
- **Fast scalability**: A major benefit of cloud storage is that you can provision new resources with only a few clicks without the need for any additional infrastructure. When faced with an unprecedented increase in data volumes, this feature aids business continuity.
- **Redundancy for backup**: Data redundancy (i.e., replicating the same data in multiple locations) is essential for an effective backup mechanism. The cloud ensures your data is kept secure in a remote location in case of a natural disaster, accident, or cyberattack.
- **Long-term cost savings**: In the long-term, cloud storage can save you significantly in the costs of hardware equipment, storage facilities, power supply, and personnel, which are sure to multiply as your organization grows.

Challenges of cloud storage

While there are undeniable advantages of adopting cloud storage, there are a few cons to remember as well. By navigating these cons or challenges, you can arrive at a pragmatic cloud storage strategy that maximizes its benefits.

- ✓ **Risk of vendor lock-in**: If all your data is stored in a single public cloud platform, there's a risk of vendor lock-in and potential inflexibilities. Address this with a hybrid or multi-cloud blueprint where there is sufficient interoperability between environments.
- ✓ **Security issues around multi-tenancy**: Public cloud environments are shared by multiple tenants, which can multiply your security vulnerabilities. You can prevent this through cloud data protection and by leveraging the private cloud for sensitive data.
- ✓ **Fragmentation of IT landscape**: Unplanned cloud storage adoption can cause your IT landscape to become fragmented over time. That's why you need a detailed strategic blueprint outlining your short, mid, and long-term cloud roadmap.
- ✓ **Outage and downtime risk**: Cloud platforms managed by external providers could suffer from an outage, rendering the data and applications stored in these environments inaccessible. Service level agreements should specify downtime metrics, and you need additional redundancy for your most critical data.
- ✓ **Short-term budget overruns**: Cloud cost worries are extremely common, where data storage and storage processes occupy more space than estimated. A cloud resource management tool can help address this, giving you visibility and control.

Selecting the right cloud storage provider

Let's look at the most critical aspects businesses need to consider when selecting a cloud storage provider.

- ○ **Storage space:** The amount of data a business processes determines the requirement for storage space. A small organization (around 250 employees) could opt for public cloud storage services, which offer employees storage space of over 15 GB each. It is recommended to compare various public cloud storage pricing plans before signing the deal.
- ○ **Maintenance & uptime:** Cloud servers need to be maintained to make sure the data stored is secure. However, downtimes and network failures can occur anytime. Therefore, understanding the maintenance and uptime needed by cloud service providers is essential. Organizations should ask their chosen cloud service providers to demonstrate their downtime plans and run checks before buying any cloud solution.
- ○ **Security:** If data is compromised, then cloud storage comes in handy as a useful backup. There is no guarantee, however, that cloud storage providers are safe from security threats. Understanding the security measures in place at the cloud storage provider is important. Two main factors need to be considered for

security: the physical security of the cloud solution provider's servers and the level of encryption applied to the data stored.

- **Speed:** The speed of downloads from the cloud has a major impact on businesses and their ability to process critical data. If cloud storage providers place a cap on the download speed, retrieving data and running applications will take longer. Therefore, organizations need to gauge the cloud storage download speeds of a provider before buying any storage space.

Internet of Things (IoT) Enabling Technologies

IoT(internet of things) enabling technologies are

1. Wireless Sensor Network
2. Cloud Computing
3. Big Data Analytics
4. Communications Protocols
5. Embedded System

1. Wireless Sensor Network(WSN) :

A **WSN** comprises distributed devices with sensors which are used to monitor the environmental and physical conditions. A **wireless sensor network** consists of end nodes, routers and coordinators. End nodes have several sensors attached to them where the data is passed to a coordinator with the help of routers. The coordinator also acts as the gateway that connects WSN to the internet.

Example –

- Weather monitoring system
- Indoor air quality monitoring system
- Soil moisture monitoring system
- Surveillance system
- Health monitoring system

2. Cloud Computing :

It provides us the means by which we can access applications as utilities over the internet. Cloud means something which is present in remote locations.

With Cloud computing, users can access any resources from anywhere like databases, webservers, storage, any device, and any software over the internet.

Characteristics –

1. Broad network access
2. On demand self-services
3. Rapid scalability
4. Measured service
5. Pay-per-use

Provides different services, such as –

- **IaaS (**Infrastructure as a service**)**
  Infrastructure as a service provides online services such as physical machines, virtual machines, servers, networking, storage and data center space on a pay per use basis. Major IaaS providers are Google Compute Engine, Amazon Web Services and Microsoft Azure etc.
  Ex : Web Hosting, Virtual Machine etc.
- **PaaS (**Platform as a service**)**
  Provides a cloud-based environment with a very thing required to support the complete life cycle of building and delivering West web based (cloud) applications – without the cost and complexity of buying and managing underlying hardware, software provisioning and hosting. Computing platforms such as hardware, operating systems and libraries etc. Basically, it provides a platform to develop applications.
  Ex : App Cloud, Google app engine
- **SaaS (**Software as a service**)**
  It is a way of delivering applications over the internet as a service. Instead of installing and maintaining software, you simply access it via the internet, freeing yourself from complex software and hardware management.
  SaaS Applications are sometimes called web-based software on demand software or hosted software.
  SaaS applications run on a SaaS provider's service and they manage security availability and performance.
  Ex : Google Docs, Gmail, office etc.

3. Big Data Analytics :

It refers to the method of studying massive volumes of data or big data. Collection of data whose volume, velocity or variety is simply too massive and tough to store, control, process and examine the data using traditional databases.

Big data is gathered from a variety of sources including social network videos, digital images, sensors and sales transaction records.

Several steps involved in analyzing big data –

1. Data cleaning
2. Munging
3. Processing
4. Visualization

Examples –

- Bank transactions
- Data generated by IoT systems for location and tracking of vehicles
- E-commerce and in Big-Basket
- Health and fitness data generated by IoT system such as a fitness bands

4. Communications Protocols :

They are the backbone of IoT systems and enable network connectivity and linking to applications. Communication protocols allow devices to exchange data over the network. Multiple protocols often describe different aspects of a single communication. A group of protocols designed to work together is known as a protocol suite; when implemented in software they are a protocol stack.

They are used in

1. Data encoding
2. Addressing schemes

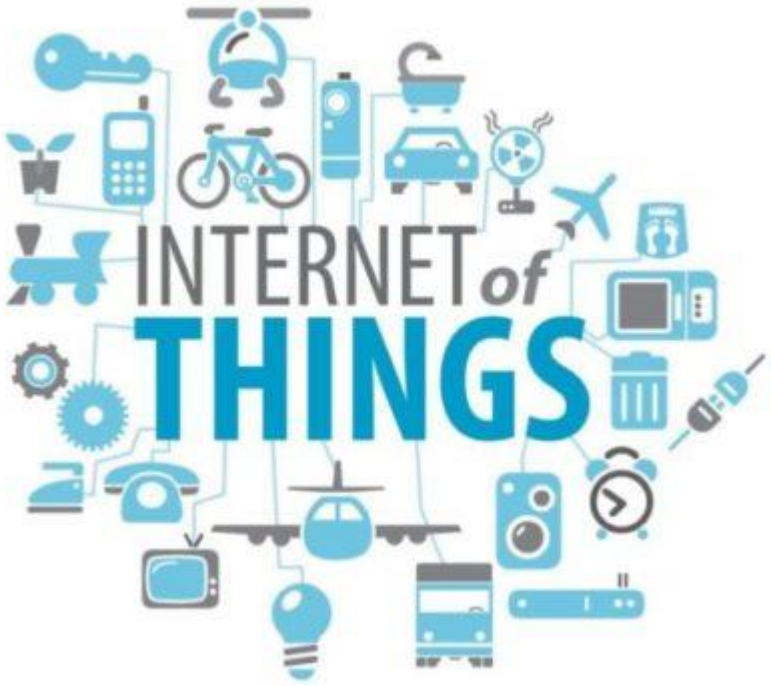
5. Embedded Systems :

It is a combination of hardware and software used to perform special tasks.
It includes microcontroller and microprocessor memory, networking units (Ethernet Wi-Fi adapters), input output units (display keyword etc.) and storage devices (flash memory).
It collects the data and sends it to the internet.
Embedded systems used in
Examples –

1. Digital camera
2. DVD player, music player
3. Industrial robots
4. Wireless Routers etc.

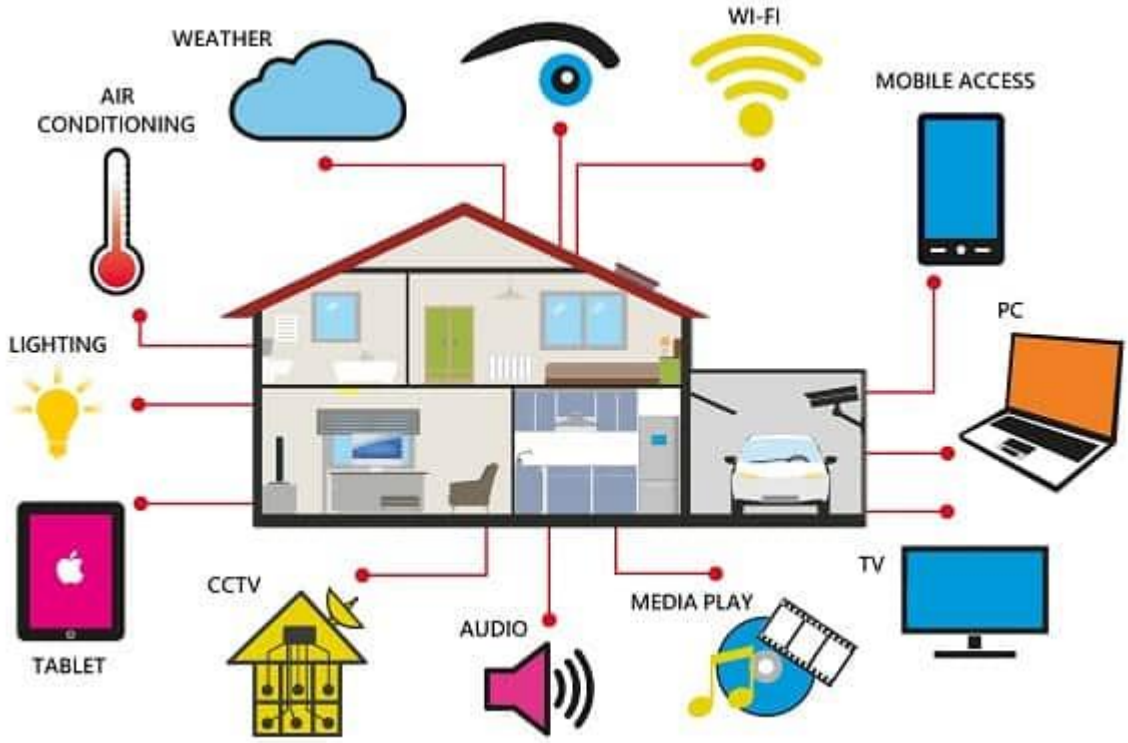
Applications of Internet of Things (IoT)

Last Modified: October 9, 2021 by Rajiv 2 Comments



Internet, one of the most significant inventions of all time helped us to connect with people using computers and Smartphone. The new generation of network IoT (internet of things), connects things with the ability to sense, control and communicate. Find out what are the applications of Internet of Things?

1. Smart Home and Office



Source: Visioforce Automation

Smart home applications with the use of smart sensors are becoming popular now. Any smart device can be configured and connected to the internet and control using simple mobile application.

Smart Door access control system

Smart locks and door access systems are one of the most popular and cost effective solutions of Internet of Things. Smart locks are easy to implement and control using a web interface or Smartphone application.

Integration with RDIF tags, smart door accessing systems can be securely implemented. Users can grant access to the doors using mobile app and lock again once the person leaves the premises.

Example: a person wants to enter your house while you are not around, you will be able to open the door for that person using Smartphone application.

Smart lighting for home and office

Smart lighting is one the attractive smart home application using internet of things. In addition to energy saving, it also enables us to manage effectively. Light ambience can be changed using smart hub devices or smart phone app.

Smart lighting can be configured to respond to voice commands and motion detectors / proximity sensors. These sensors will activate the light when someone enters the room or leaving the room. Moreover, it can be configured to turn on when the ambient light is below certain threshold (turn on during sun light is low).

Automated Gate and garage

Using smart sensor technology and internet of things, gates and garages can be controlled (operated) conveniently. Once you are about to enter the house or after leaving the premises, you may open or close the gate using mobile devices.

Smart thermostats and humidity controllers

Smart thermostats are cost effective and convenient smart home solutions which can be controlled using an internet connection and smart hub device (or using Smartphone app).

Common sensors for home/office automation:

- Motion / proximity sensors
- Voice controlled sensor
- Light sensor
- Temperature and humidity sensors
- Smoke/fire sensor
- Precipitation sensor

Traffic Management

Analyzing traffic over a period of time gives an insight of possible trends and pattern that could occur during peak hours. It will help to inform commuters to take alternative routes to avoid congestion and delay.

Smart lighting on streets

Smart lighting is an effective solution to save energy in the cities. Smart sensors can detect presents of people or vehicles in the proximity and increase light intensity when someone pass by.

Once the person or vehicle is away from that area, smart light will automatically reduce light intensity to save energy. During emergency situations, maximum light intensity will be activated to support recovery activities.

Since the smart lighting systems are connected to control and monitoring network, any faulty light units will be automatically reported and necessary maintenance will be initiated.

Pollution monitoring and reporting

Increasing air pollution is one of the challenges we are facing in every growing cities. In order to solve this issue, smart sensors are deployed across the cities to continuously monitor any changes.

Some of the common sensors are temperature, air quality (like CO2 level, haze, and smoke), moisture etc… Interconnected smart sensors collects data, sends these data to the monitoring stations and initiates warning messages during bad air quality detection.

Smart Parking Solutions

Smart sensors installed on parking area are collecting information about availability of parking slots and updating it to the database real time. Once the spot is occupied, it will be updated without any delay.

Service providers and customers can plan and manage parking issues with the use of smart parking solutions.

Water / waste management

Populations in cities are increasing every year, based on statistics this trend will grow in coming years. Increase in population contributes to increase in wastes as well.

Many cities are adapting recycling of water using water treatment units. With IoT system, the amount of waste water, consumption in a geographical area and trend of waste produced can be analyzed effectively.

IoT and smart sensor technology enables us to manage this issue efficiently. With smart waste management system, authorities will be able to predict the amount of waste produced in a particular location, how to process properly, trigger clearance of waste and analyze data for future planning etc.…

Example: smart sensors implemented on trash bins can send alerts to the waste management system once the bin is full (or reached threshold limit). If the waste quantity in the bin is low, it will not be emptied.

Service Oriented Architecture (SOA)

A Service-Oriented Architecture or SOA is a design pattern which is designed to build distributed systems that deliver services to other applications through the protocol. It is only a concept and not limited to any programming language or platform.

What is Service?

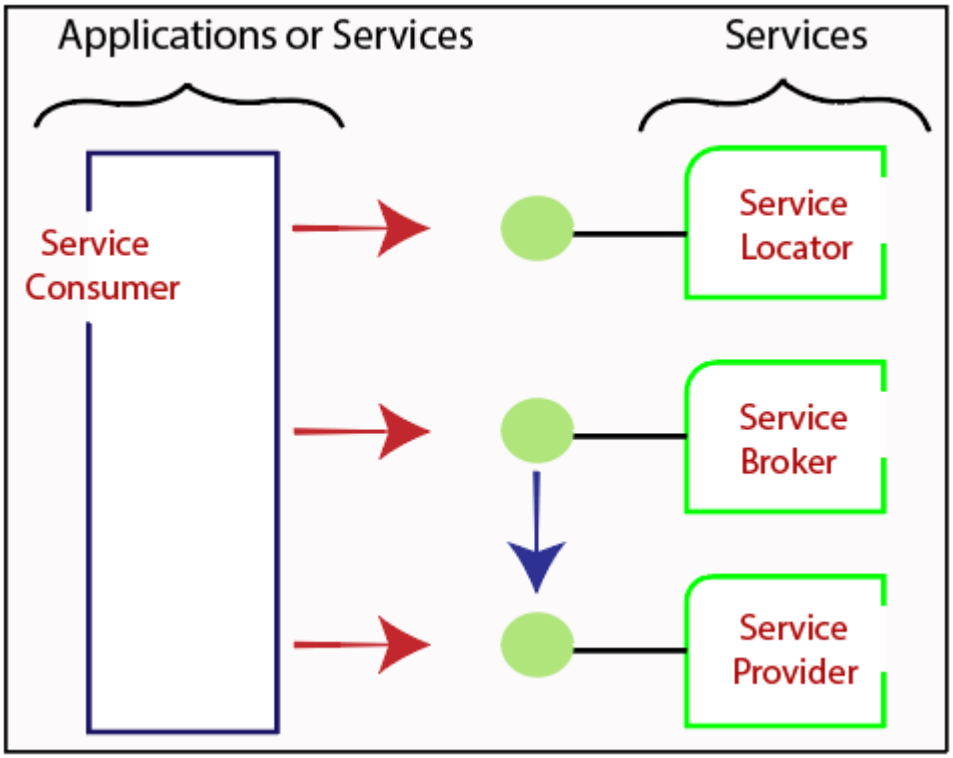
A service is a well-defined, self-contained function that represents a unit of functionality. A service can exchange information from another service. It is not dependent on the state of another service. It uses a loosely coupled, message-based communication model to communicate with applications and other services.

Service Connections

The figure given below illustrates the service-oriented architecture. Service consumer sends a service request to the service provider, and the service provider sends the service response to the service consumer. The service connection is understandable to both the service consumer and service provider.

Service-Oriented Terminologies

Let's see some important service-oriented terminologies:



- **Services** - The services are the logical entities defined by one or more published interfaces.
- **Service provider** - It is a software entity that implements a service specification.
- **Service consumer** - It can be called as a requestor or client that calls a service provider. A service consumer can be another service or an end-user application.
- **Service locator** - It is a service provider that acts as a registry. It is responsible for examining service provider interfaces and service locations.
- **Service broker** - It is a service provider that pass service requests to one or more additional service providers.

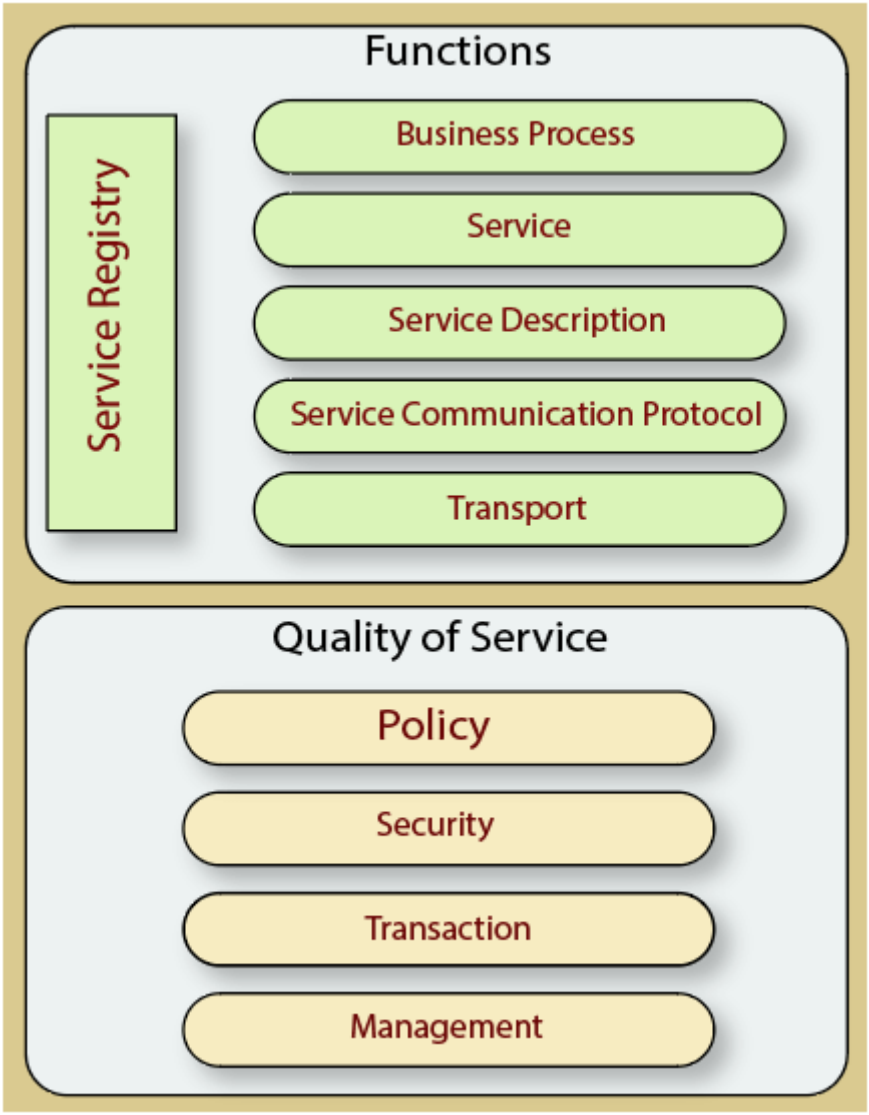
Characteristics of SOA

The services have the following characteristics:

- They are loosely coupled.
- They support interoperability.
- They are location-transparent
- They are self-contained.

Components of service-oriented architecture

The service-oriented architecture stack can be categorized into two parts - functional aspects and quality of service aspects.



Functional aspects

The functional aspect contains:

- Transport - It transports the service requests from the service consumer to the service provider and service responses from the service provider to the service consumer.
- Service Communication Protocol - It allows the service provider and the service consumer to communicate with each other.
- Service Description - It describes the service and data required to invoke it.

- Service - It is an actual service.
- Business Process - It represents the group of services called in a particular sequence associated with the particular rules to meet the business requirements.
- Service Registry - It contains the description of data which is used by service providers to publish their services.

Quality of Service aspects

The quality of service aspects contains:

- Policy - It represents the set of protocols according to which a service provider make and provide the services to consumers.
- Security - It represents the set of protocols required for identification and authorization.
- Transaction - It provides the surety of consistent result. This means, if we use the group of services to complete a business function, either all must complete or none of the complete.
- Management - It defines the set of attributes used to manage the services.

Web Services in Cloud Computing

The Internet is the worldwide connectivity of hundreds of thousands of computers belonging to many different networks.

A web service is a standardized method for propagating messages between client and server applications on the World Wide Web. A web service is a software module that aims to accomplish a specific set of tasks. Web services can be found and implemented over a network in cloud computing.

The web service would be able to provide the functionality to the client that invoked the web service.

A web service is a set of open protocols and standards that allow data exchange between different applications or systems. Web services can be used by software programs written in different programming languages and on different platforms to exchange data through computer networks such as the Internet. In the same way, communication on a computer can be inter-processed.

Any software, application, or cloud technology that uses a standardized Web protocol (**HTTP or HTTPS**) to connect, interoperate, and exchange data messages over the Internet-usually XML (Extensible Markup Language) is considered a Web service. Is.

Web services allow programs developed in different languages to be connected between a client and a server by exchanging data over a web service. A client invokes a web service by submitting an XML request, to which the service responds with an XML response.

- Web services functions
- It is possible to access it via the Internet or intranet network.
- XML messaging protocol that is standardized.
- Operating system or programming language independent.
- Using the XML standard is self-describing.

A simple location approach can be used to detect this.

Web Service Components

XML and HTTP is the most fundamental web service platform. All typical web services use the following components:

1. SOAP (Simple Object Access Protocol)

SOAP stands for "Simple Object Access Protocol". It is a transport-independent messaging protocol. SOAP is built on sending XML data in the form of SOAP messages. A document known as an XML document is attached to each message.

Only the structure of an XML document, not the content, follows a pattern. The great thing about web services and SOAP is that everything is sent through HTTP, the standard web protocol.

Every SOAP document requires a root element known as an element. In an XML document, the root element is the first element.

The "envelope" is divided into two halves. The header comes first, followed by the body. Routing data, or information that directs the XML document to which client it should be sent, is contained in the header. The real message will be in the body.

2. UDDI (Universal Description, Search, and Integration)

UDDI is a standard for specifying, publishing and searching online service providers. It provides a specification that helps in hosting the data through web services. UDDI provides a repository where WSDL files can be hosted so that a client application can search the WSDL file to learn about the various actions provided by the web service. As a result, the client application will have full access to UDDI, which acts as the database for all WSDL files.

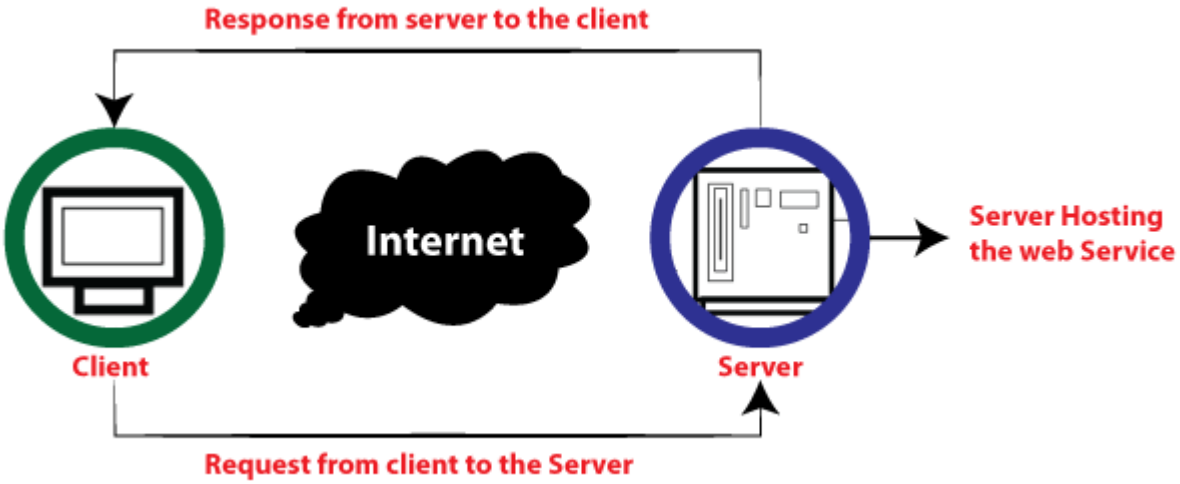
The UDDI Registry will keep the information needed for online services, such as a telephone directory containing the name, address, and phone number of a certain person so that client applications can find where it is.

3. WSDL (Web Services Description Language)

The client implementing the web service must be aware of the location of the web service. If a web service cannot be found, it cannot be used. Second, the client application must understand what the web service does to implement the correct web service. WSDL, or Web Service Description Language, is used to accomplish this. A WSDL file is another XML-based file that describes what a web service does with a client application. The client application will understand where the web service is located and how to access it using the WSDL document.

How does web service work?

The diagram shows a simplified version of how a web service would function. The client will use requests to send a sequence of web service calls to the server hosting the actual web service.



Remote procedure calls are used to perform these requests. The calls to the methods hosted by the respective web service are known as Remote Procedure Calls (RPC). Example: Flipkart provides a web service that displays the prices of items offered on Flipkart.com. The front end or presentation layer can be written in .NET or Java, but the web service can be communicated using a programming language.

The data exchanged between the client and the server, XML, is the most important part of web service design. XML (Extensible Markup Language) is a simple, intermediate language understood by various programming languages. It is the equivalent of HTML.

As a result, when programs communicate with each other, they use XML It forms a common platform for applications written in different programming languages to communicate with each other.

Web services employ SOAP (Simple Object Access Protocol) to transmit XML data between applications. The data is sent using standard HTTP. A SOAP message is data sent from a web service to an application. An XML document is all that is contained in a SOAP message. The

client application that calls the web service can be built in any programming language as the content is written in XML.

Features of Web Service

Web services have the following characteristics:

(a) XML-based: A web service's information representation and record transport layers employ XML. There is no need for networking, operating system, or platform bindings when using XML. At the mid-level, web offering-based applications are highly interactive.

(b) Loosely Coupled: The subscriber of an Internet service provider may not necessarily be directly connected to that service provider. The user interface for a web service provider may change over time without affecting the user's ability to interact with the service provider. A strongly coupled system means that the decisions of the mentor and the server are inextricably linked, indicating that if one interface changes, the other must be updated.

A loosely connected architecture makes software systems more manageable and easier to integrate between different structures.

(c) Ability to be synchronous or asynchronous: Synchronicity refers to the client's connection to the execution of the function. Asynchronous operations allow the client to initiate a task and continue with other tasks. The client is blocked, and the client must wait for the service to complete its operation before continuing in synchronous invocation.

Asynchronous clients get their results later, but synchronous clients get their effect immediately when the service is complete. The ability to enable loosely connected systems requires asynchronous capabilities.

(d) Coarse Grain: Object-oriented systems, such as Java, make their services available differently. At the corporate level, an operation is too great for a character technique to be useful. Building a Java application from the ground up requires the development of several granular strategies, which are then combined into a coarse grain provider that is consumed by the buyer or service.

Corporations should be coarse-grained, as should the interfaces they expose. Building web services is an easy way to define coarse-grained services that have access to substantial business enterprise logic.

(e) Supports remote procedural calls: Consumers can use XML-based protocols to call procedures, functions, and methods on remote objects that use web services. A web service must support the input and output framework of the remote system.

Enterprise-wide component development Over the years, JavaBeans (EJBs) and .NET components have become more prevalent in architectural and enterprise deployments. Several RPC techniques are used to both allocate and access them.

A web function can support RPC by providing its services, similar to a traditional role, or translating incoming invocations into an EJB or .NET component invocation.

(f) Supports document exchanges: One of the most attractive features of XML for communicating with data and complex entities.

Virtualization in Cloud Computing

Virtualization is the "creation of a virtual (rather than actual) version of something, such as a server, a desktop, a storage device, an operating system or network resources".

In other words, Virtualization is a technique, which allows to share a single physical instance of a resource or an application among multiple customers and organizations. It does by assigning a logical name to a physical storage and providing a pointer to that physical resource when demanded.

What is the concept behind the Virtualization?

Creation of a virtual machine over existing operating system and hardware is known as Hardware Virtualization. A Virtual machine provides an environment that is logically separated from the underlying hardware.

The machine on which the virtual machine is going to create is known as **Host Machine** and that virtual machine is referred as a **Guest Machine**

Types of Virtualization:

1. Hardware Virtualization.
2. Operating system Virtualization.
3. Server Virtualization.
4. Storage Virtualization.

1) Hardware Virtualization:

When the virtual machine software or virtual machine manager *(VMM) is directly installed on the hardware system* is known as hardware virtualization.

The main job of hypervisor is to control and monitoring the processor, memory and other hardware resources.

After virtualization of hardware system we can install different operating system on it and run different applications on those OS.

Usage:

Hardware virtualization is mainly done for the server platforms, because controlling virtual machines is much easier than controlling a physical server.

2) Operating System Virtualization:

When the virtual machine software or virtual machine manager *(VMM) is installed on the Host operating system* instead of directly on the hardware system is known as operating system virtualization.

Usage:

Operating System Virtualization is mainly used for testing the applications on different platforms of OS.

3) Server Virtualization:

When the virtual machine software or virtual machine manager *(VMM) is directly installed on the Server system* is known as server virtualization.

Usage:

Server virtualization is done because a single physical server can be divided into multiple servers on the demand basis and for balancing the load.

4) Storage Virtualization:

Storage virtualization is the *process of grouping the physical storage from multiple network storage devices so that it looks like a single storage device*.

Storage virtualization is also implemented by using software applications.

Usage:

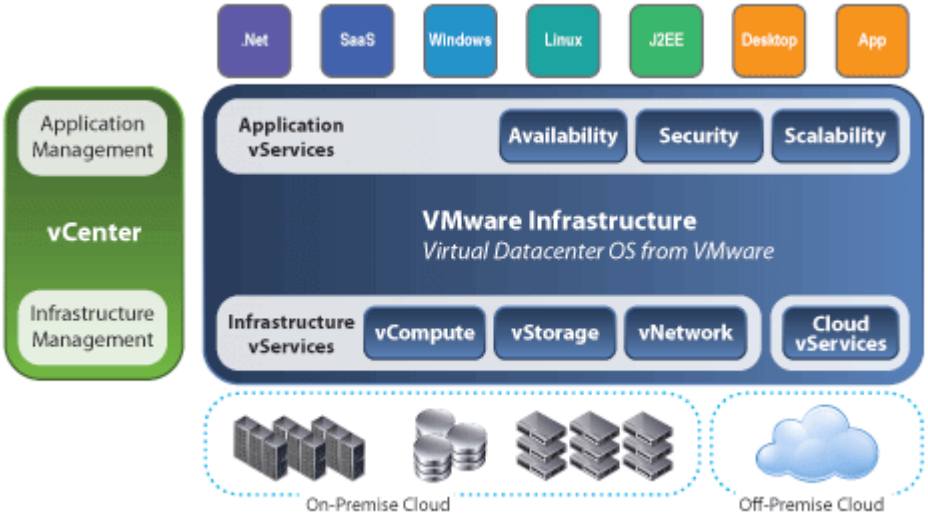
Storage virtualization is mainly done for back-up and recovery purposes.

How does virtualization work in cloud computing?

Virtualization plays a very important role in the cloud computing technology, normally in the cloud computing, users share the data present in the clouds like application etc, but actually with the help of virtualization users shares the Infrastructure.

The **main usage of Virtualization Technology** is to provide the applications with the standard versions to their cloud users, suppose if the next version of that application is released, then cloud provider has to provide the latest version to their cloud users and practically it is possible because it is more expensive.

To overcome this problem we use basically virtualization technology, By using virtualization, all severs and the software application which are required by other cloud providers are maintained by the third party people, and the cloud providers has to pay the money on monthly or annual basis.



Conclusion

Mainly Virtualization means, running multiple operating systems on a single machine but sharing all the hardware resources. And it helps us to provide the pool of IT resources so that we can share these IT resources in order get benefits in the business.

What is Emulation?

In computing, the emulator is a hardware or software that enables one device (named **Host**) to function like other systems (named **Guest**). It is a perfect way to execute the hardware and software in any system. Emulation brings greater overhead, but it also has its benefits. It is relatively inexpensive, easily accessible and allows us to run programs that have become redundant in the available system.

An emulator changes the **CPU** instructions required for the architecture and executes it on another architecture successfully. The emulation systems could be accessed remotely by anyone and are very simpler to use. Without affecting the underlying OS, it is an excellent capacity for embedded and OS development. Without considering the host's capabilities, emulation will usually manage the size of the **design under test (DUT)**.

IMPLEMENTATION LEVELS OF VIRTUALIZATION IN CLOUD COMPUTING

It is not simple to set up virtualization. Your computer runs on an operating system that gets configured on some particular hardware. It is not feasible or easy to run a different operating system using the same hardware.

To do this, you will need a hypervisor. Now, what is the role of the hypervisor? It is a bridge between the hardware and the virtual operating system, which allows smooth functioning.

Talking of the Implementation levels of virtualization in cloud computing, there are a total of five levels that are commonly used. Let us now look closely at each of these levels of virtualization implementation in cloud computing.

1.) Instruction Set Architecture Level (ISA)

ISA virtualization can work through ISA emulation. This is used to run many legacy codes that were written for a different configuration of hardware. These codes run on any virtual machine using the ISA. With this, a binary code that originally needed some additional layers to run is now capable of running on the x86 machines. It can also be tweaked to run on the x64 machine. With ISA, it is possible to make the virtual machine hardware agnostic.

For the basic emulation, an interpreter is needed, which interprets the source code and then converts it into a hardware format that can be read. This then allows processing. This is one of the five implementation levels of virtualization in cloud computing.

2.) Hardware Abstraction Level (HAL)

True to its name HAL lets the virtualization perform at the level of the hardware. This makes use of a hypervisor which is used for functioning. At this level, the virtual machine is formed, and this manages the hardware using the process of virtualization. It allows the virtualization of each of the hardware components, which could be the input-output device, the memory, the processor, etc.

Multiple users will not be able to use the same hardware and also use multiple virtualization instances at the very same time. This is mostly used in the cloud-based infrastructure.

3.) Operating System Level

At the level of the operating system, the virtualization model is capable of creating a layer that is abstract between the operating system and the application. This is an isolated container

that is on the operating system and the physical server, which makes use of the software and hardware. Each of these then functions in the form of a server.

When there are several users, and no one wants to share the hardware, then this is where the virtualization level is used. Every user will get his virtual environment using a virtual hardware resource that is dedicated. In this way, there is no question of any conflict.

4.) Library Level

The operating system is cumbersome, and this is when the applications make use of the API that is from the libraries at a user level. These APIs are documented well, and this is why the library virtualization level is preferred in these scenarios. API hooks make it possible as it controls the link of communication from the application to the system.

5.) Application Level

The application-level virtualization is used when there is a desire to virtualize only one application and is the last of the implementation levels of virtualization in cloud computing. One does not need to virtualize the entire environment of the platform.

This is generally used when you run virtual machines that use high-level languages. The application will sit above the virtualization layer, which in turn sits on the application program.

It lets the high-level language programs compiled to be used in the application level of the virtual machine run seamlessly.

CONCLUSION

There are in total of five implementation levels of virtualization in cloud computing. However, every enterprise may not use each one of the different levels of virtualization implementation in cloud computing. The level used is based on the working of the company and also on its preference for the level of virtualization. The company will use the virtual machine to develop and test across multiple platforms. Cloud-based applications are on the rise, which makes virtualization a must-have thing for enterprises all over the world.

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 operat-ing 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 virtualiza-tion 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.

1. 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 physi-cal 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 functional-ity, 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. 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 hyper-visor is smaller than that of a monolithic hypervisor. Essentially, a hypervisor must be able to convert physical devices into virtual resources dedicated for the deployed VM to use.

1.1 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 [7]. 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 [62] and Oracle VM [42].

The core components of a Xen system are the hypervisor, kernel, and applications. The organi-zation 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

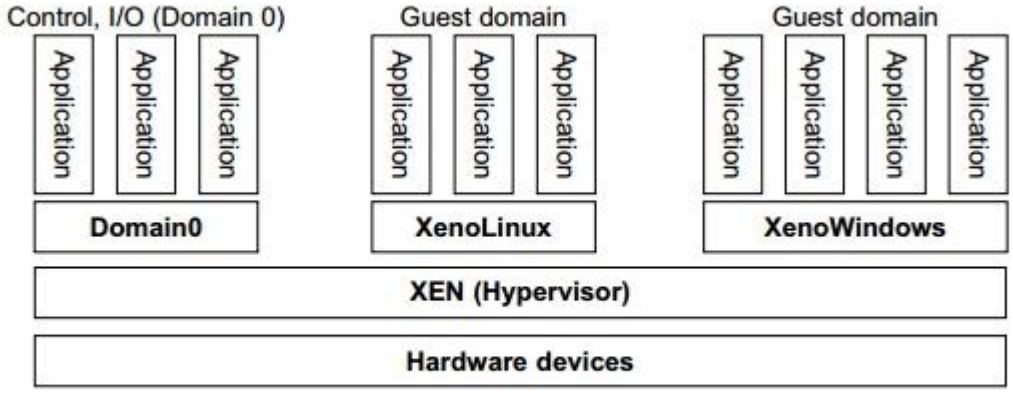


FIGURE 3.5

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

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, con-figuration changes are made, software is installed, and patches are applied. In such an environment, the VM state is akin to a tree: At any point, execution can go

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

2. Binary Translation with Full Virtualization

Depending on implementation technologies, hardware virtualization can be classified into two cate-gories: 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 virtuali-zation software layer is built between the host OS and guest OS. These two classes of VM architec-ture are introduced next.

2.1 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.

2.2 Binary Translation of Guest OS Requests Using a VMM

This approach was implemented by VMware and many other software companies. As shown in Figure 3.6, VMware puts the VMM at Ring 0 and the guest OS at Ring 1. The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions. The method used in this emulation is called binary translation. Therefore, full vir-tualization combines binary translation and direct execution. The guest OS is completely decoupled from the underlying hardware. Consequently, the guest OS is unaware that it is being virtualized.

The performance of full virtualization may not be ideal, because it involves binary translation which is rather time-consuming. In particular, the full virtualization of I/O-intensive applications is a really a big challenge. Binary translation employs a code cache to

store translated hot instructions to improve performance, but it increases the cost of memory usage. At the time of this writing, the performance of full virtualization on the x86 architecture is typically 80 percent to 97 percent that of the host machine.

2.3 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

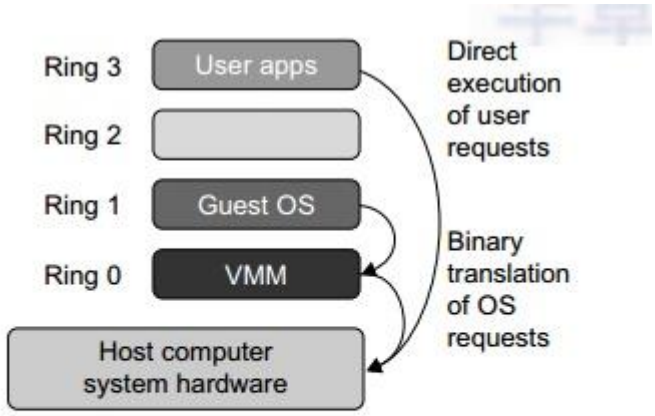


FIGURE 3.6

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.

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.

3. Para-Virtualization with Compiler Support

Para-virtualization needs to modify the guest operating systems. A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications. Performance degradation is a critical issue of a virtualized system. No one wants to use a VM if it is much slower than using a physical machine. The virtualization layer can be inserted at different positions in a machine soft-ware 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 paravirtualized 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.

3.1 Para-Virtualization Architecture

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

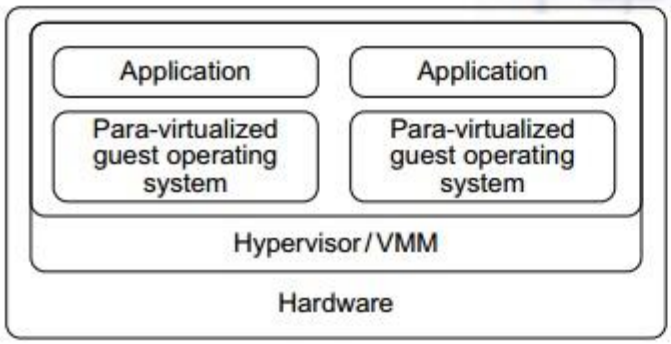


FIGURE 3.7

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.)

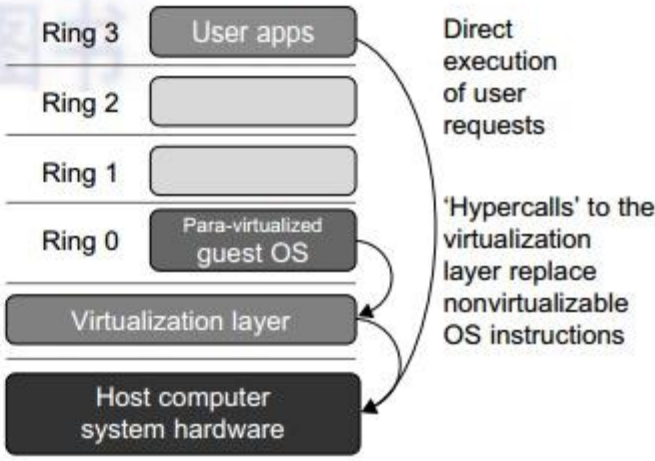


FIGURE 3.8

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

(*Courtesy of VMWare [71]*)

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.

3.2 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.

3.3 Para-Virtualization with Compiler Support

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

time. The guest OS kernel is modified to replace the privileged and sensitive instructions with hypercalls to the hypervi-sor or VMM. Xen assumes such a para-virtualization architecture.

The guest OS running in a guest domain may run at Ring 1 instead of at Ring 0. This implies that the guest OS may not be able to execute some privileged and sensitive instructions. The privileged instructions are implemented by hypercalls to the hypervisor. After replacing the instructions with hypercalls, the modified guest OS emulates the behavior of the original guest OS. On an UNIX system, a system call involves an interrupt or service routine. The hypercalls apply a dedicated service routine in Xen.

VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

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

1. Hardware Support for Virtualization

Modern operating systems and processors permit multiple processes to run simultaneously. If there is no protection mechanism in a processor, all instructions from different processes will access the hardware directly and cause a system crash. Therefore, all processors have at least two modes, user mode and supervisor mode, to ensure controlled access of critical hardware. Instructions running in supervisor mode are called privileged instructions. Other instructions are unprivileged instructions. In a virtualized environment, it is more difficult to make OSes and applications run correctly because there are more layers in the machine stack. Example 3.4 discusses Intel’s hardware support approach.

At the time of this writing, 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 Workstation assumes the host-based virtualization. Xen is a hypervisor for use in IA-32, x86-64, Itanium, and PowerPC 970 hosts. Actually, Xen modifies Linux as the lowest and most privileged layer, or a hypervisor.

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

Example 3.4 Hardware Support for Virtualization in the Intel x86 Processor

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

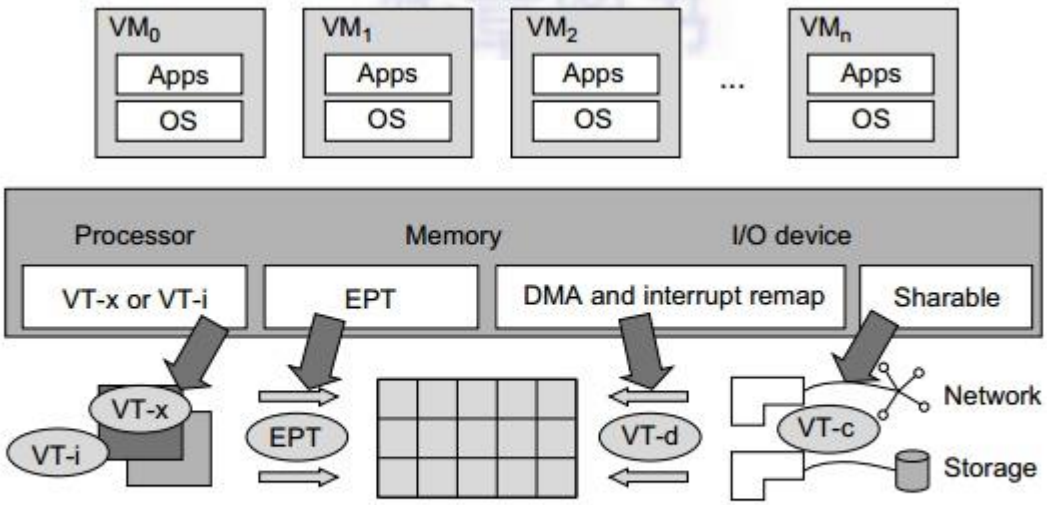


FIGURE 3.10

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

2. 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 exe-cuted, 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 instruc-tions 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 nor-mally. 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.

2.1 Hardware-Assisted CPU Virtualization

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

Although x86 processors are not virtualizable primarily, great effort is taken to virtualize them. They are used widely in comparing RISC processors that the bulk of x86-based legacy systems cannot discard easily. Virtuali-zation of x86 processors is detailed in the following sections. Intel's VT-x technology is an example of hardware-assisted virtualization, as shown

in Figure 3.11. Intel calls the privilege level of x86 processors the VMX Root Mode. In order to control the start and stop of a VM and allocate a memory page to maintain the

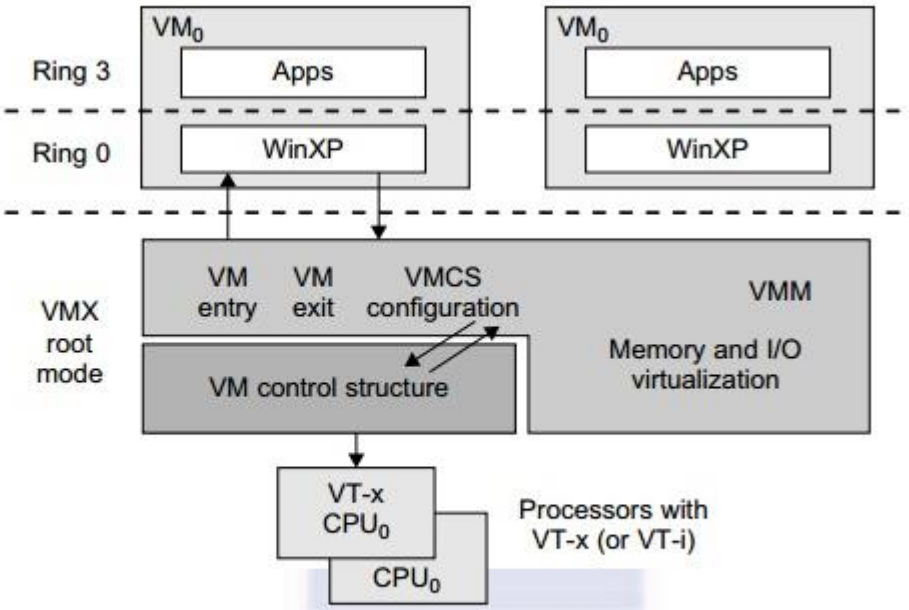


FIGURE 3.11

Intel hardware-assisted CPU virtualization.

CPU state for VMs, a set of additional instructions is added. At the time of this writing, Xen, VMware, and the Microsoft Virtual PC all implement their hypervisors by using the VT-x technology.

Generally, hardware-assisted virtualization should have high efficiency. However, since the transition from the hypervisor to the guest OS incurs high overhead switches between processor modes, it sometimes cannot outperform binary translation. Hence, virtualization systems such as VMware now use a hybrid approach, in which a few tasks are offloaded to the hardware but the rest is still done in software. In addition, para-virtualization and hardware-assisted virtualization can be combined to improve the performance further.

3. Memory Virtualization

Virtual memory virtualization is similar to the virtual memory support provided by modern operat-ing 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 shows the two-level memory mapping procedure.

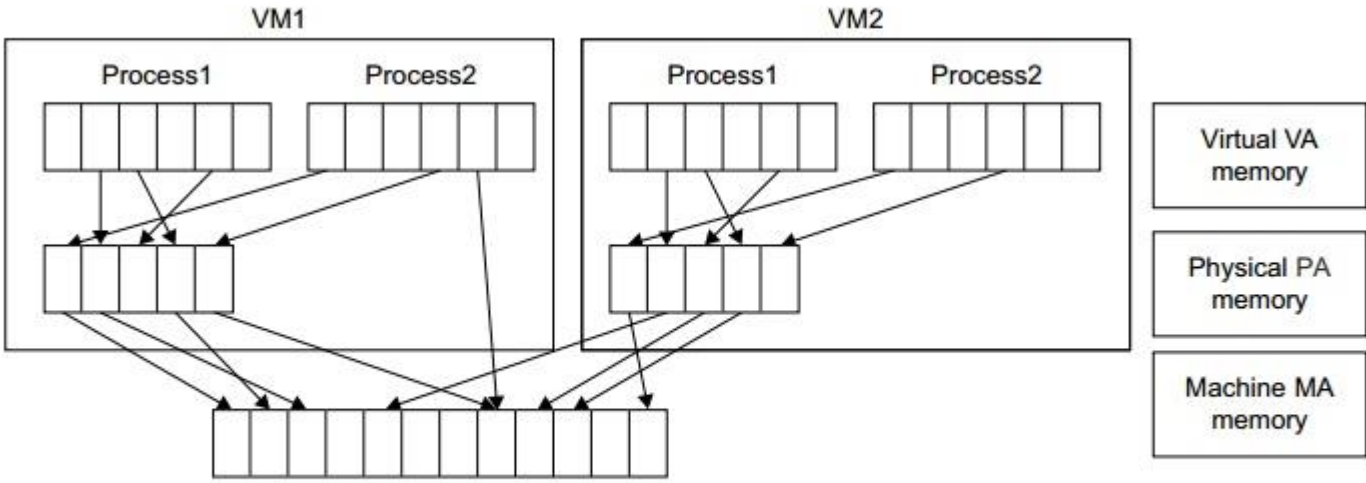


FIGURE 3.12

Two-level memory mapping procedure.

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

Example 3.6 Extended Page Table by Intel for Memory Virtualization

Since the efficiency of the software shadow page table technique was too low, Intel developed a hardware-based EPT technique to improve it, as illustrated in Figure 3.13. In addition, Intel offers a Virtual Processor ID (VPID) to improve use of the TLB. Therefore, the performance of memory virtualization is greatly improved. In Figure 3.13, the page tables of the guest OS and EPT are all four-level.

When a virtual address needs to be translated, the CPU will first look for the L4 page table pointed to by Guest CR3. Since the address in Guest CR3 is a physical address in the guest OS, the CPU needs to convert the Guest CR3 GPA to the host physical address (HPA) using EPT. In this procedure, the CPU will check the EPT TLB to see if the translation is there. If there is no required translation in the EPT TLB, the CPU will look for it in the EPT. If the CPU cannot find the translation in the EPT, an EPT violation exception will be raised.

When the GPA of the L4 page table is obtained, the CPU will calculate the GPA of the L3 page table by using the GVA and the content of the L4 page table. If the entry corresponding to the GVA in the L4

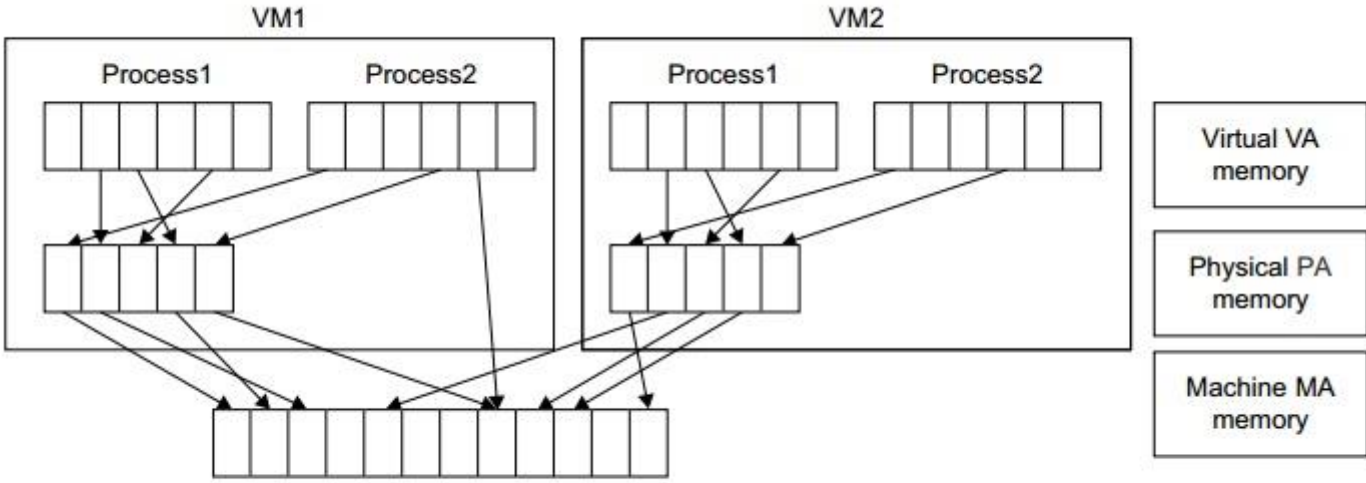


FIGURE 3.12

Two-level memory mapping procedure.

page table is a page fault, the CPU will generate a page fault interrupt and will let the guest OS kernel handle the interrupt. When the PGA of the L3 page table is obtained, the CPU will look for the EPT to get the HPA of the L3 page table, as described earlier. To get the HPA corresponding to a GVA, the CPU needs to look for the EPT five times, and each time, the

memory needs to be accessed four times. There-fore, there are 20 memory accesses in the worst case, which is still very slow. To overcome this short-coming, Intel increased the size of the EPT TLB to decrease the number of memory accesses.

4. 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.

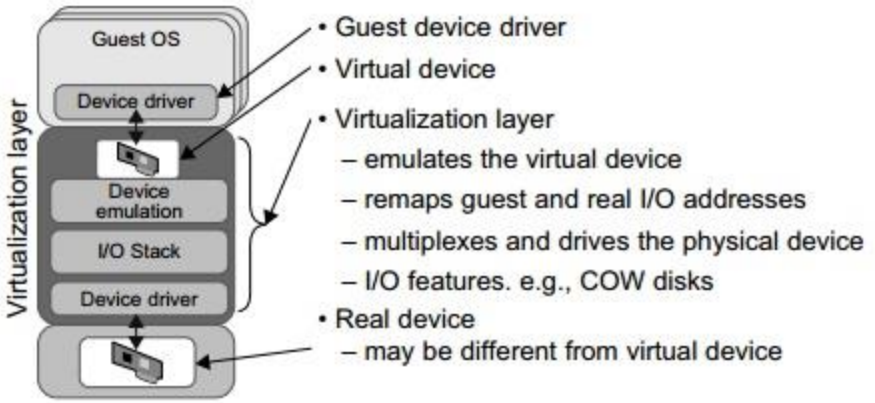


FIGURE 3.14

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.

All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated in software. This software is located in the VMM and acts as a virtual device. The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices. The full device emulation approach is shown in Figure 3.14.

A single hardware device can be shared by multiple VMs that run concurrently. However, software emulation runs much slower than the hardware it emulates [10,15]. 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 dri-ver 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.

Example 3.6 Extended Page Table by Intel for Memory Virtualization

Since the efficiency of the software shadow page table technique was too low, Intel developed a hardware-based EPT technique to improve it, as illustrated in Figure 3.13. In addition, Intel offers a Virtual Processor ID (VPID) to improve use of the TLB. Therefore, the performance of memory virtualization is greatly improved. In Figure 3.13, the page tables of the guest OS and EPT are all four-level.

When a virtual address needs to be translated, the CPU will first look for the L4 page table pointed to by Guest CR3. Since the address in Guest CR3 is a physical address in the guest OS, the CPU needs to convert the Guest CR3 GPA to the host physical address (HPA) using EPT. In this procedure, the CPU will check the EPT TLB to see if the translation is there. If there is no required translation in the EPT TLB, the CPU will look for it in the EPT. If the CPU cannot find the translation in the EPT, an EPT violation exception will be raised.

When the GPA of the L4 page table is obtained, the CPU will calculate the GPA of the L3 page table by using the GVA and the content of the L4 page table. If the entry corresponding to the GVA in the L4

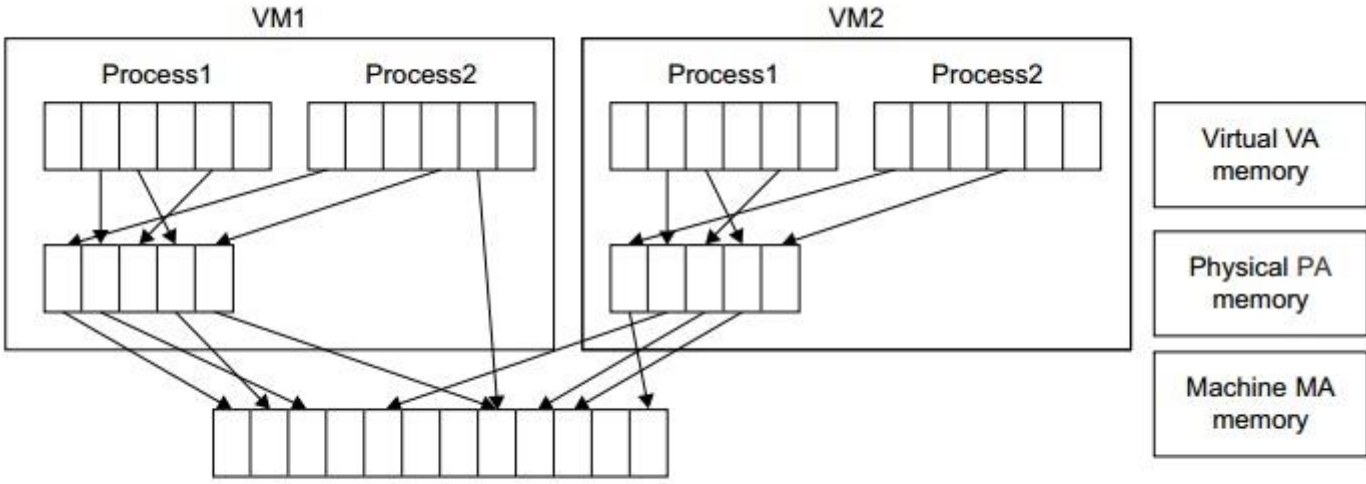


FIGURE 3.12

Two-level memory mapping procedure.

page table is a page fault, the CPU will generate a page fault interrupt and will let the guest OS kernel handle the interrupt. When the PGA of the L3 page table is obtained, the CPU will look for the EPT to get the HPA of the L3 page table, as described earlier. To get the HPA corresponding to a GVA, the CPU needs to look for the EPT five times, and each time, the memory needs to be accessed four times. There-fore, there are 20 memory accesses in the worst case, which is still very slow. To overcome this short-coming, Intel increased the size of the EPT TLB to decrease the number of memory accesses.

4. 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.

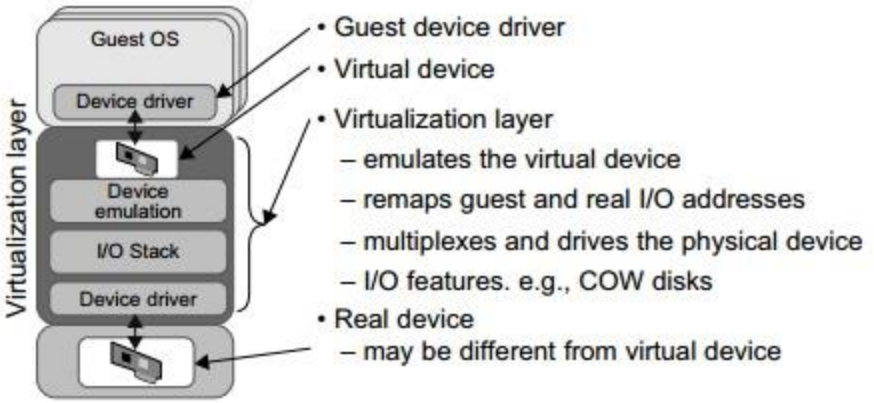


FIGURE 3.14

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.

All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated in software. This software is located in the VMM and acts as a virtual device. The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices. The full device emulation approach is shown in Figure 3.14.

A single hardware device can be shared by multiple VMs that run concurrently. However, software emulation runs much slower than the hardware it emulates [10,15]. 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 dri-ver 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 reassign-ment, it may have been set to an arbitrary state (e.g., DMA to some arbitrary memory locations) that can function incorrectly or even crash the whole system. Since software-based I/O virtualization requires a very high overhead of device emulation, hardware-assisted I/O virtualization is critical. Intel VT-d supports the remapping of I/O DMA transfers and device-generated interrupts. The architecture of VT-d provides the flexibility to support multiple usage models that may run unmodified, special-purpose, or "virtualization-aware" guest OSes.

Another way to help I/O virtualization is via self-virtualized I/O (SV-IO) [47]. The key idea of SV-IO is to harness the rich resources of a multicore processor. All tasks associated with virtualizing an I/O device are encapsulated in SV-IO. It provides virtual devices and an associated access API to VMs and a management API to the VMM. SV-IO defines one virtual interface (VIF) for every kind of virtua-lized I/O device, such as virtual network interfaces, virtual block devices (disk), virtual camera devices, and others. The guest OS interacts with the VIFs via VIF device drivers. Each VIF consists of two mes-sage queues. One is for outgoing messages to the devices and the other is for incoming messages from the devices. In addition, each VIF has a unique ID for identifying it in SV-IO.

Example 3.7 VMware Workstation for I/O Virtualization

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

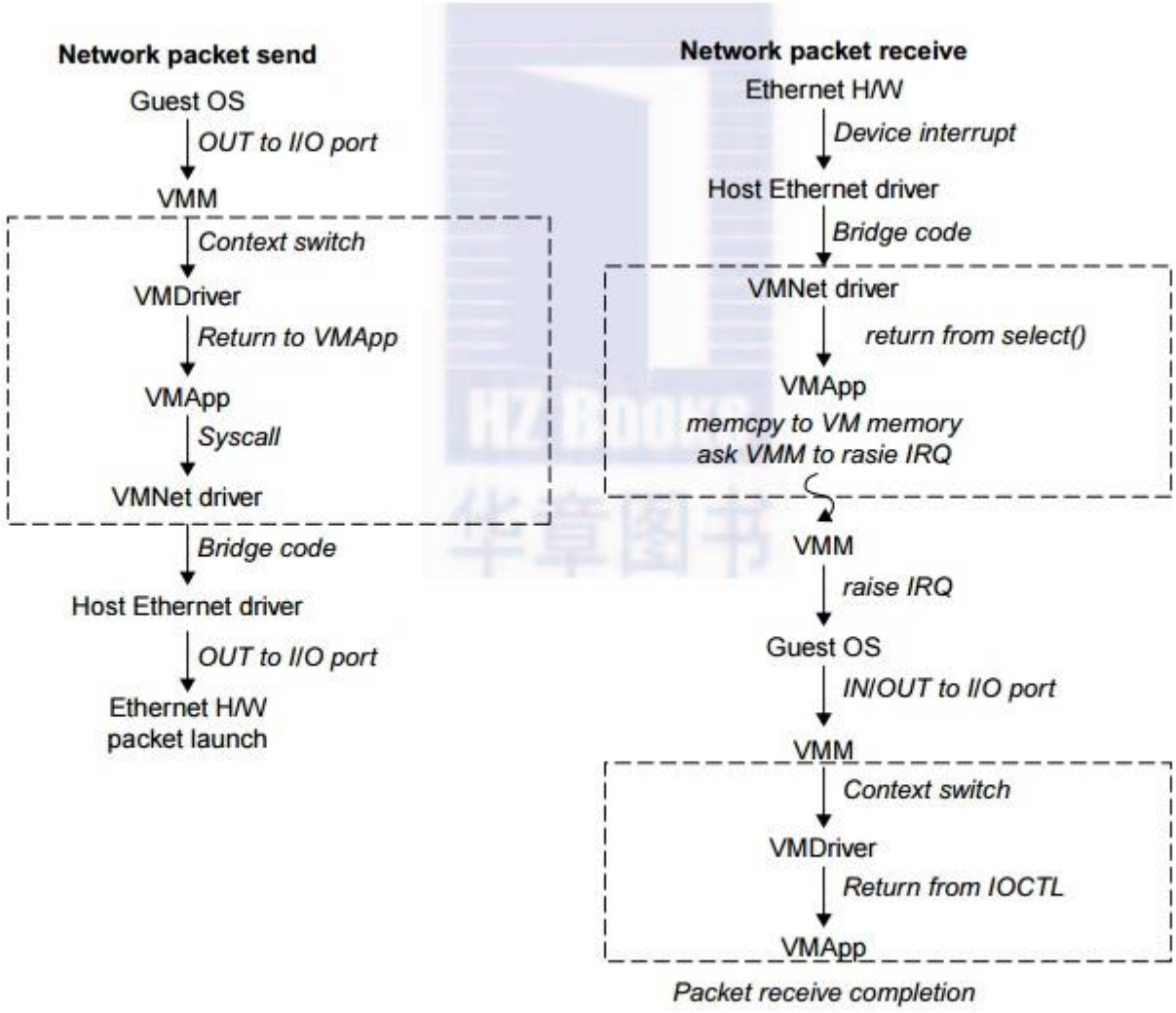


FIGURE 3.15

Functional blocks involved in sending and receiving network packets.

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

5. Virtualization in Multi-Core Processors

Virtualizing a multi-core processor is relatively more complicated than virtualizing a uni-core processor. Though multicore processors are claimed to have higher performance by

integrating multiple processor cores in a single chip, muti-core virtualiuzation has raised some new challenges to computer architects, compiler constructors, system designers, and application programmers. There are mainly two difficulties: Application programs must be parallelized to use all cores fully, and software must explicitly assign tasks to the cores, which is a very complex problem.

Concerning the first challenge, new programming models, languages, and libraries are needed to make parallel programming easier. The second challenge has spawned research involving scheduling algorithms and resource management policies. Yet these efforts cannot balance well among performance, complexity, and other issues. What is worse, as technology scales, a new challenge called dynamic heterogeneity is emerging to mix the fat CPU core and thin GPU cores on the same chip, which further complicates the multi-core or many-core resource management. The dynamic heterogeneity of hardware infrastructure mainly comes from less reliable transistors and increased complexity in using the transistors [33,66].

5.1 Physical versus Virtual Processor Cores

Wells, et al. [74] proposed a multicore virtualization method to allow hardware designers to get an abstraction of the low-level details of the processor cores. This technique alleviates the burden and inefficiency of managing hardware resources by software. It is located under the ISA and remains unmodified by the operating system or VMM (hypervisor). Figure 3.16 illustrates the technique of a software-visible VCPU moving from one core to another and temporarily suspending execution of a VCPU when there are no appropriate cores on which it can run.

5.2 Virtual Hierarchy

The emerging many-core chip multiprocessors (CMPs) provides a new computing landscape. Instead of supporting time-sharing jobs on one or a few cores, we can use the abundant cores in a space-sharing, where single-threaded or multithreaded jobs are simultaneously assigned to separate groups of cores for long time intervals. This idea was originally suggested by Marty and Hill [39]. To optimize for space-shared workloads, they propose using virtual hierarchies to overlay a coherence and caching hierarchy onto a physical processor. Unlike a fixed physical hierarchy, a virtual hierarchy can adapt to fit how the work is space shared for improved performance and performance isolation.

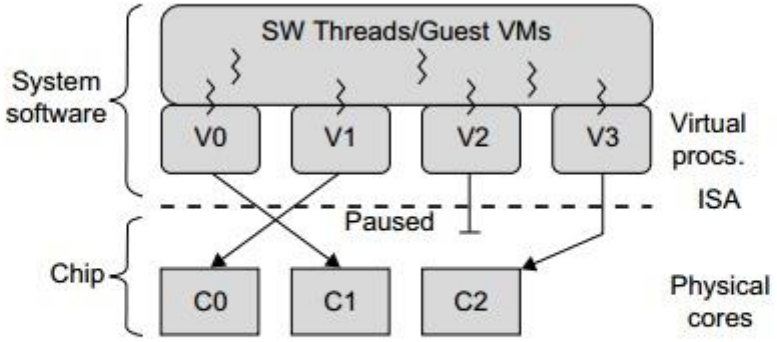


FIGURE 3.16

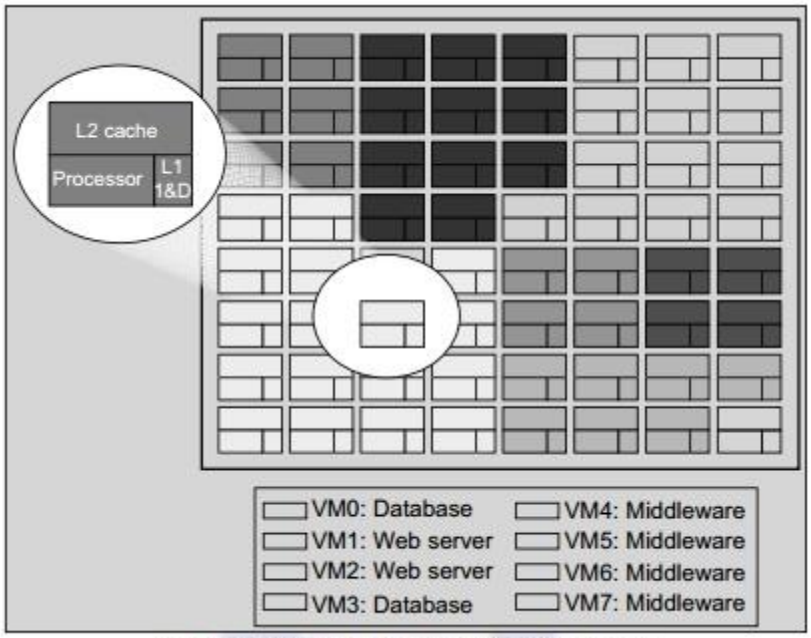
Multicore virtualization method that exposes four VCPUs to the software, when only three cores are actually present.

Today's many-core CMPs use a physical hierarchy of two or more cache levels that statically determine the cache allocation and mapping. A virtual hierarchy is a cache hierarchy that can adapt to fit the workload or mix of workloads [39]. The hierarchy's first level locates data blocks close to the cores needing them for faster access, establishes a shared-cache domain, and establishes a point of coherence for faster communication. When a miss leaves a tile, it first attempts to locate the block (or sharers) within the first level. The first level can also pro-vide isolation between independent workloads. A miss at the L1 cache can invoke the L2 access.

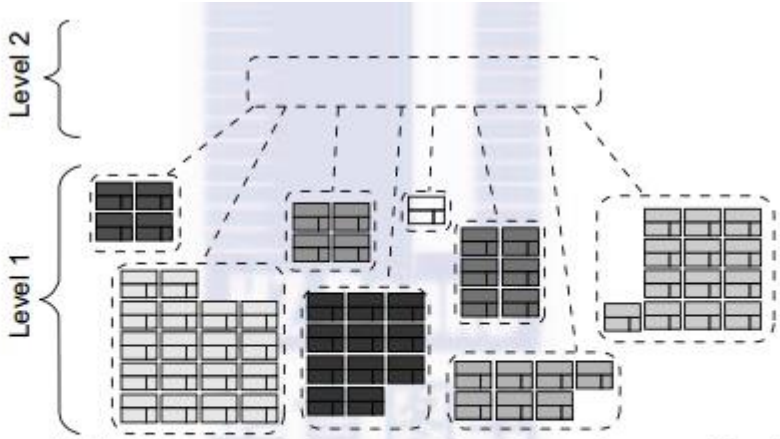
The idea is illustrated in Figure 3.17(a). Space sharing is applied to assign three workloads to three clusters of virtual cores: namely VM0 and VM3 for database workload, VM1 and VM2 for web server workload, and VM4–VM7 for middleware workload. The basic assumption is that each workload runs in its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

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

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



(a) Mapping of VMs into adjacent cores



(b) Multiple virtual clusters assigned to various workloads

FIGURE 3.17

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

What is microservices architecture?

Microservices architecture (often shortened to microservices) refers to an architectural style for developing applications. Microservices allow a large application to be separated into smaller independent parts, with each part having its own realm of responsibility. To serve a single user request, a microservices-based application can call on many internal microservices to compose its response.

Containers are a well-suited microservices architecture example, since they let you focus on developing the services without worrying about the dependencies. Modern cloud-native applications are usually built as microservices using containers.

Learn how Google Kubernetes Engine can help you create microservices-based Within a microservices architecture, each microservice is a single service built to accommodate an application feature and handle discrete tasks. Each microservice communicates with other services through simple interfaces to solve business problems.applications using containers

Microservices architecture defined

A microservices architecture is a type of application architecture where the application is developed as a collection of services. It provides the framework to develop, deploy, and maintain microservices architecture diagrams and services independently.

What is microservices architecture used for?

Typically, microservices are used to speed up application development. Microservices architectures built using Java are common, especially Spring Boot ones. It's also common to compare microservices versus service-oriented architecture. Both have the same objective, which is to break up monolithic applications into smaller components, but they have different approaches. Here are some microservices architecture examples:

Website migration

A complex website that's hosted on a monolithic platform can be migrated to a cloud-based and container-based microservices platform.

Media content

Using microservices architecture, images and video assets can be stored in a scalable object storage system and served directly to web or mobile.

Transactions and invoices

Payment processing and ordering can be separated as independent units of services so payments continue to be accepted if invoicing is not working.

Data processing

A microservices platform can extend cloud support for existing modular data processing services.

What are Microservices?

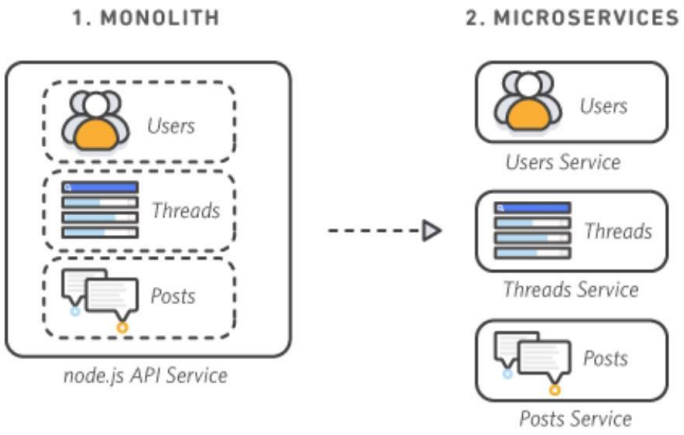
Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs. These services are owned by small, self-contained teams.

Microservices architectures make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market for new features.

Monolithic vs. Microservices Architecture

With monolithic architectures, all processes are tightly coupled and run as a single service. This means that if one process of the application experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features becomes more complex as the code base grows. This complexity limits experimentation and makes it difficult to implement new ideas. Monolithic architectures add risk for application availability because many dependent and tightly coupled processes increase the impact of a single process failure.

With a microservices architecture, an application is built as independent components that run each application process as a service. These services communicate via a well-defined interface using lightweight APIs. Services are built for business capabilities and each service performs a single function. Because they are independently run, each service can be updated, deployed, and scaled to meet demand for specific functions of an application.



Breaking a monolithic application into microservices

Characteristics of Microservices

Autonomous

Each component service in a microservices architecture can be developed, deployed, operated, and scaled without affecting the functioning of other services. Services do not need to share any of their code or implementation with other services. Any communication between individual components happens via well-defined APIs.

Specialized

Each service is designed for a set of capabilities and focuses on solving a specific problem. If developers contribute more code to a service over time and the service becomes complex, it can be broken into smaller services.

Benefits of Microservices

Agility

Microservices foster an organization of small, independent teams that take ownership of their services. Teams act within a small and well understood context, and are empowered to work more independently and more quickly. This shortens development cycle times. You benefit significantly from the aggregate throughput of the organization.

Flexible Scaling

Microservices allow each service to be independently scaled to meet demand for the application feature it supports. This enables teams to right-size infrastructure needs, accurately measure the cost of a feature, and maintain availability if a service experiences a spike in demand.

Easy Deployment

Microservices enable continuous integration and continuous delivery, making it easy to try out new ideas and to roll back if something doesn't work. The low cost of failure enables experimentation, makes it easier to update code, and accelerates time-to-market for new features.

Technological Freedom

Microservices architectures don't follow a "one size fits all" approach. Teams have the freedom to choose the best tool to solve their specific problems. As a consequence, teams building microservices can choose the best tool for each job.

Reusable Code

Dividing software into small, well-defined modules enables teams to use functions for multiple purposes. A service written for a certain function can be used as a building block for another feature. This allows an application to bootstrap off itself, as developers can create new capabilities without writing code from scratch.

Resilience

Service independence increases an application's resistance to failure. In a monolithic architecture, if a single component fails, it can cause the entire application to fail. With microservices, applications handle total service failure by degrading functionality and not crashing the entire application.

What is a Web Service?

- **Web service is a standardized medium to propagate communication between the client and server applications on the WWW (World Wide Web). A web service is a software module that is designed to perform a certain set of tasks.**

Web services in cloud computing can be searched for over the network and can also be invoked accordingly.

When invoked, the web service would be able to provide the functionality to the client, which invokes that web service.

Web Services Architecture

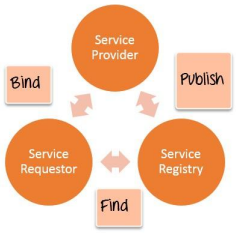
Every framework needs some sort of architecture to make sure the entire framework works as desired, similarly, in web services. The Web Services Architecture consists of three distinct roles as given below :

Provider – The provider creates the web service and makes it available to client applications who want to use it.

Requestor – A requestor is nothing but the client application that needs to contact a web service. The client application can be a .Net, Java, or any other language based application which looks for some sort of functionality via a web service.

Broker – The broker is nothing but the application which provides access to the UDDI. The UDDI, as discussed in the earlier topic, enables the client application to locate the web service.

The diagram below showcases how the Service provider, the Service requestor and Service registry interact with each other.



Web Services Architecture

Publish – A provider informs the broker (service registry) about the existence of the web service by using the broker's publish interface to make the service accessible to clients

Find – The requestor consults the broker to locate a published web service

Bind – With the information it gained from the broker(service registry) about the web service, the requestor is able to bind, or invoke, the web service.

Challenges of Microservices Architecture

Microservice architecture is more complex than the legacy system. The microservice environment becomes more complicated because the team has to manage and support many moving parts. Here are some of the top challenges that an organization face in their microservices journey.

1. Bounded Context

2.Dynamic Scale up and Scale Down

3.Monitoring

4.Fault Tolerance

5.Cyclic dependencies

6.DevOps Culture

Bounded context: The bounded context concept originated in Domain-Driven Design (DDD) circles. It promotes the Object model first approach to service, defining a data model that service is responsible for and is bound to. A bounded context clarifies, encapsulates, and defines the specific responsibility to the model. It ensures that the domain will not be distracted from the outside. Each model must have a context implicitly defined within a sub-domain, and every context defines boundaries.

In other words, the service owns its data and is responsible for its integrity and mutability. It supports the most important feature of microservices, which is independence and decoupling.

Dynamic scale up and scale down: The loads on the different microservices may be at a different instance of the type. As well as auto-scaling up your microservice should auto-scale down. It reduces the cost of the microservices. We can distribute the load dynamically.

Monitoring: The traditional way of monitoring will not align well with microservices because we have multiple services making up the same functionality previously supported by a single application. When an error arises in the application, finding the root cause can be challenging.

Fault Tolerance: Fault tolerance is the individual service that does not bring down the overall system. The application can operate at a certain degree of satisfaction when the failure occurs. Without fault tolerance, a single failure in the system may cause a total breakdown. The circuit breaker can achieve fault tolerance. The circuit breaker is a pattern that wraps the request to external service and detects when they are faulty. Microservices need to tolerate both internal and external failure.

Cyclic Dependency: Dependency management across different services, and its functionality is very important. The cyclic dependency can create a problem, if not identified and resolved promptly.

DevOps Culture: Microservices fits perfectly into the DevOps. It provides faster delivery service, visibility across data, and cost-effective data. It can extend their use of containerization switch from Service-Oriented-Architecture (SOA) to Microservice Architecture (MSA).

Top 5 Reasons Why DevOps Is Important

by Vepambattu Chandu · Feb. 05, 19 · DevOps Zone · Analysis

DevOps is no more than a set of processes that coordinate to unify development teams and processes to complement software development. The main reason behind DevOps' popularity is that it allows enterprises to create and improve products at a faster pace than traditional software development methods.

Devops Popularity

According to the Previous survey, the implementation rate has increased significantly.

In 2016, 66 % of global organizations had adopted self-development, 19 percent had not adopted DevOps, and 15 percent had not yet decided.

As of 2017, 74 percent of global organizations adopted DevOps, 16 percent did not adopt DevOps, and 10 percent were not decided.

1. Shorter Development Cycles, Faster Innovation

When we have a biased response from the development and operations teams, it is often difficult to tell if the application is operational. When development teams simply submit a request, the cycle times are unnecessarily extended.

With joint development and operations efforts, the team's applications are ready to use more quickly. This is important because companies succeed on the basis of their ability to innovate faster than their competitors.

2. Reduce Implementation Failure, Reflections and Recovery Time

The main reason for the failure of the teams in the implementation failure is due to programming defects. With shorter development cycles, DevOps promotes frequent code versions. This, in turn, makes it easy to detect code defects. Therefore, teams can use their time to reduce the number of implementation failures using agile programming principles that require collaboration and standard programming. Recovery time is an important issue because you should expect some failure. But recovery is much faster when development teams and operations work together to share ideas and take into account the challenges of both teams during development.

3. Better Communication and Cooperation

Improved DevOps software development culture. The common teams are happier and more productive. Culture focuses on performance rather than individual goals. When teams trust each other, they can experiment and innovate more effectively. Teams can focus on bringing the product to market or production, and their key performance indicators must be organized accordingly.

It no longer involves "passing" the application to the processes and waiting to see what is happening. Processes do not need to wait for a different team to solve a problem. The process becomes increasingly transparent as all individuals work towards a common goal.

4. Greater Competencies

High efficiency helps accelerate development and makes it less prone to errors. There are ways to automate DevOps tasks. Continuous integration servers automate the code testing process, reducing the amount of manual work required. This means that software engineers can focus on completing tasks that can not be automated.

Speeding up tools are another chance to increase efficiency. For example:

The scalable infrastructure, such as cloud-based platforms, increases the team's access to hardware resources. As a result, testing and deployment are accelerated.

Acceleration tools can be used to compile code more quickly.

Parallel workflows can be integrated into the continuous delivery chain to avoid delays; one more team is expected to complete its work.

Using an environment avoids a useless task of transferring data between environments. This means that you do not have to use a development environment, a different testing environment, and a third implementation.

5. Reduce Costs and IT Staff

All the benefits of DevOps translate into reduced general costs and requirements of IT staff. DevOps development teams require IT staff to be 35 percent less and IT costs 30 percent lower.

1. Continuous Development:

This stage involves committing code to version control tools such as Git or SVN for maintaining the different versions of the code, and tools like Ant, Maven, Gradle for building/packaging the code into an executable file that can be forwarded to the QAs for testing.

2. Continuous Integration:

The stage is a critical point in the whole DevOps Lifecycle. It deals with integrating the different stages of the DevOps lifecycle and is, therefore, the key in automating the whole DevOps Process.

3. Continuous Deployment:

In this stage the code is built, the environment or the application is containerized and is pushed onto the desired server. The key processes in this stage are Configuration Management, Virtualization, and Containerization.

4. Continuous Testing:

The stage deals with automated testing of the application pushed by the developer. If there is an error, the message is sent back to the integration tool, this tool, in turn, notifies the developer of the error, If the test was a success, the message is sent to Integration-tool which pushes the build on the production server.

5. Continuous Monitoring:

The stage continuously monitors the deployed application for bugs or crashes. It can also be set up to collect user feedback. The collected data is then sent to the developers to improve the application.

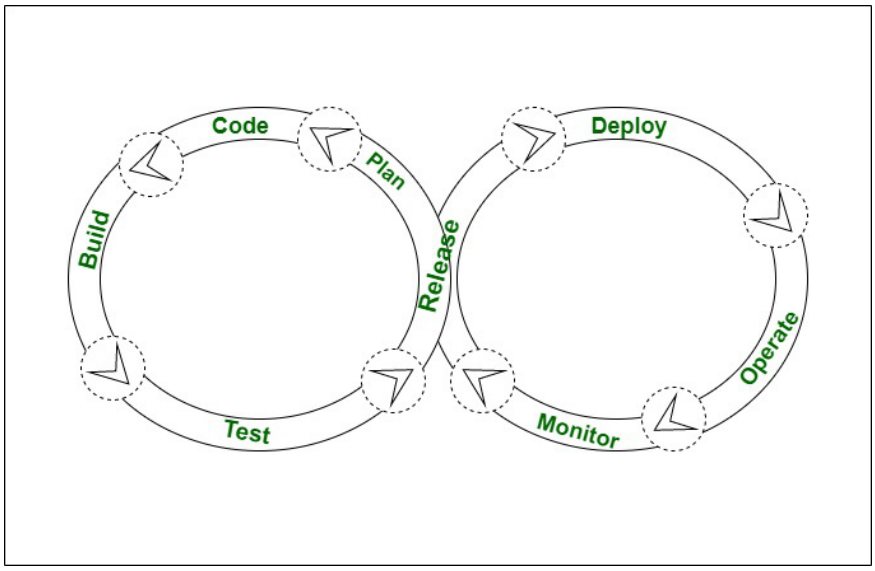
Lifecycle of DevOps

Last Updated : 05 Aug, 2020

DevOps Lifecycle is the set of phases that includes in DevOps for taking part development and operation group duties for quicker software program delivery. DevOps follows positive techniques that consist of code, build, test, release, deploy, operate, display and plan.

DevOps lifecycle follows a range of phases such as non-stop development, non-stop integration, non-stop testing, non-stop monitoring, and non-stop feedback. Each segment of the DevOps lifecycle related to some equipment and applied sciences to obtain the process. Some of the frequently used equipment are open source and are carried out primarily based upon the commercial enterprise requirements. DevOps lifecycle is effortless to manipulate and it helps satisfactory delivery.

The Lifecycle of DevOps :
Below is the diagram which indicates the structure of the DevOps lifecycle.



1. **Code –**
   Coding is the first step in the development of DevOps. In this step, the developers and coders write the code on any platform to develop the product for a customer.

2. **Build –**
   The 2nd step is to construct the model, in which the simple model of the product has constructed the use of an appropriate programming language.

3. **Test –**
   The third step is to test or to check where the constructed merchandise and other products are going to

examine with the use of the automation checking out equipment such as Bugzilla and selenium web driver.

4. **Release –**
   This step includes planning, scheduling and controlling the constructed method in a one of a kind environment.
5. **Deploy –**
   The step after the release phase will be Deployment where all the products, files and documents which are to deployed are carried out on the server.

6. **Operate –**
   After the deployment phase i.e. after the deployment of the product or application, it is delivered to the client for use where he makes use of that product or software for everyday lifestyles purposes.

7. **Monitor –**
   In this step, the delivered merchandise or software to a client has been monitored to observe down any up-time and down-time failures or if any errors are there or not.

8. **Plan –**
   The last phase is Planning. After monitoring it gathers all the information, data, feedback and comments from the client and plans the modifications that need to be performed to make it better.

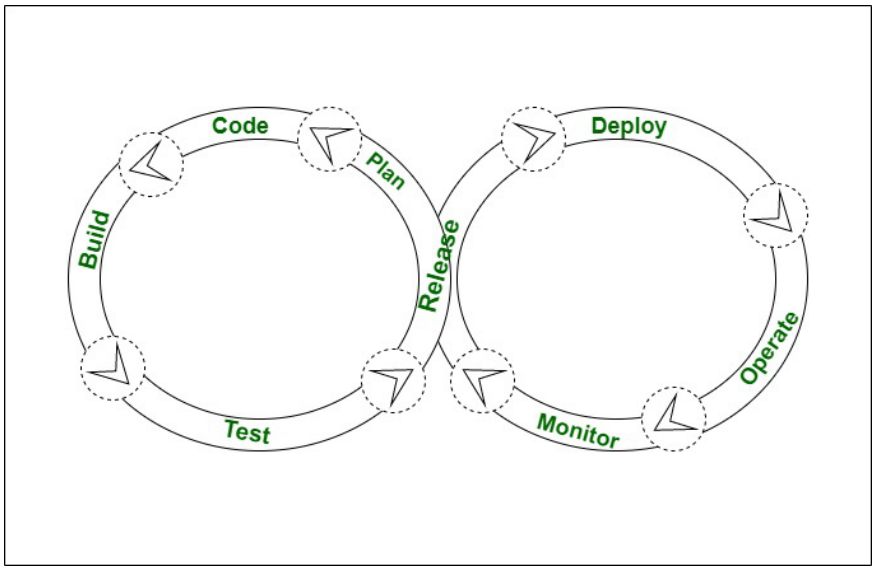
Lifecycle of DevOps

Last Updated : 05 Aug, 2020

DevOps Lifecycle is the set of phases that includes in DevOps for taking part development and operation group duties for quicker software program delivery. DevOps follows positive techniques that consist of code, build, test, release, deploy, operate, display and plan.

DevOps lifecycle follows a range of phases such as non-stop development, non-stop integration, non-stop testing, non-stop monitoring, and non-stop feedback. Each segment of the DevOps lifecycle related to some equipment and applied sciences to obtain the process. Some of the frequently used equipment are open source and are carried out primarily based upon the commercial enterprise requirements. DevOps lifecycle is effortless to manipulate and it helps satisfactory delivery.

The Lifecycle of DevOps :
Below is the diagram which indicates the structure of the DevOps lifecycle.



1. **Code –**
   Coding is the first step in the development of DevOps. In this step, the developers and coders write the code on any platform to develop the product for a customer.

2. **Build –**
   The 2nd step is to construct the model, in which the simple model of the product has constructed the use of an appropriate programming language.

3. **Test –**
   The third step is to test or to check where the constructed merchandise and other products are going to

examine with the use of the automation checking out equipment such as Bugzilla and selenium web driver.

4. **Release –**
   This step includes planning, scheduling and controlling the constructed method in a one of a kind environment.
5. **Deploy –**
   The step after the release phase will be Deployment where all the products, files and documents which are to deployed are carried out on the server.

6. **Operate –**
   After the deployment phase i.e. after the deployment of the product or application, it is delivered to the client for use where he makes use of that product or software for everyday lifestyles purposes.

7. **Monitor –**
   In this step, the delivered merchandise or software to a client has been monitored to observe down any up-time and down-time failures or if any errors are there or not.

8. **Plan –**
   The last phase is Planning. After monitoring it gathers all the information, data, feedback and comments from the client and plans the modifications that need to be performed to make it better.

What is Server Virtualization in Cloud Computing?

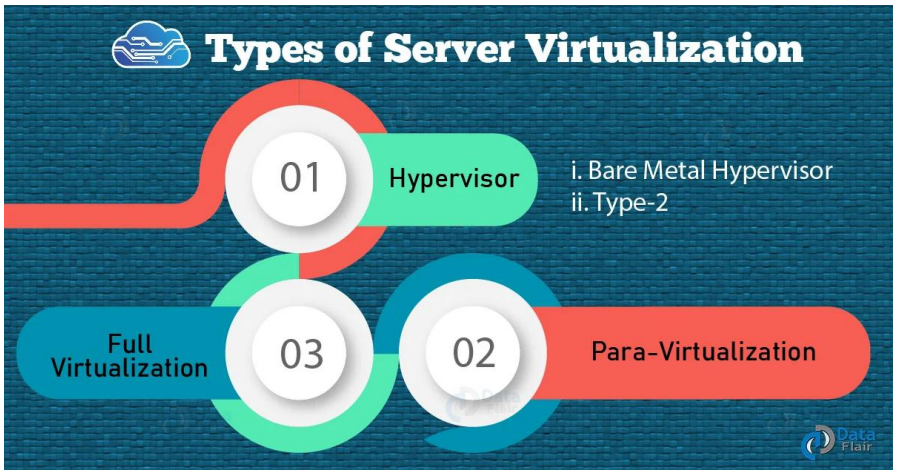
Server virtualization is a partition of physical servers into multiple virtual servers. Here, each virtual server is running its own operating system and applications. It can be said that server virtualization in cloud computing is the masking of server resources.

The server is familiar with the identity of individual physical servers. The single physical server is divided into multiple isolated virtual servers, with the help of software

We can use server virtualization in IT infrastructure, this can reduce cost by increasing the utilization of existing servers. Server virtualization generally benefits from small to medium scale applications.

Types of Server Virtualization

There are 3 types of server virtualization in cloud computing:



i. Hypervisor

A Hypervisor is a layer between the operating system and hardware. The hypervisor is the reason behind the successful running of multiple operating systems.

It can also perform tasks such as handling queues, dispatching and returning the hardware request. Host operating system works on the top of the hypervisor, we use it to administer and manage the virtual machines.

Types of Hypervisor

The hypervisor uses to enable server virtualization in Cloud Computing. There are two

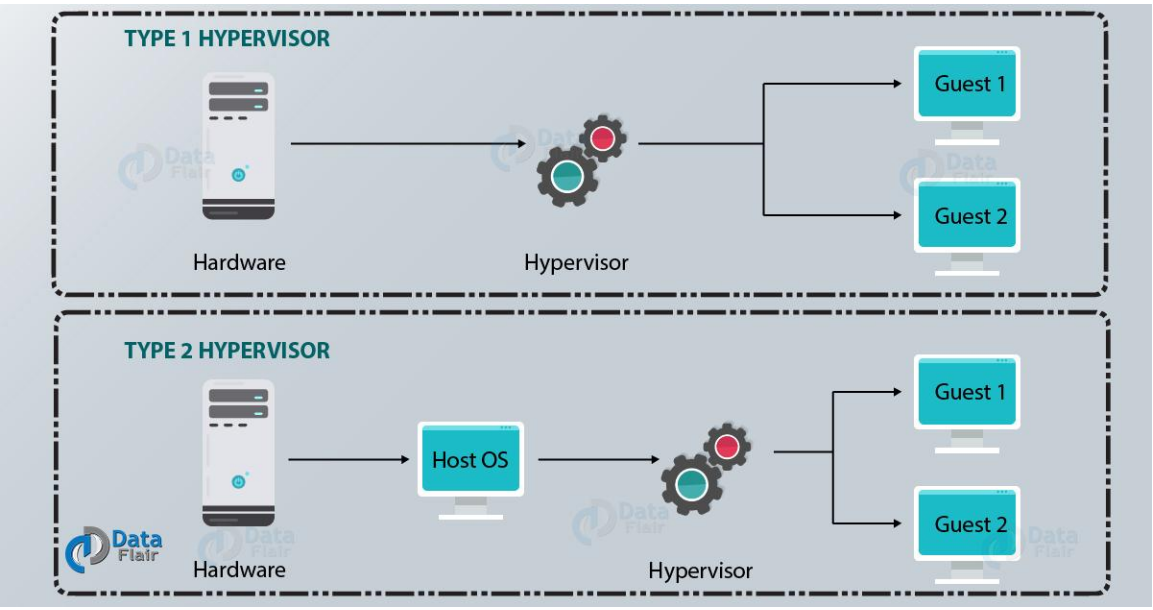
types of hypervisor such as-

i. Bare Metal Hypervisor

The Bare-metal hypervisor is installed directly on the top of the host hardware. It manages all the hardware resources which are installed inside the tin. The hardware resource is further allocated to the virtual machine. VMware vSphere ESXi is an example of the bare metal hypervisor.

ii. Type-2 Hoster

The second type of hypervisor runs directly on the top of the conventional operating system. Type 2 hypervisor has some architecture limitation. They are quite popular in a nonproduction environment and VMware Workstation for VirtualBox is the example of type-2.



ii. Para-Virtualization

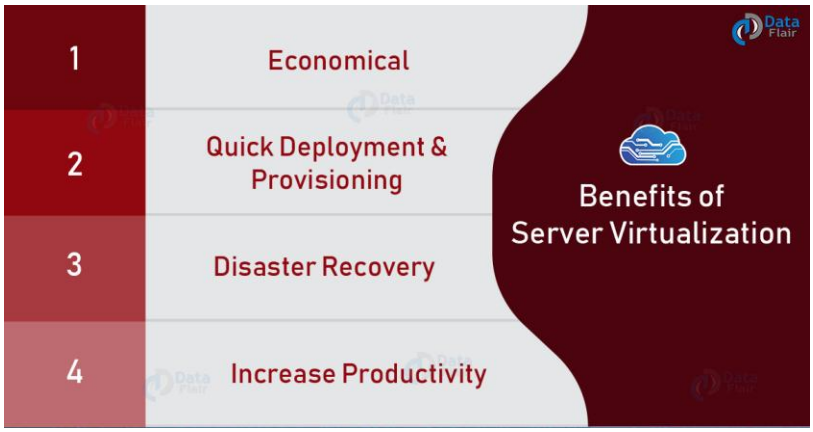
In Para-virtualization model, simulation in trapping overhead in software virtualizations. It is based on the hypervisor and the guest operating system and modified entry compiled for installing it in a virtual machine.

After the modification, the overall performance is increased as the guest operating system communicates directly with the hypervisor.

iii. Full Virtualization

Full virtualizations can emulate the underlying hardware. It is quite similar to Para-virtualization. Here, machine operation used by the operating system which is further used to perform input-output or modify the system status.

Server Virtualization Benefits



i. Economical

This is one of the major benefits of server virtualization because it can divide a single server into multiple virtual servers which eliminate the cost of physical hardware.

ii. Quick Deployment and Provisioning

Within minutes, you can perform the provisioning and deployment process. Here, Server Virtualization allows replicating an existing virtual machine.

iii. Disaster Recovery

A data virtually move from one server to another, quickly and safely. You can store the data anywhere and retrieved from anywhere, this process consumes less time and downtime will be very less.

iv. Increase Productivity

If the physical servers are less in amount then it will be easy for them to maintain. In addition, there are many tools available for making provision and convert services as efficiently as possible.