1

CLOUD COMPUTING

Unit Structure

1.0 OBJECTIVE

- To understand the concept of cloud computing.
- To study the characteristics and benefits of cloud computing.
- To Understand the basic concepts of distributed computing.
- To learn about Service Oriented and Utility Oriented Computing.

1.1 INTRODUCTION TO CLOUD COMPUTING

Cloud computing technologies are:

- Virtualization
- Service-Oriented Architecture (SOA)
- Grid Computing
- Utility Computing

Virtualization

"Virtualization means the process of creating a virtual environment to run multiple applications and operating systems on the same server".

Types of Virtualization

1. Hardware virtualization
2. Server virtualization
3. Storage virtualization
4. Operating system virtualization
5. Data Virtualization

Service-Oriented Architecture (SOA)

Service-Oriented Architecture enables organizations to access on-demand cloud-based computing solutions on the report of the change of business requirement.

Applications of Service-Oriented Architecture

1. It is used in the healthcare industry.
2. It is used to create different mobile applications and games using SOA.
3. In the air force, SOA infrastructure is used to establish situational awareness systems.

Fig. 1. SOA

Grid computing

Grid computing is also known as distributed computing. It is a type of processor architecture that merges various different computing resources from multiple locations to achieve a common goal.

Types of machines used in computing:

1. Control Node: It is a group of servers which administers the whole network.

2. Provider: It is a computer which contributes its resources in the network resource pool.

3. User: It uses the resources on the network.

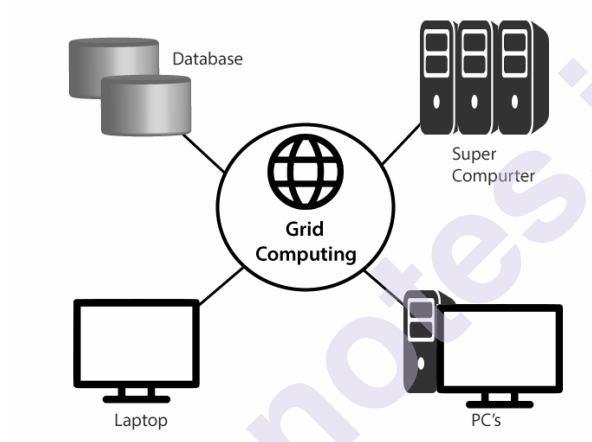


Fig.2 Grid Computing

Utility Computing

Utility computing is the bulk trending IT service model. It supply on-demand computing resources (i.e. computation, storage, and programming services via API) and infrastructure based on the pay per use method.

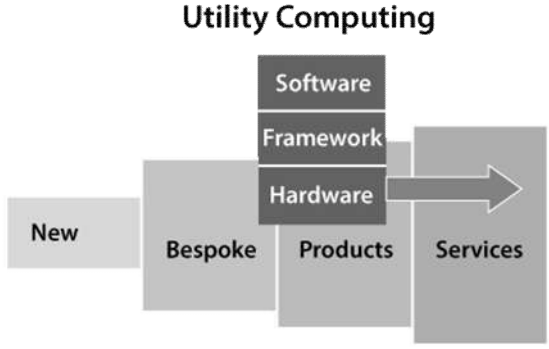


Fig.3. Utility Computing

- Cloud computing gives guarantee to transform computing into a utility delivered over the internet.
- Enterprise architecture is a function within IT departments that has developed over time, playing a high value role in managing transitions to new technologies, such as cloud computing.

1.2 CHARACTERISTICS AND BENEFITS OF CLOUD COMPUTING

- Cloud computing refers to different technologies, services, and concepts.
- It is associated with virtualized infrastructure or hardware on demand, utility computing, IT outsourcing, platform and software as a service, and many other things that now are the heart of the IT industry.
- Figure 4 shows too many different ideas included in current definitions of cloud computing systems.
- The term cloud has an abstraction of the network in system diagrams. This meaning is also put into cloud computing, which refers to an Internet-centric way of computing.
- The Internet plays a fundamental role in cloud computing, it shows the medium or the platform through which many cloud computing services are delivered and made accessible.

Definition of cloud computing:

"Cloud computing introduces both the applications delivered as services over the Internet and the hardware and system software in the data centers that provide those services."

- Cloud computing as aoccurance touching on the entire stack: from the underlying hardware to the high-level software services and its applications.
- Here is the concept of everything as a service, like XaaS, the different components of a system IT infrastructure, development platforms, databases, and so on can be delivered, measured, and consequently cost as a service.
- Cloud computing is a model for defining ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., like networks, servers, storage, applications, and services) that can be instantly provisioned and free with less management effort or service provider interaction.

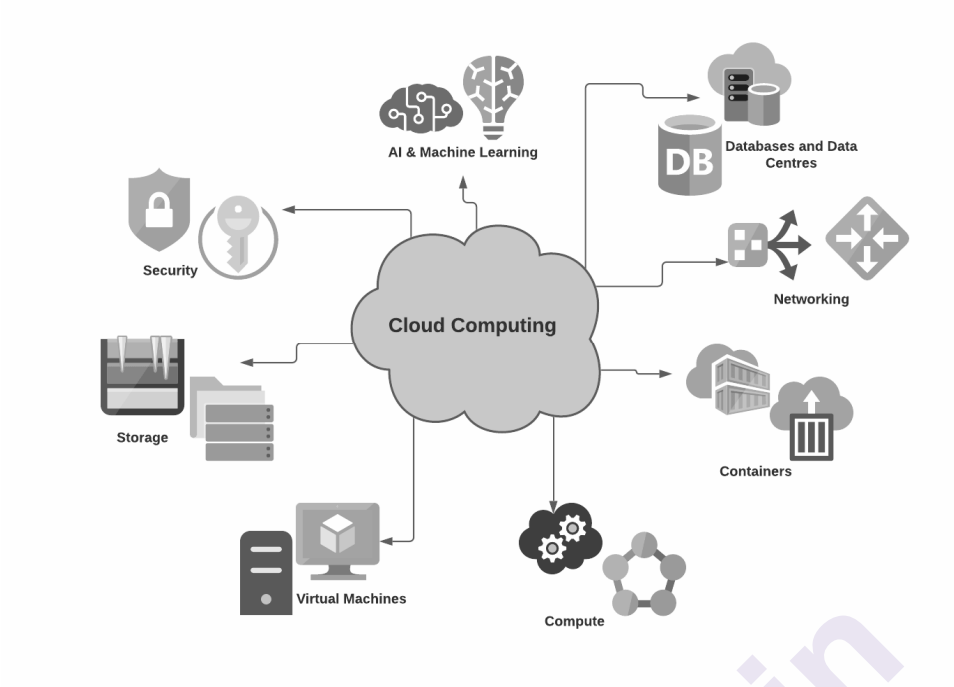


Fig. 4 Cloud computing environment

- Utility-oriented approach is an important aspect of cloud computing .
- Cloud computing concentrates on delivering services with a given pricing model, in most cases a pay-per-use method.
- It is also possible to access online storage, rent the virtual hardware, or use development platforms and pay only for their effective usage, with no or minimal up-front costs.
- All these operations can be executed and billed simply by entering the credit card details and accessing the exposed services through a Web browser.
- The criteria to disfavor whether a service is delivered in the cloud computing style:
  - The service is accessible through a Web browser or a Web services application programming interface.
  - Zero capital expenditure
  - pay only for what you use

- **Characteristics**

  - **Resources Pooling**

Cloud providers pulled the computing resources to give services to multiple customers with the help of a multi-tenant model.

- **On-Demand Self-Service**

It is one of the key and valuable features of Cloud Computing as the user can regularly monitor the server uptime, capabilities, and allotted network storage.

- **Easy Maintenance**

The servers are easy to maintain and the downtime is required very less and even in some situations, there is no downtime.

- **Large Network Access**

The user can access the data of the cloud or upload the data to the cloud from anywhere just with the help of a device and an internet connection.

- **Availability**

The potential of the Cloud can be altered as per the use and can be extended a lot. It studies storage usage and allows the user to purchase extra Cloud storage if needed for a very small amount.

- **Automatic System**

Cloud computing automatically analyzes the data needed and supports a metering capability at some level of services.

- **Economical**

It is a one-time investment as the company has to buy the storage and a small part of it can be provided to the many companies which save the host from monthly or yearly costs.

- **Security**

Security creates a snapshot of the data stored so that the data may not get lost even if one of the single servers gets damaged.

- **Pay as you used**

In cloud computing, the user has to pay only for the service they have utilized and used.

- **Measured Service**

Cloud Computing resources used to handle and the company uses it for recording.

1.3 BASIC CONCEPTS OF DISTRIBUTED SYSTEMS

- In Cloud Computing whenever a need and demand from thirds party, that make its available their service to party which is essentially large DS.

- The characterization of a distributed system is define using definition:

"A distributed system is a collection of independent computers that appears to its users as a single coherent system."

- This includes a variety of computer systems, but it affirms two most important elements that distinguish a distributed system.
- DS is composed of multiple independent components and that these components are recognized as a single entity by users.
- This is true for the case of cloud computing, in which clouds hide the complex architecture they depend on and provide a single interface to users.
- The primary use of distributed systems is to share resources and utilize them in a better way.
- The infrastructure, runtime environments, and services are rented to users.
- In fact, one of the navigational factors of cloud computing has been the availability of the large computing facilities or services of IT such as Amazon, Google.
- This offers their computing capabilities as a service provided opportunities for best utilization of their infrastructure.
- Distributed systems often show other properties such as heterogeneity, openness, scalability, transparency, concurrency, continuous availability, and independent failures.
- To some areas these also distinguish clouds, especially in the factors of scalability, concurrency, and continuous availability.
- There are three major milestones that have led to cloud computing facilities:

1. Mainframe Computing

- Mainframes controlling the large computational facilities with multiple processing units.
- Mainframes are powerful, highly reliable computers functional for large data movement and massive input and output operations.
- The Mainframe computing is used by large organizations for huge data processing tasks like online transactions, enterprise resource planning, and other operations that require the processing of significant amounts of data.
- One of the most highlighted features of mainframes computing was the ability to be highly reliable computers that were "always on" and be up to tolerating failures transparently.

- System shutdown process was needed to replace failed components, and the system worked properly without any interruption.
- Batch processing is the main application of mainframes computing. Not only are their popularity and deployments reduced, but also extended versions of such systems are presently in use for transaction processing.
- Examples, such as online banking, airline ticket booking, supermarket and telcos, and government services.

2. Cluster computing

- Cluster computing started using a low-cost another method to the use of mainframes and supercomputers.
- The technology advancement that makes faster and more powerful mainframes and supercomputers in time generated an increased availability of cheap product machines as a side effect.
- These machines are connected by a high-bandwidth network and controlled by certain software tools that handle them as an individual system.
- Cluster technology contributed to the natural selection of tools and frameworks for distributed computing, example including Condor, Parallel Virtual Machine (PVM), and Message Passing Interface (MPI)
- One of the engaging features of clusters was that the computational power of commodity machines could be advantages to solve problems that were previously manageable only on expensive supercomputers.
- Besides, the clusters could be easily extended if more computational power was needed.

3. Grid computing

- Grid computing is a description of the power grid, grid computing suggests a new approach to access large computational power, large storage facilities, and a different variety of services.
- Users can consume resources in the same manner as they use other utilities such as power, gas, and water.
- Grids initially developed as aggregations of geographically scatter clusters by means of Internet connections.
- These types of clusters belonged to different organizations, and arrangements were made among them to share the computational power.
- This is different from a large cluster, a computing grid was a dynamic collection of heterogeneous computing nodes, and its scale was nationwide or even worldwide.

- Several developments made possible the spreading of computing grids:

a. Clusters is a quite common resources

b. they were frequently underutilized

c. New problems were requiring computational power that move beyond the capability of single clusters

d. The improvements in networking and the spreading of the Internet made possible long-distance, high-bandwidth connectivity. All these elements guide the development of grids.

- Cloud computing is frequently examined as the successor of grid computing.
- In actuality, it related aspects of all these three major technologies. Computing clouds are deployed in large data centers provided by a single organization that provides services to others.
- In the case of mainframes clouds are distinguished by the fact of having virtually unbounded capacity, being liberal to failures, and being always on..
- In the case of clusters, The computing nodes that shape the infrastructure of computing clouds are commodity machines.
- The services made available by a cloud vendor are consumed on a pay-per-use basis, and clouds fully implement the utility vision established by grid computing.

1.4 WEB 2.0

- Cloud computing delivers its services through the Web which is the primary interface.
- The Web encloses a set of technologies and services that ease interactive information sharing, collaboration, user-centered design, and application composition.
- This transforms the Web into a rich platform for developing applications, known as Web 2.0.
- This term captures a new way in which developers design applications and deliver these services with the Internet and gives new experiences for users of these applications and services.
- Web 2.0 provides interactivity and flexibility to Web pages, providing added user experience by gaining Web-based access to all the functions that are normally present in desktop applications.
- This potential is obtained by integrating a collection of standards and technologies such as XML, Asynchronous JavaScript and XML (AJAX), Web Services.

- These technologies allow us to build applications advantageous to the contribution of users, who presently become providers of content.
- The Internet opens new opportunities and markets for the Web, the services of which can now be accessed from a variety of devices such as mobile phones, car dashboards, TV sets, etc.
- Web 2.0 applications are especially dynamic, they upgrade continuously, and new updates and features are integrated at a constant rate.
- Web 2.0 applications aim to strengthen the "long tail" of Internet users by making it available to everyone in terms of either media accessibility.
- Examples of Web 2.0 applications are Google Documents, Google Maps, Flickr, Facebook, Twitter, YouTube, Blogger..
- Social Websites take the biggest advantage of Web 2.0 applications. The level of interaction in Websites such as Facebook or Flickr would not have been possible without the support of AJAX technology.
- Facebook is a social networking site that leverages user activity to enable content, and Blogger, like any other blogging website, provides an online diary that is fed by users.
- Web 2.0. Applications and frameworks for implementing rich Internet applications (RIAs) are fundamental for building cloud services approachable to the extensive public.
- Web 2.0 applications definitely contributed to making people more accustomed to the use of the Internet in their everyday lives and
- Web 2.0 which opened the path to the acceptance of cloud computing as a standard.
- The IT infrastructure is offered through a Web interface.

1.5 SERVICE ORIENTED COMPUTING

- In this Service orientation computing defines the core reference model for cloud computing systems.
- This approach acquires the concept of services as the main element of application and system development.
- Service-oriented computing helps to develop rapid, low-cost, flexible, interoperable, and evolvable applications and systems.
- A service is an abstraction act for a self-describing and platform challenger component that can execute any function, any from a simple function to a complex business process.

- Virtually any segment of code that performs a task can be changed into a service and expose its functionalities through a network-accessible protocol.
- A service needs to be loosely coupled, reusable, programming language independent, and location transparent. Loose coupling enables services to obey different frameworks more easily and makes them reusable.
- Independence from a specific platform increases services accessibility.
- Accordingly, a wider range of clients, which can improve services in global registries and consume them in a location transparent manner, can be served.
- Services are controlled and accumulated into a service-oriented architecture, which is a logical way to arrange software systems to provide end users or other entities distributed over the network with services through published and discoverable interfaces.
- Service oriented computing establishes and broadcasts two important concepts, which are also fundamental to cloud computing:

1. quality of service (QoS)
2. Software-as-a-Service (SaaS)

- **Quality of service**:
  - recognize a set of functional and nonfunctional attributes that can be used to estimate the behavior of a service from different Viewpoints.
  - QoS could be performance metrics such as response time, or security attributes, transactional integrity, reliability, scalability, and availability.
  - QoS requirements are formed between the client and the provider through an SLA that recognizes the least values for the QoS attributes that are required to be satisfied upon the service call.
- **Software-as-a-Service**
  - The idea of Software-as-a-Service introduces a new delivery model for applications.
  - The word has been inherited from the application service providers (ASPs), which bring software services-based solutions over the wide area network from a central datacenter and make them available on a rental basis.
  - "The ASP is responsible for maintaining the infrastructure and making available the application, and the client is discharged from maintenance costs and difficult upgrades.

- This SD model is possible because economies of scale are reached by means of timeshare.
- The SaaS approach achieves its full development with service-oriented computing.
- Loosely coupled software components allow the delivery of complex business processes and transactions as a service while allowing applications to be composed on the fly and services to be reused from everywhere and by anyone.

1.6 UTILITY-ORIENTED COMPUTING

- Utility computing is an observation of computing that defines a service provisioning model for calculating services in which resources such as storage, compute power, applications, and infrastructure are packaged and offered on a pay/use basis.
- The concept of providing computing as a utility like natural gas, water, power, and telephone connection has a long past but has become a reality with the advent of cloud computing.
- This vision can be observed as a "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility, just as the telephone system is a public utility. The computer utility could become the pillar of a new and important industry."
- Utility-Oriented Computing is the first trace of this service-provisioning model that began in the mainframe era.
- IBM and other mainframe providers offered mainframe power to organizations like banks and government agencies throughout their data centers.
- The business model initiated with utility computing brings new requirements and conduct to improvements in mainframe technology.
- Alo provide additional features such as operating systems, process control, and user-metering facilities.
- The idea of computing as utility abided and extended from the business domain to the educational sector with the advent of cluster computing.
- Not only businesses but also research institutes became acquainted with the idea of exploiting an external IT infrastructure on request.
- Computational science is one of the major operating factors for building computing clusters, still requiring large computing power for addressing problems, and not all the institutions were able to fulfill their computing requirements internally.

- The capillary scattering of the Internet and the Web enables the technological means to notice utility computing on a worldwide scale and through simple interfaces.
- Computing grids provided a planet scale distributed computing infrastructure that was approachable on demand. Computing grids bring the concept of utility computing to a new level.
- With the help of utility computing accessible on a wider scale, it is easier to provide a trading infrastructure where grid products storage, computation, and services are offered for or sold.
- Here E-commerce technology provided the infrastructure support for utility computing. Example, significant interest in buying any kind of good online spreads to the wider public: food, clothes, multimedia products, and online services such as storage space and Web hosting.
- Applications were not only dispenses, they started to be composed as a network of services provided by different entities.
- These services, accessible through the Internet, were made available by charging on the report to usage.
- SOC widened the concept of what could have been retrieved as a utility in a computer system, not only measuring power and storage but also services and application components could be employed and integrated on demand.

1.7 LET US SUM UP

- The vision and opportunities of cloud computing along with its characteristics and challenges.
- The cloud computing paradigm emerged as a result of the maturity and convergence of several of its supporting models and technologies, namely distributed computing, virtualization, Web 2.0, service orientation, and utility computing.
- The only element that is shared among all the different views of cloud computing is that cloud systems support dynamic provisioning of IT services and adopt a utility-based cost model to price these services.
- This concept is applied across the entire computing stack and enables the dynamic provisioning of IT infrastructure and runtime environments in the form of cloud-hosted platforms for the development of scal- able applications and their services. This vision is what inspires the Cloud Computing Reference Model.
- This model identifies three major market segments for cloud computing: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

1.8 LIST OF REFERENCES

- Mastering Cloud Computing, RajkumarBuyya, Christian Vecchiola, S ThamaraiSelvi, Tata McGraw Hill Education Private Limited, 2013.

1.9 UNIT END EXERCISES

1. What is the innovative characteristic of cloud computing?
2. Which are the technologies on which cloud computing relies?
3. Provide a brief characterization of a distributed system.
4. Define cloud computing and identify its core features.
5. What is the major revolution introduced by Web 2.0?
6. Describe the main characteristics of a service orientation.
7. What is utility computing?
8. Explain the Cloud Computing Reference Model.

2

ELEMENTS OF PARALLEL COMPUTING

Unit Structure

2.0 OBJECTIVES

- To understand the concept of Parallel & Distributed computing.
- To study the elements of Parallel Distributed computings.
- To study the different types of technologies for Distributed Computing.

2.1 INTRODUCTION

- The analogous/ simultaneous development in availability of big data and in the number of simultaneous users on the Internet places particular pressure on the need to take out computing tasks in parallel, or simultaneously.
- Parallel and distributed computing occurs around many different topic areas in computer science, including algorithms, computer architecture, networks, operating systems, and software engineering.
- During the early 21st century there was volatile growth in multiprocessor design and other strategies for complex applications to run rapidly.
- Parallel and distributed computing assemble on fundamental systems concepts, such as concurrency, mutual exclusion, consistency in state or memory manipulation, message-passing, and shared-memory models.

- The first steps in this conduct direction to the development of parallel computing, which encloses techniques, architectures, and systems for performing multiple activities in parallel.
- The term parallel computing has indistinct edges with the term distributed computing and is often used in place of the latter term.
- In this chapter, we associate it with its proper characterization, which involves the introduction of parallelism within a single computer by coordinating the activity of multiple processors together.

2.2 ELEMENTS OF PARALLEL COMPUTING

Parallel processing

- "Processing of multiple tasks simultaneously on multiple processors is called parallel processing."
- The parallel program consists of multiple active processes or tasks simultaneously solving a particular problem.
- A given task is divided into multiple subtasks using a divide and conquer technique (data structure), and each subtask is processed on a different central processing unit (CPU).
- Programming on a multiprocessor system with the divide and conquer technique is called parallel programming.
- Many applications this day require more computing power than a traditional sequential computer can offer.
- Parallel processing provides a cost effective solution to this problem by increasing the number of CPUs in a computer and by computing an efficient communication system between them.
- The workload can then be shared between different processors. This arrangement results in higher computing power and performance than a single processor system supply.
- The development of parallel processing is being determined by many factors. The noticeable among them consist the following:
  - Computational requirements are regularly increasing in the areas of both scientific and business computing. The technical computing problems, which need high speed estimation power, are associated with life sciences, aerospace, geographical information systems, mechanical design and analysis..
  - Sequential architectures are achieving physical limitations as they are compulsion by the speed of light and thermodynamics laws. The speed at which sequential CPUs can handle is reaching saturation point, and hence a substitute way to get high computational speed is to connect multiple CPUs.

- Hardware refinement in pipelining, superscalar, and the like are non scalable and require sophisticated compiler technology. Evolving like compiler technology is a hard task.

Hardware architectures for parallel processing

- The basic elements of parallel processing are CPUs. The number of instruction and data streams that can be processed at the same time, computing systems are categorize into the following:

1. Single-instruction, single-data (SISD)
2. Single-instruction, multiple-data (SIMD)
3. Multiple-instruction, single-data (MISD)
4. Multiple-instruction, multiple-data (MIMD)

1. Single-instruction, single-data (SISD) systems

- An SISD computing system is a uniprocessor machine capable of executing a single instruction,which operates on a single data stream shown in figure. 1.
- In SISD, machine instructions are processed linearly, therefore computers that acquire this model are popularly called sequential computers.
- Most conventional computers are developed using the SISD model. All the instructions and data to be handled have to be stored in primary memory.
- The rate of the processing element in the SISD model is restricted by the rate at which the computer can transfer information internally.
- Presiding representative SISD systems are IBM PC, Macintosh, and workstations.

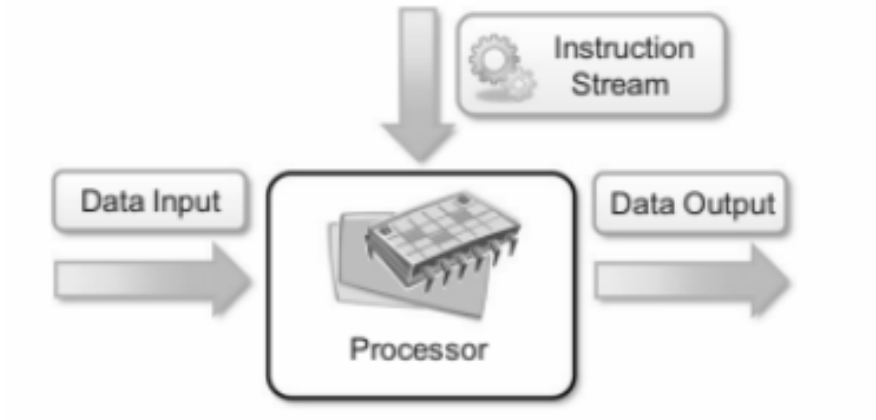


Fig. 1 Single-instruction, single-data (SISD) architecture.

2. Single-instruction, multiple-data (SIMD) systems

- SIMD computing system is a multiprocessor machine that executes the same instruction on all the CPUs but utilizes different data streams which is shown in figure 2.
- Machines based on an SIMD model are well matched to scientific computing since they imply lots of vector and matrix operations.
- For example, statements such as

Ci = Ai * Bi

can be passed to all the processing elements, arranged data elements of vectors A and B can be divided into multiple sets, and each processing element can process one data set. presiding representative SIMD systems are Cray's vector processing machine and Thinking Machines' cm

.**Multiple-instruction, single-data (MISD) systems**

- MISD computing system is a multiprocessor machine efficient of executing different instructions on different processing elements but all of them operating on the same data set shown in figure 3. For example, statements such as

y = sin(x) + cos(x) + tan(x)

Which carry out different operations on the same data set. Machines built using the MISD model are not beneficial in most of the applications; a few machines are assembled, but none of them are accessible commercially. They became more of an intellectual effort than a practical configuration.

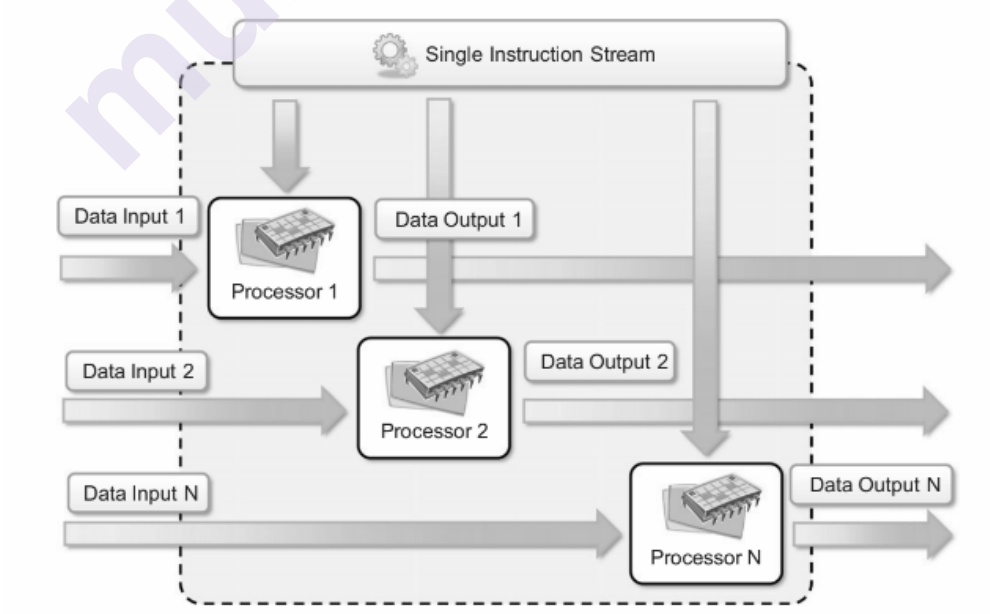


Fig. 2 Single-instruction, multiple-data (SIMD) architecture.

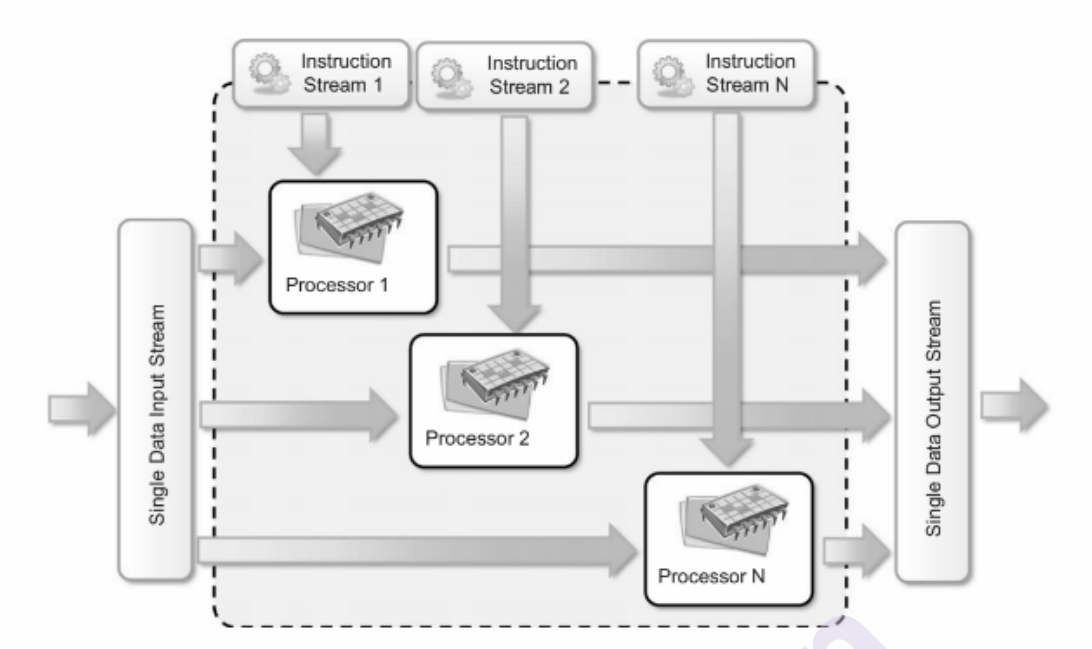


Fig. 3 Multiple-instruction, single-data (MISD) architecture.

3. Multiple-instruction, multiple-data (MIMD) systems

- A MIMD computing system is a multiprocessor machine ability of executing multiple instructions on multiple data sets shown in figure 4. Each processing element in the MIMD model has individual instruction and data streams, Therefore machines built using this model are well matched to any kind of application.
- As opposed to SIMD and MISD machines, processing elements in MIMD machines work asynchronously.
- MIMD machines are mainly classified into shared-memory MIMD and distributed-memory MIMD based on the way processing elements are coupled to the main memory.
- In the Shared memory MIMD machines, MIMD model, all the processing elements are connected to a single global memory and they all have access to it shown in figure 4.
- Systems established on this model are also called tightly coupled multiprocessor systems. The communication between processing elements in this model takes place through the shared memory; modification of the data stored in the global memory by one processing element is noticeable to all other processing elements.
- Presiding representative shared memory MIMD systems are Silicon Graphics machines and Sun/IBM's Symmetric Multi-Processing.

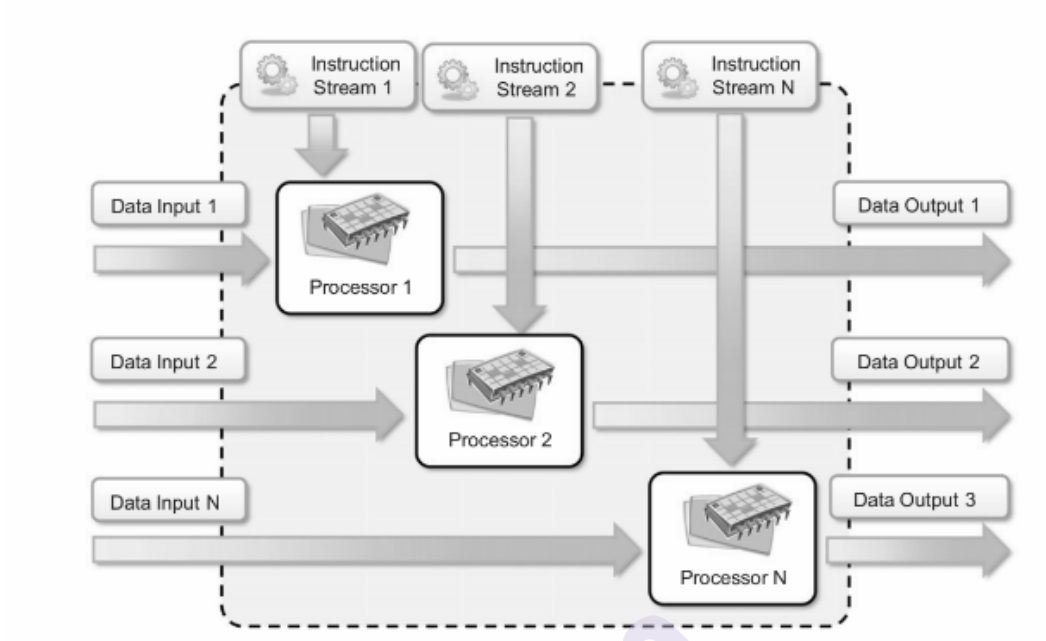


Fig. 4 Multiple-instructions, multiple-data (MIMD) architecture.

Approaches to parallel programming

- A sequential program is one that executes on a single processor and has a single line of control.
- To make many processors cumulative work on a single program, the program must be split into smaller independent chunks so that each processor can work on separate chunks of the problem.
- The program decay in this way is a parallel program. An extensive variety of parallel programming approaches are available.
- The most important between them are the following:
  - Data parallelism
  - Process parallelism
  - Farmer-and-worker model
- These types of models are all suitable for task level parallelism.
- In the point of data parallelism, the divide and conquer technique is used to divide data into multiple sets, and each data set is processed on different processing elements using the identical instruction.
- This approach is highly compatible for processing on machines based on the SIMD model. In the case of process parallelism, a given operation has multiple activities that can be processed on multiple processors.
- In the example of the farmer and worker model, a task distribution approach is used: one processor is designated as master and all other

remaining processing elements are designated as slaves; the master allocates jobs to slave processing elements and, on fulfillment, they tell the master, which in turn collects results.

Levels of parallelism

- Levels of parallelism are marked based on the lumps of code (like a grain size) that can be a probable candidate for parallelism. Below Table lists the categories of code granularity for parallelism.
- All these approaches have a common goal:
  - To boost processor efficiency by hiding latency.
  - To conceal latency, there must be another thread ready to run every time a lengthy operation occurs.

The plan is to execute concurrently two or more single-threaded applications, such as compiling, text formatting, database searching, and device simulation.

- As shown in the table and depicted in figure 5, parallelism within an application can be discovered at several levels.
  - Large grain (or task level)
  - Medium grain (or control level)
  - Fine grain (data level)
  - Very fine grain (multiple-instruction issue)

Levels of Parallelism

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

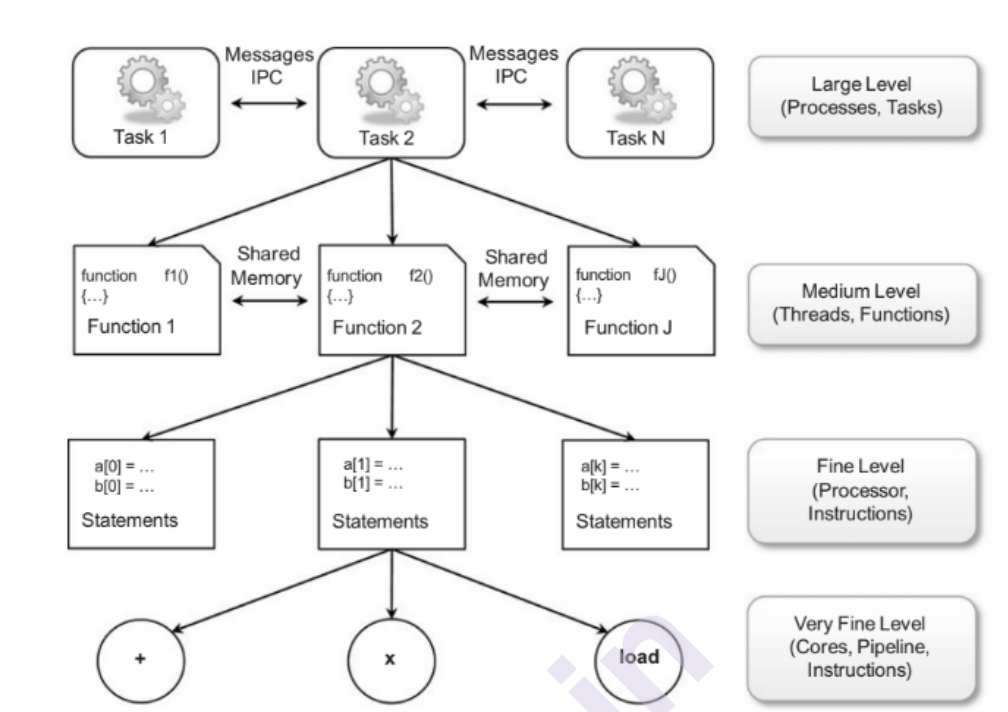


Fig. 5 Levels of parallelism in an application.

2.3 ELEMENTS OF DISTRIBUTED COMPUTING

- In elements of distributed computing, extend these concepts and explore how multiple activities can be done by exploiting systems collected from multiple heterogeneous machines and systems.
- Here we will learn the most common guidelines and patterns for implementing distributed computing systems from the perspective of the software designer.
- Distributed computing learning the models, architectures, and algorithms used for building and managing distributed systems.
- "A distributed system is a collection of independent computers that appears to its users as a single coherent system."
- The various types of distributed computing systems that are mainly focused on unified usage and aggregation of distributed resources.
- Communication is a fundamental feature of distributed computing.
- Distributed systems are composed of more than one computer that participate together, it is necessary to enable some sort of data and information exchange between them, which generally occurs through the network
- A distributed system is one in which components discovered at networked computers communicate and coordinate their actions only by passing messages.

- The components of a distributed system communicate with some sort of message passing. This is a term that encloses several communication models.

Components of a distributed system

- A distributed system is the result of the interconnection of several components that traverse the entire computing stack from hardware to software.
- It comes out from the collaboration of several elements that by working in conjunction give users the illusion of a single coherent system.
- Figure 6 shows an overview of the different layers that are involved in providing the services of a distributed system.

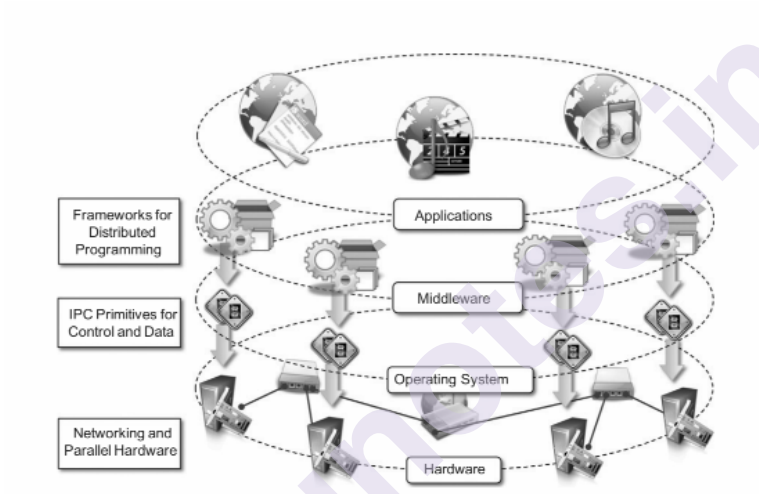


Fig. 6 A layered view of a distributed system.

- At the bottom layer, computer and network hardware represent the physical infrastructure, these components are directly controlled by the operating system, which provides the basic services for interprocess communication (IPC), process scheduling and management, and resource management in terms of file system and local devices.
- The middleware layer strengthens such services to build a uniform environment for the development and deployment of distributed applications. This layer supports the programming paradigms for distributed systems.
- The top of the distributed system stack is acted by the applications and services designed and developed to use the middleware. These can obey several purposes and often expose their features in the form of graphical user interfaces (GUIs) approachable locally or through the Internet via a Web browser.

Architectural styles for distributed computing

- Architectural styles are mainly used to find the vocabulary of components and connectors that are used as instances of the style together with a set of constraints on how they can be combined.
- Architectural styles for distributed systems are helpful in understanding the different roles of components in the system and how they are distributed across multiple machines.
- Organization of the architectural styles into two major classes:
  - Software architectural styles
  - System architectural styles
- The first class has the relation to the logical organization of the software, the second class contains all those styles that express the physical organization of distributed software systems in terms of their major components.

Component and connectors

- These are the basic elements with which architectural styles are defined.
- A component represents a unit of software that encapsulates a function or a feature of the system.
- Examples of components can be programs, objects, processes, pipes, and filters.
- A connector is a communication mechanism that enables cooperation and coordination among components.
- Differently from components, connectors are not encapsulated in a single entity, but they are implemented in a distributed manner over many system components.

Software architectural styles

- Software architectural styles are based on the logical arrangement of software components.
- They are helpful because they provide an intuitive view of the whole system, despite its physical deployment.
- They also recognize the main abstractions that are used to shape the components of the system and the expected interaction patterns between them.
- These models represent the foundations on top of which distributed systems are designed from a logical point of view.
- Architectural styles are categorized as shown below.

Category	Most Common Architectural Styles
Data-centered	Repository Blackboard
Data flow	Pipe and filter Batch sequential
Virtual machine	Rule-based system Interpreter
Call and return	Main program and subroutine call/top-down systems Object-oriented systems Layered systems
Independent components	Communicating processes Event systems

2.4 TECHNOLOGIES FOR DISTRIBUTED COMPUTING

2.4.1 Remote procedure call

- RPC is the fundamental abstraction that allows the execution of procedures on client's request.
- RPC allows increasing the concept of a procedure call completely outside the boundaries of a process and a single memory address space.
- The called procedure and calling procedure may reside on the same system or they may be on different systems in a network.
- Figure 7 shows the crucial components that enable an RPC system.

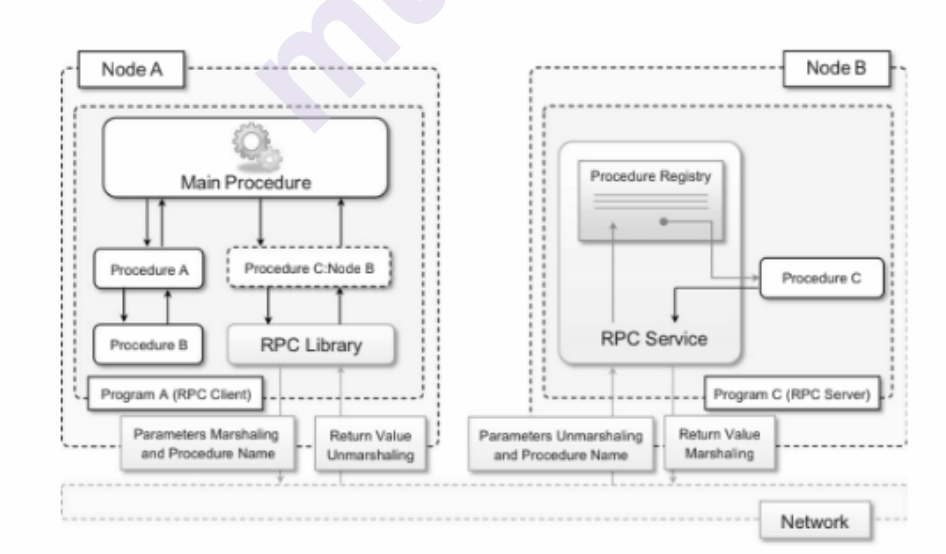


Fig. 7 The RPC reference model.

- Therefore, developing a system strengthening RPC for IPC contain the following steps:
  - Design and implementation of the server procedures that will be uncovered for remote invocation.
  - Registration of remote procedures with the RPC server on the node where they will be made accessible.
  - Design and implementation of the client code that invokes the remote procedures (RPC).

2.4.2 Distributed object frameworks

1. Distributed object frameworks enhance object-oriented programming systems by allowing objects to be distributed across a heterogeneous network and provide facilities so that they can clearly act as though they were unavailable on the same address space.
2. Distributed object frameworks strengthen the basic mechanism introduced with RPC and extend it to enable the remote invocation of object methods and to keep watch on references to objects made available through a network connection.
3. With respect to the RPC model, the infrastructure manages types that are exposed through well known interfaces rather than procedures. Therefore, the common interaction pattern will be like this:

a. The server process keeps track of a registry of active objects that are made available to other processes. On the report of the specific implementation, active objects can be published using interface definitions / class definitions.

b. The client process, with a given addressing scheme, obtains a reference to the active remote object. This reference is acted by a pointer to an instance that is of a shared type of interface and class definition.

c. The client process invokes the methods on the active object by calling them through the reference previously secured. Parameters and return values are marshaled to take place in the case of RPC.

Object activation and lifetime

- The life span of an object instance is a crucial element in distributed object-oriented systems.
- The single memory address space scenario, objects are definitely created by the programmer, and their references are made available by passing them from one object instance to another.
- A distributed framework introduces additional issues that require a different management of the lifetime of objects revealed through remote interfaces.

Common object request broker architecture (CORBA)

- "CORBA is a specification described by the Object Management Group (OMG) for providing cross platform and cross-language interoperability between distributed components." The specification was fundamentally designed to provide an interoperation standard that could be effectively used at the industrial level.
- A main and important component in the CORBA architecture is the Object Request Broker (ORB), which reacts as a central object bus.
- A CORBA object registers with the ORB the interface it is uncovering, and clients can obtain a reference to that interface and invoke methods on it.

Distributed component object model (DCOM/COM1)

- DCOM, behind time integrated and developed into COM1, is the solution provided by Microsoft for distributed object programming before the introduction of .NET technology.
- DCOM allows a set of features allowing the use of COM components beyond the process boundaries.
- A COM object recognizes a component that encapsulates a set of and related operations; it was designed to be easily clubbed into another application to strengthen the features disclosed through its interface.

Java remote method invocation (RMI)

- Java RMI is a standard technology provided by Java Oracle for enabling RPC call among distributed Java objects.
- RMI defines an infrastructure that enables the invocation of methods on objects that are found on different Java Virtual Machines (JVMs) residing either on the local node or on a remote one.

.NET remoting

- .NET Remoting is the technology enabling IPC among .NET applications.
- It provides developers with a uniform platform for retrieving remote objects from within any application developed in any of the languages supported by .NET.

Service-oriented computing

- Service oriented computing arrange distributed systems in terms of services, which represent the great abstraction for building systems.

- Service orientation expresses applications and software systems as aggregations of services that are correlated within a service-oriented architecture (SOA).
- A service encapsulates a software component that enables a set of coherent and related functionalities that can be reused and integrated into huge and more complex applications. The term service is a general abstraction that encompasses various different implementations using different technologies and protocols.
- Four major characteristics that identify a service:

1. Boundaries are explicit.
2. Services are autonomous
3. Services divide the schema and contracts, not class or interface definitions.
4. Service compatibility is determined based on policy.

Service-oriented architecture

- SOA is an architectural style supporting service orientation.
- It arranges a software system into a collection of interacting services.
- SOA encloses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
- SOA based computing packages functionalities into a set of interoperable services, which can be non-discriminatory into different software systems belonging to individual business domains.
- The following guiding principles which characterize SOA platforms, are winning features within an enterprise context:
  - Standardized service contract.
  - Loose coupling
  - Abstraction.
  - Reusability.
  - Autonomy
  - Lack of state
  - Discoverability
  - Composability

2.5 LET US SUM UP

- Parallel and distributed computing emerged as a solution for solving complex
- Parallel computing introduces models and architectures for performing multiple tasks within a single computing node or a set of tightly coupled nodes with homogeneous hardware.
- Parallelism is achieved by leveraging hardware capable of processing multiple instructions in parallel.
- Distributed systems constitute a large umbrella under which several different software systems are classified.

2.6 LIST OF REFERENCES

- Mastering Cloud Computing, RajkumarBuyya, Christian Vecchiola, S ThamaraiSelvi, Tata McGraw Hill Education Private Limited, 2013.

2.7 UNIT END EXERCISES

1. What is the difference between parallel and distributed computing?
2. What is a SIMD architecture?
3. Describe the different levels of parallelism that can be obtained in a computing system.
4. What is a distributed system? What are the components that characterize it?

3

CLOUD COMPUTING ARCHITECTURE

Unit Structure

3.0 OBJECTIVE

- To understand the architecture of cloud computing.
- To understand the different types of cloud computing servcies.
- To understand the enterprise architecture used in cloud computing.
- To study the different types of clouds.

3.1 INTRODUCTION

- Cloud Computing can be defined as the exercise of using a network of remote servers hosted on the Internet to store, manage, and process data, alternatively a local server or a may be a personal computer.
- Organizations offering such types of cloud computing services are called cloud providers and charge for cloud computing services based on their usage.
- Grids and clusters are the base for cloud computing.

3.2 CLOUD COMPUTING ARCHITECTURE

- In the representation, a cloud is implemented using a datacenter, a collection of clusters, or a heterogeneous distributed system which is composed of desktop personal computers, workstations, and servers.
- Clouds are built which depend on one or more data centers.

- When we deliver the specific service to the end user, different layers can be stacked on top of the virtual infrastructure like a virtual machine manager, a development platform, or a specific application middleware.
- The cloud computing paradigm came out as an output of the convergence of various existing models, technologies, and concepts that switch the way we deliver and use IT services.
- A definition of Cloud computing:

"Cloud computing is a utility-oriented and Internet-centric way of delivering IT services on demand. These services cover the entire computing stack: from the hardware infrastructure packaged as a set of virtual machines to software services such as development platforms and distributed applications."

3.3 THE CLOUD REFERENCE MODEL

- IT service can be consumed as a utility and delivered through a network
- Cloud computing supports these IT services, most likely the Internet.
- The characterization in cloud computing includes various aspects: infrastructure, development platforms, application and services.

4.3.1 Architecture of Cloud Computing

- The cloud computing a layered view covering the whole stack from hardware appliances to software systems.
- Cloud resources in this layer are implemented using a datacenter in which hundreds and thousands of nodes are stacked in conjunction.
- Cloud infrastructure can be heterogeneous in nature because it contains a variety of resources, like clusters and even networked personal computers, can be used to build it.
- Database systems and other storage services can also be a portion of the infrastructure.
- The physical infrastructure is handled by the middleware, the objectives of which are to provide a suitable runtime environment for applications and to utilize the resources.
- At the bottom layer of the stack, virtualization technologies are used to assure runtime environment customization, application isolation, sandboxing, and quality of service.
- Hardware virtualization is one of the most commonly used at this level.
- Hypervisors control the small resources and show the distributed infrastructure as a collection of virtual machines.

- With the use of virtual machine technology it is enable to elegantly partition the hardware resources for example CPU and memory and to virtualize specific devices, thus fulfilling the requirements of users and applications.

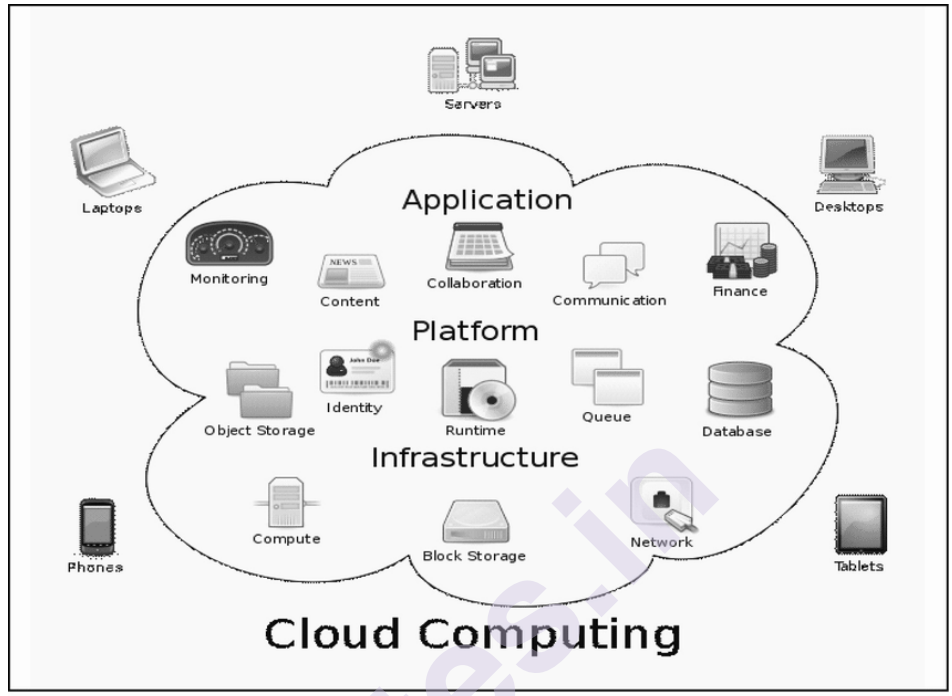


Fig. 1 The cloud computing architecture.

- Cloud computing solutions are generally matched with storage and network virtualization plans of action, which enable the infrastructure to be fully virtualized and controlled.
- As per the particular service provided to the end users, other virtualization technology can be used, for example, programming-level virtualization helps in creating a flexible runtime environment where applications can be executed and controlled.
- This implies that applications hosted on the cloud developed with a specific type of technology or a programming language, such as Java, .NET, or Python.
- Infrastructure management is the main function of core middleware, which provides support such as debate of the quality of service, admission control, execution management and monitoring, accounting, and billing.
- The cloud hosting platforms and resources are generally classified as a
  - Infrastructure-as-a-Service (IaaS) solution.

It provides both the management layer and the physical infrastructure; some of others provide the management layer.. In this second case, the

management layer is frequently integrated with other IaaS solutions that provide physical infrastructure and adds value.

- IaaS solutions are appropriate for designing the system infrastructure but it provides limited services to develop applications. This type of service is provided by cloud programming environments and tools,
- The span of tools consist of Web-based interfaces, command-line tools, and frameworks for concurrent and distributed programming. In this outline, users develop their applications specifically for the cloud by using the API reveal at the user-level middleware. For this reason, this method is known as Platform-as-a-Service.
- The top most layer of the reference model consists of services delivered at the application level. These are called Software-as-a-Service.
- Web-based applications depend on the cloud to provide service to end users. The cloud provided by IaaS and PaaS solutions permit independent software vendors to deliver their application services over the Internet.
- Behavior automatically is an implementation of SaaS which should feature, whereas PaaS and IaaS provide this functionality as a part of the API shows to users.
- The reference model also describes the concept of everything as a Service. This is one of the most prime elements of cloud computing: Cloud services from different providers can be combined to provide a completely large integrated solution covering all the computing stack of a system.

3.4 CLOUD COMPUTING SERVICES: SAAS, PAAS, IAAS

Software as a service (SaaS)

- Software-as-a-Service (SaaS) is a software delivery model that gives access to applications through the Internet as a Web Based service.
- It provides free users from complex hardware and software management by offloading such tasks to third parties, which build applications accessible to multiple users through a Web browser.
- Example:customers neither need to install anything on their premises nor have to pay costs to buy the software and the need licenses. They directly access the application website, enter their username and password and other billing details, and can immediately use the application.
- On the source side, they keep maintaining specific details and features of each customer's infrastructure and make it available on user request.

- SaaS is a software delivery model, (one-to-many) whereby an application is shared across multiple users.
- Example includes CRM3 and ERP4 applications that add up common needs for almost all enterprises, from small to medium-sized and large businesses.
- This structure relives the development of software platforms that provide a general set of features and support specialization and ease of integration of new components.
- SaaS applications are naturally multitenant.
- The term SaaS was then invented in 2001 by the Software Information & Industry Association (SIIA).
- The analysis done by SIIA was mainly aligned to cover application service providers (ASPs) and all their variations, which imprison the concept of software applications consumed as a service in a wide sense.
- ASPs Core characteristics of SaaS:
  - The product sold to customers is an application approach.
  - The application is centrally managed.
  - The service delivered is one-to-many.
  - The service provides is an integrated solution delivered on the contract, which means provided as promised

Platform as a service

- Platform-as-a-Service (PaaS) which provides a development and deployment platform for running applications in the cloud.
- They compose the middleware on top of which applications are built.
- Following figure shows a general overview of the features characterizing the PaaS.

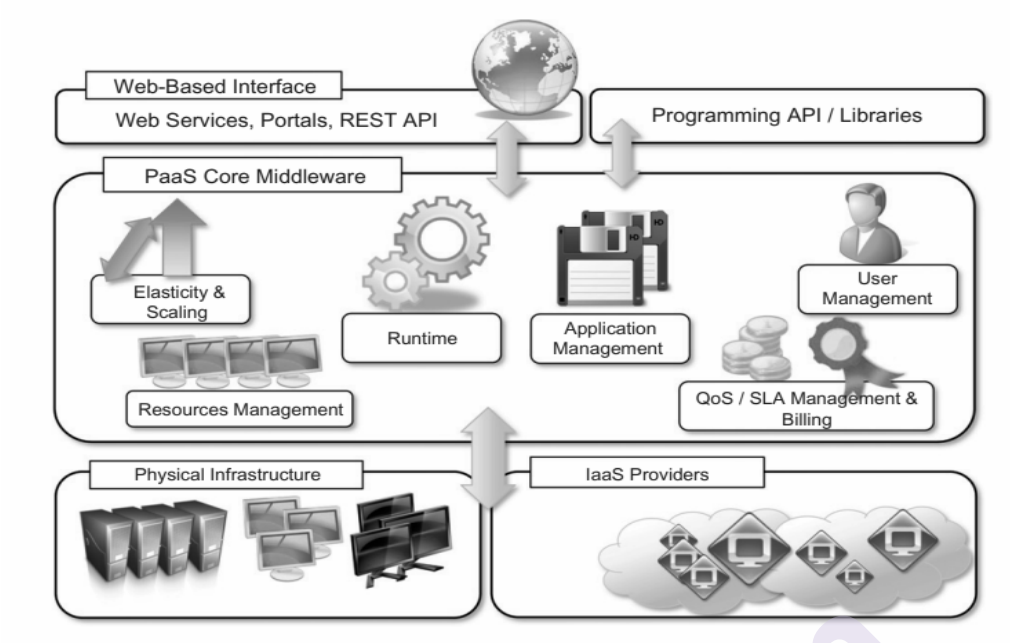


Fig.2. The Platform-as-a-Service reference model

- Application management is the key functionality of the middleware systems.
- PaaS provides applications with a runtime environment and does not shows any service for managing the underlying infrastructure.
- They automate the process of deploying applications to the infrastructure, configuring application components, provisioning and configuring supporting technologies such as load balancers and databases, and managing system change based on policies set by the user.
- The core middleware is responsible for managing the resources and scaling applications on demand, according to the adherence made with users.
- The core middleware exposes interfaces that enable programming and installing applications on the cloud.
- The PaaS model provides a complete object model for representing an application and provides a programming language-based approach.
- In this point the traditional development environments can be used to design and develop applications, which are then deployed on the cloud by using the APIs revealed by the PaaS provider.
- PaaS offers middleware for developing applications together with the infrastructure.

Infrastructure as a service or hardware as a service

- Infrastructure as a Service is the most popular model and developed market segment of cloud computing.
- They deliver customizable infrastructure on request.
- The IaaS offers single servers for entire infrastructures, including network devices, load balancers, and database and Web servers.
- The main aim of this technology used to deliver and implement these solutions is hardware virtualization:
  - one or more virtual machines configured and interconnected
  - Virtual machines also constitute the atomic components that are installed and charged according to the specific features of the virtual hardware:
    - memory
    - number of processors, and
    - disk storage
- IaaS shows all the benefits of hardware virtualization:
  - workload partitioning
    - application isolation
    - sandboxing, and
    - hardware tuning
- HaaS allows better utilization of the IT infrastructure and provides a more safe environment for executing third party applications.
- Figure 2 shows a total view of the components setup an Infrastructure-as-a-Service.
- It is possible to identified three principal layers:
  - the physical infrastructure
  - the software management infrastructure
  - the user interface
- At the top layer the user interface allow to access the services exposed by the software management infrastructure. This types of an interface is generally based on Web technologies:
  - Web services
  - RESTful APIs, and

- mash-ups

- These automation allow applications or final users to access the services exposed by the underlying infrastructure.

- A central role of the scheduler, is in charge of allocating the execution of virtual machine instances. The scheduler communicates with the other components that perform a variety of tasks.

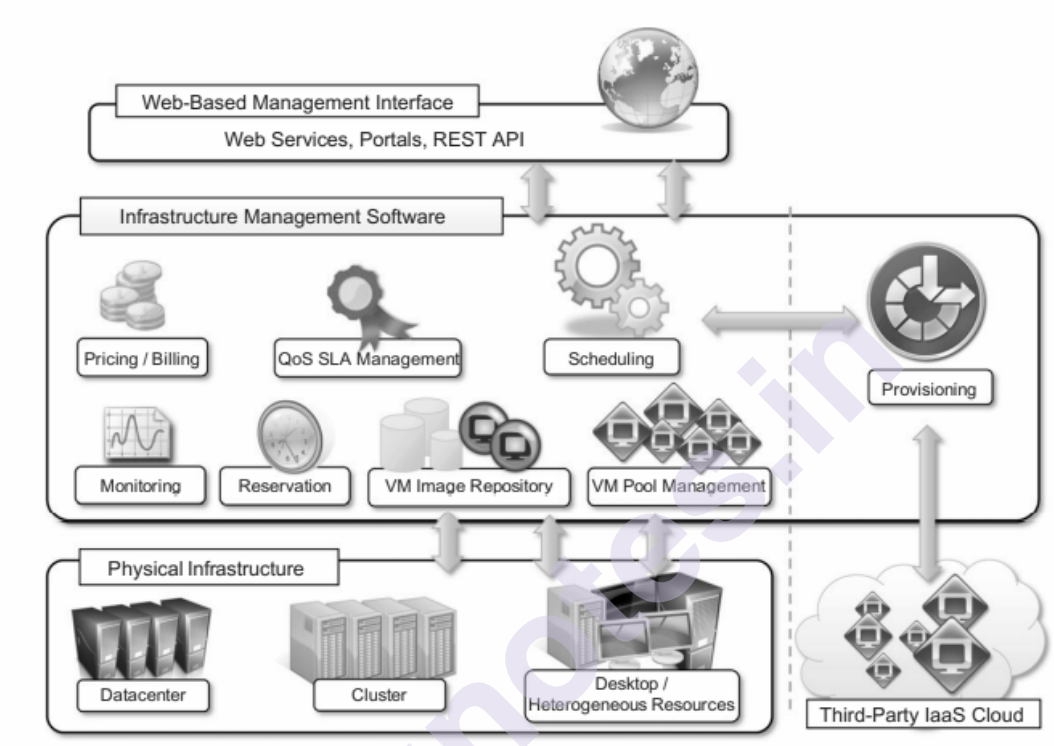


Fig.3. Infrastructure-as-a-Service reference implementation

3.5 TYPES OF CLOUDS

- Clouds compose the primary outcome of cloud computing.

- They are a type of parallel and distributed system, physical and virtual computers conferred as a unified computing asset.

- Clouds set up the infrastructure on top of services that are implemented and delivered to customers. Such infrastructures can be of different types and provide useful information about the nature and the services offered by the cloud.

- A more convenient classification is given according to the administrative domain of a cloud: It identifies the boundaries within which cloud computing services are implemented, provides hints on the underlying infrastructure take on to support such services, and qualifies them. It is then possible to evolve four different types of cloud:

- Public clouds.
  - The cloud is open to the wider public.
- Private clouds.
  - The cloud is executed within the private property of an institution and generally made accessible to the members of the institution
- Hybrid clouds.
  - The cloud is a combination of the two previous clouds and most likely identifies a private cloud that has been augmented with services hosted in a public cloud.
- Community clouds.
  - The cloud is distinguished by a multi administrative domain consisting of different deployment models (public, private, and hybrid).

Public clouds

- Public clouds account for the first expression of cloud computing.
- They are an awareness of the canonical view of cloud computing in which the services provided are made available to anyone, from anywhere, and at any time through the Network.
- From a structural point of view they are a distributed system, most likely composed of one or more data centers connected together, on top of which the specific services offered by the cloud are implemented.
- Any customer can easily agree with the cloud provider, enter her username and password and billing details.
- They extend results to reduce IT infrastructure costs and serve as a viable option for handling peak loads on the local infrastructure.
- They are used for small enterprises, which are able to initiate their businesses without large up-front investments by completely relying on public infrastructure for their IT needs.
- A public cloud can recommend any type of service such as infrastructure, platform, or applications. For example, Amazon EC2 is a public cloud that delivers infrastructure as a service. Google AppEngine is also called public cloud that provides an application development platform as a service; and SalesForceservice.com is a public cloud that provides software as a service.

Private clouds

- Private clouds, which are the same as public clouds, but their resource provisioning model is restricted within the boundaries of an organization.
- Private clouds have the benefit of keeping the core business operations in house by depending on the existing IT infrastructure and reducing the cost of maintaining it once the cloud has been set up.
- The private cloud can provide services to a different range of users.
- private clouds is the possibility of testing applications and systems at a comparatively less price rather than public clouds before implementing them on the public virtual infrastructure.
- The main advantages of a private cloud computing infrastructure:

1. Customer information protection.
2. Infrastructure ensuring SLAs.
3. Compliance with standard procedures and operations.

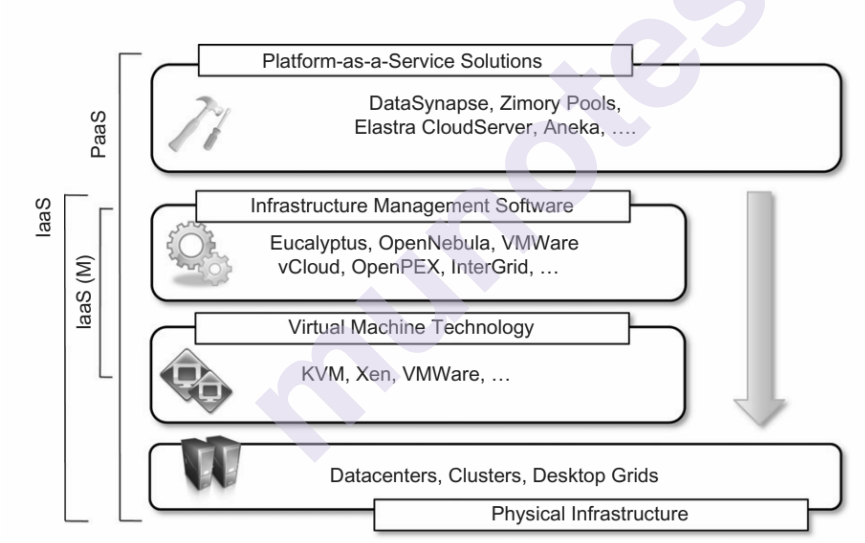


Fig.4. Private Clouds hardware and software stack.

Hybrid clouds

- A hybrid cloud could be an attractive opportunity for taking advantage of the best of the private and public clouds. This shows the development and diffusion of hybrid clouds.
- Hybrid clouds enable enterprises to utilize existing IT infrastructures, maintain sensitive information within the area, and naturally increase and reduce by provisioning external resources and releasing them when they're no longer needed.

- Figure 5 demonstrate the a general overview of a hybrid cloud:
  - It is a heterogeneous distributed system consisting of a private cloud that integrates supplementary services or resources from one or more public clouds.
  - For this intention they are also called heterogeneous clouds.
  - Hybrid clouds look into scalability issues by leveraging external resources for exceeding

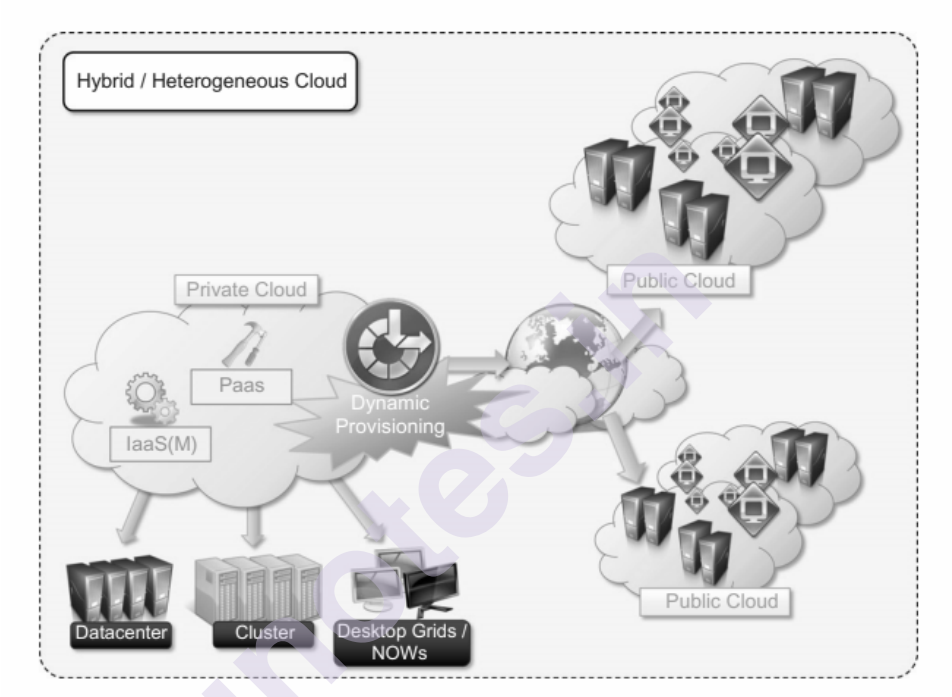


Fig.5. Hybrid/heterogeneous cloud overview.

Community clouds

- Community clouds are distributed systems created by integration of services of different clouds to handle the specific requirement of an industry, a community, or a business sector.
- The National Institute of Standards and Technologies characteristic community clouds as follows:
  - The infrastructure is shared by different organizations and supports a certain community that has shared concerns.
  - It may be controlled by the organizations or a third party and may exist on premise or off premise.

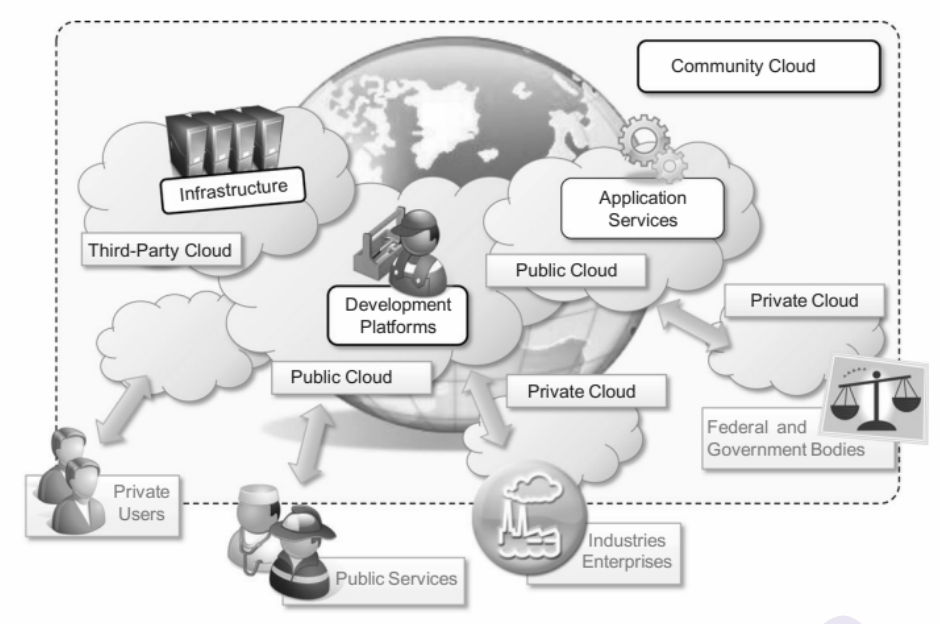


Fig. 6 general view community clouds

- Figure 6 shows a view of the usage scenario of community clouds, jointly with reference architecture.
- The users of a distinct community cloud fall into a well identified community, sharing the same concerns or needs such as government bodies, industries, or even simple users, but all of them concentrate on the same problem for their interaction with the cloud.
- Community clouds are the services that are generally delivered within the institution that owns the cloud.
- Candidate district for community clouds are as follows:

○ **Media industry**

■ Where Companies are finding low-cost, agile, and simple solutions to better the efficiency of content production.

■ Most media involve an expanded ecosystem of partners.

■ Community clouds can provide a shared environment where services can ease business to business participation and give the horsepower in terms of aggregate bandwidth, CPU, and storage required to efficiently support media production.

○ **Healthcare industry.**

■ In the healthcare industry, there are different storyline in which community clouds are used.

■ Community clouds provide a global platform on which to share information and knowledge without telling sensitive data maintained within the private infrastructure.

- The naturally hybrid deployment model of community clouds supports the storing of patient data in a private cloud while using the shared infrastructure for noncritical services and automating processes within hospitals.

- **Energy and other core industries.**

- These industries concern different service providers, vendors, and organizations, a community cloud can give the right type of infrastructure to create an open and upright market.

- **Public sector.**

- The public sector can limit the adoption of public cloud offerings.

- governmental processes involve several institutions and agencies

- Aimed at providing strategic solutions at local, national, and international administrative levels.

- involve business-to-administration, citizen-to-administration, and possibly business-to-business processes.

- Examples, invoice approval, infrastructure planning, and public hearings.

- **Scientific research.**

- THis is an interesting example of community clouds.

- In this point, the common interest in handling and using different organizations to split a large distributed infrastructure is scientific computing.

The Advantages of community clouds:

- Openness.

Clouds are open systems in which fair competition between different solutions can occur.

- **Community.**

Providing resources and services, the infrastructure turns out to be more scalable.

- **Graceful failures.**

There is no single provider & vendor in control of the infrastructure, there is no chance of a single point of failure.

- **Convenience and control.**

There is no dispute between convenience and control because the cloud is shared and owned by the community, which makes all the decisions through a collective representative process.

- **Environmental sustainability.**

These clouds tend to be more organic by increasing and shrinking in a symbiotic relationship to support the demand of the community.

3.6 SUMMARY

- Three service models. Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS).
- Four deployment models. Public clouds, private clouds, community clouds, and hybrid clouds.
- Cloud computing has been rapidly adopted in industry, there are several open research challenges in areas such as management of cloud computing systems, their security, and social and organizational issues.

3.7 REFERENCE FOR FURTHER READING

- Enterprise Cloud Computing Technology, Architecture, Applications, GautamShroff, Cambridge University Press, 2010
- Mastering In Cloud Computing, RajkumarBuyya, Christian Vecchiola And ThamariSelvi S, Tata Mcgraw-Hill Education, 2013
- Cloud Computing: A Practical Approach, Anthony T Velte, Tata Mcgraw Hill, 2009

3.8 UNIT END EXERCISES

1. What does Infrastructure-as-a-Service refer to?
2. What are the main characteristics of a Platform-as-a-Service solution?
3. What does the acronym SaaS mean? How does it relate to cloud computing?
4. Classify the various types of clouds.
5. Give an example of the public cloud.

4

VIRTUALIZATION

Unit Structure

4.0 OBJECTIVE

- To understand the fundamental components of cloud computing
- To study the application running on an execution environment using virtualization.
- To understand the different virtualization techniques.
- To study the characteristics of Virtualized Environments.

4.1 INTRODUCTION

- Virtualization is a large universe of technologies and concepts of an abstract environment whether virtual hardware or an operating system to run applications.
- The word virtualization is often synonymous with hardware virtualization, which plays a fundamental role in effectively delivering Infrastructure-as-a-Service (IaaS) solutions for cloud computing.
- Virtualization technologies provide virtual environments at the operating system level, the programming language level, and the application level.
- Virtualization technologies provide a virtual environment for not only executing applications but also for storage, memory, and networking.

1. **Increased performance and computing capacity**.

a. At present, the average end-user desktop computer is powerful enough to meet almost all the requirements of everyday computing, with extra capacity that is rarely used.

b. All these desktop computers have resources enough to host a virtual machine manager and execute a virtual machine with by far acceptable performance.

c. The same deliberation applies to the high-end side of the PC market, where supercomputers can provide huge compute power that can make room for the execution of hundreds or thousands of virtual machines.

2. **Underutilized hardware and software resources.**

a. Hardware and software underutilization is occurring due to

i. Increased performance and computing capacity

ii. The effect of finite use of resources.

3. **Lack of space**

a. The ongoing need for additional capacity, whether storage or compute power, makes data centers grow rapidly.

b. Companies such as Google and Microsoft expand their infrastructures by building data centers as large as football fields that are able to host thousands of nodes.

c. Although this is viable for IT giants, in most cases enterprises cannot afford to build another data center to accommodate additional resource capacity.

d. This condition, along with hardware underutilization, has led to the diffusion of a technique called server consolidation,1 for which virtualization technologies are fundamental.

4. **Greening initiatives**.

a. Nowadays, companies are increasingly looking for ways to reduce the amount of energy they consume and to reduce their carbon footprint.

b. Data centers are one of the major power consumers; they contribute consistently to the impact that a company has on the environment.

c. Preserving a data center operation not only involves keeping servers on, but a great deal of energy is also consumed in keeping them cool.

d. Infrastructures for cooling have a significant impact on the carbon footprint of a data center.

e. Hence, reducing the number of servers through server consolidation will surely reduce the impact of cooling and power consumption of a data center. Virtualization technologies can provide an efficient way of consolidating servers.

5. **Rise of administrative costs**.

a. Power consumption and cooling costs become higher than the cost of IT equipment nowadays.

b. The increased demand for extra capacity, which translates into more servers in a data center.

c. Which is responsible for a significant increment in administrative costs.

d. Common system administration duties consist of hardware monitoring, defective hardware replacement, server setup and updates, server resources monitoring, and backups.

e. These are labor-intensive operations, and the higher the number of servers that have to be managed, the higher the administrative costs.

f. Virtualization helps to reduce the number of required servers for a given workload, thus reducing the cost of the administrative manpower.

4.2 CHARACTERISTICS OF VIRTUALIZED ENVIRONMENTS

- Virtualization is a big concept that refers to the creation of a virtual version of something, whether hardware, a software environment, storage, or a network.
- A virtualized environment has three major components: guest, host, and virtualization layer.
- The guest acts for the system component that communicates with the virtualization layer rather than with the host, as would normally happen.
- The host shows the original environment where the guest is expected to be managed.
- The virtualization layer is controlling for recreating the same or a different environment where the guest will operate shown in figure 1.

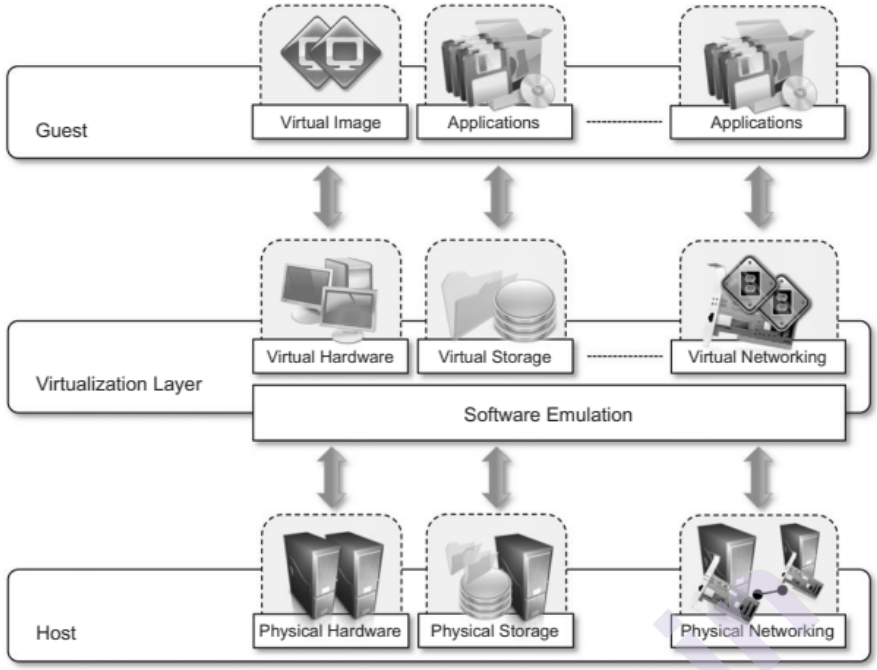


Figure 1. Virtualization reference model.

- The most instinctive and popular is represented by hardware virtualization, which also composes the original realization of the virtualization concept.
- In the hardware virtualization, the guest is represented by a system image made up of an operating system and installed applications. These are fixed on top of virtual hardware that is handled and managed by the virtualization layer, also called as the virtual machine manager. The host is represented by the physical hardware, and the operating system, that describe the environment where the virtual machine manager is running.
- In the case of virtual storage, the guests might be client applications or users that interact with the virtual storage management software deployed on top of the real storage system.
- Virtual networking is also similar as above: The guest applications and users communicate with a virtual network, such as a VPN, which is managed by specific software (also called VPN client) that employs the physical network available on the node. VPNs are functional for creating the illusion of being within a different physical network and thus acquiring the resources in it, which would otherwise not be available.

4.2.1 Increased security

- The capability to handle the execution of a guest in a completely clear manner opens new potential for delivering a secure, controlled execution environment.

- The virtual machine shows an emulated environment in which the guest is accomplished.
- All the operations of the guest are normally performed opposite to the virtual machine, which then translates and reflects them to the host.
- This level of dissimulation enables the virtual machine manager to control and filter the activity of the guest, thus avert some harmful operations from being performed.
- Resources exposed by the host can then be hidden or simply secured from the guest.
- Besides, sensitive information that is contained in the host can be naturally hidden without the need to install complex security policies.
- Increased security is a requirement when coordinating with untrusted code.
- The JVM and the .NET runtime provide substantial security policies for customizing the execution environment of applications.
- Hardware virtualization solutions like VMware Desktop, VirtualBox, and Parallels provide the ability to create a virtual computer with customized virtual hardware on top of which a new operating system can be installed.
- The file system exposed by the virtual computer is completely different from the one of the host machine.
- This is the perfect environment for running applications without influencing other users in the environment.

4.2.2 Managed execution

- Virtualization of the execution environment not only enables increased security, but a wider range of features also can be implemented.
- The sharing, aggregation, emulation, and isolation are the most applicable features which is shown in figure 2

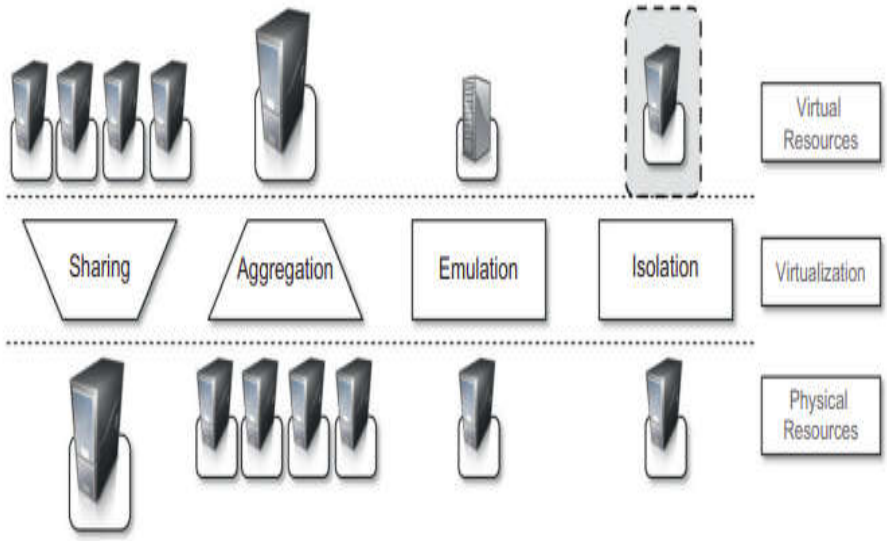


Figure 2. Functions enabled by managed execution.

1. **Sharing.**

a. Virtualization enables the formation of a separate computing environment within the same host.

b. In this way it is possible to fully utilize the capabilities of a powerful guest, which would otherwise be underutilized.

c. Sharing is an important feature in virtualized data centers, where this basic feature is used to reduce the number of active servers and limit power consumption.

2. **Aggregation.**

a. Virtualization also enables aggregation, which is the opposite action.

b. A group of different hosts can be bound together and represented to guests as a single virtual host.

c. This function is naturally implemented in middleware for distributed computing, with a classical example represented by cluster management software, which harnesses the physical resources of a homogeneous group of machines and represents them as a single resource.

3. **Emulation.**

a. Guest programs are executed inside an environment that is controlled by the virtualization layer, which ultimately is a program.

b. This enables controlling and tuning the environment that is revealed to guests. For illustration, a completely different environment with respect to the host can be emulated, thus allowing the execution of

guest programs needs specific characteristics that are not available in the physical host.

c. This feature becomes very important for testing purposes, where a specific guest has to be authenticated against different platforms or architectures and the wide range of options is not easily attainable during the development.

4. **Isolation.**

a. Virtualization enables guests whether they are OS, applications, or other entities with a completely different environment, in which they are carried out.

b. The guest program accomplishes its activity by interacting with an abstraction layer, which gives access to the underlying resources.

c. Isolation comes with several benefits; for example, it enables multiple guests to run on the one and the same host without interfering with each other.

d. Second, it provides a segregation between the host and the guest.

e. The virtual machine can sieve the activity of the guest and prevent harmful operations against the host.

4.2.3 Portability

- The concept of portability registers in different ways according to the specific type of virtualization considered.
- In the point of a hardware virtualization solution, the guest is filled into a virtual image that, in most cases, can be securely moved and executed on top of various virtual machines.
- Virtual images are generally exclusive formats that need a specific virtual machine manager to be executed. In the programming level virtualization, as implemented by the JVM or the .NET runtime, the binary code act for application components can be run without any recompilation on any implementation of the corresponding virtual machine.
- This makes the application development cycle more pliable and application deployment very straightforward: One version of the application, in most cases, is able to run on different platforms with no updates.
- Last, portability allows your own system to always be with you and ready to use as long as the required virtual machine manager is available.
- This requirement is, in general, less rigorous than having all the applications and services you need available to you anywhere you go.

4.3 TAXONOMY OF VIRTUALIZATION TECHNIQUES

- Virtualization covers an extensive range of emulation techniques that are applied to different areas of computing.
- A classification of these techniques helps us better recognize their characteristics and use shown in figure 3.

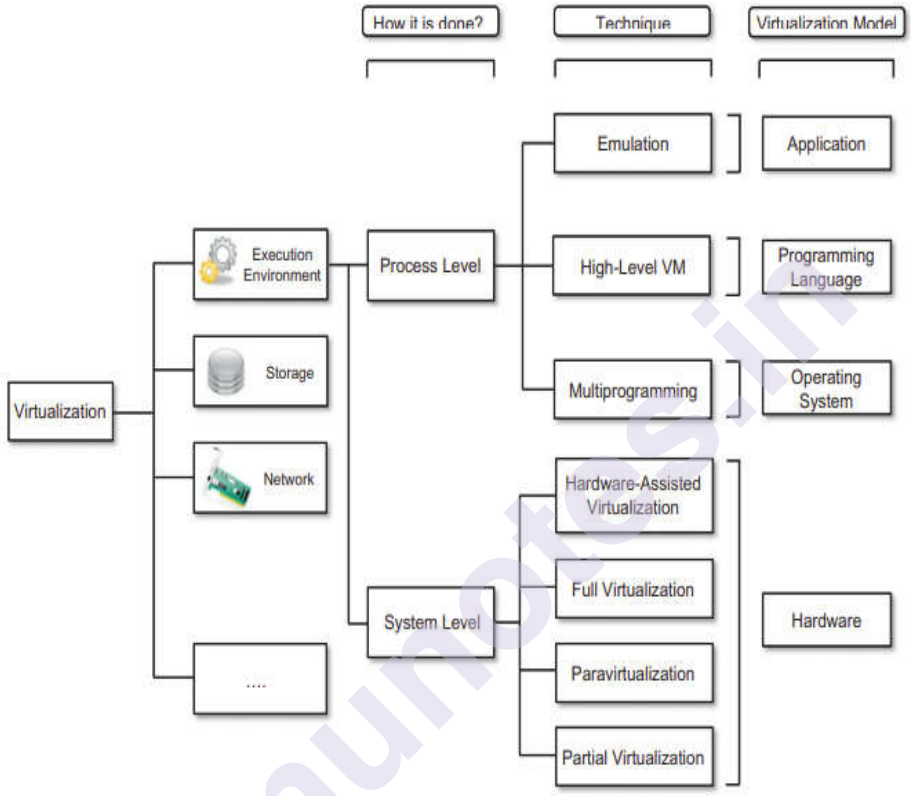


Figure 3. A taxonomy of virtualization techniques.

- The first classification disfavours against the service or entity that is being emulated. Virtualization is generally used to emulate execution environments, storage, and networks.
- Among these categories, execution virtualization constitutes the oldest, most popular, and most developed area.
- The execution virtualization techniques are divided into two major categories by considering the type of host they require.
- **Process-level techniques** are implemented on top of an existing operating system, which has full power of the hardware.
- **System-level techniques** are implemented directly on hardware and do not require a minimum of support from an existing operating system.

- Above two categories we can list out various techniques that provide the guest a different type of virtual computing environment: bare hardware, operating system resources, low-level programming language, and application libraries.

Execution virtualization

- Execution virtualization consists of all methods that aim to emulate an execution environment that is different from the one hosting the virtualization layer.
- All these techniques focus their interest on providing support for the execution of programs, whether these are the operating system, a binary specification of a program compiled against an abstract machine model, or an application.
- Hence execution virtualization can be executed directly on top of the hardware by the OS, an application, or libraries dynamically or statically connected to an application image.

1. Machine reference model

- Virtualizing an execution environment at different levels of the computing stack requires a reference model that defines the interfaces between the levels of abstractions, which hide implementation details.
- From this viewpoint, virtualization techniques actually replace one of the layers and intercept the calls that are directed toward it.
- Therefore, a clear uncoupling between layers clarify their implementation, which only requires the emulation of the interfaces and a proper interaction with the underlying layer.
- Modern computing systems can be represented in terms of the reference model expressed in figure 4.

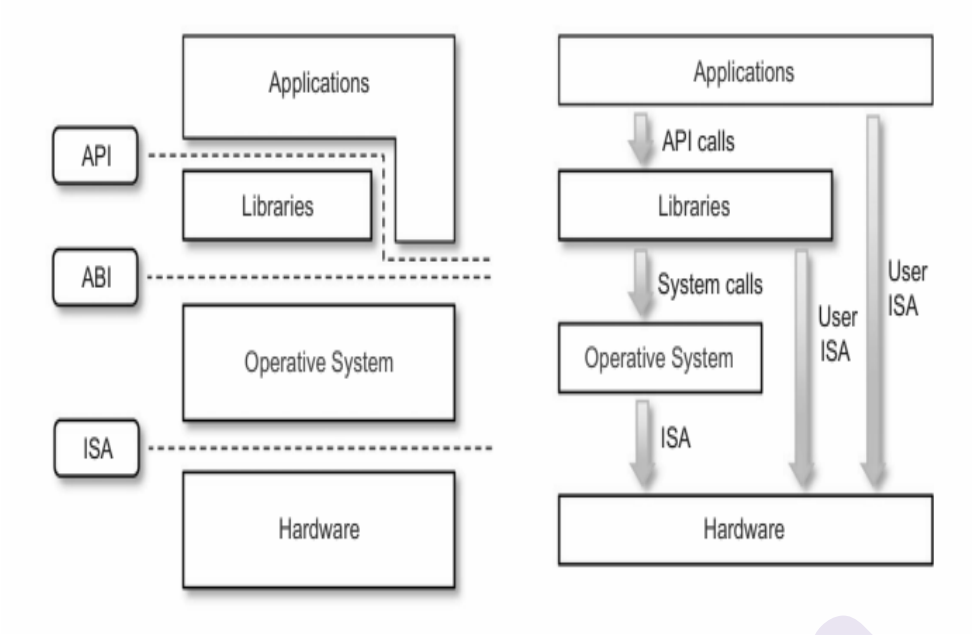


Figure 4 A machine reference model.

2. **Hardware-level virtualization**

- Virtualization technique that enables an abstract execution environment in terms of computer hardware on peak of which a guest operating system can be run.
- In this model, the guest is actuated by the OS, the host by the physical computer hardware, the virtual machine by its emulation, and the virtual machine manager by the hypervisor shown in figure 5.
- The hypervisor is generally a program or a combination of application, software and hardware that allows the abstraction of the underlying physical hardware.

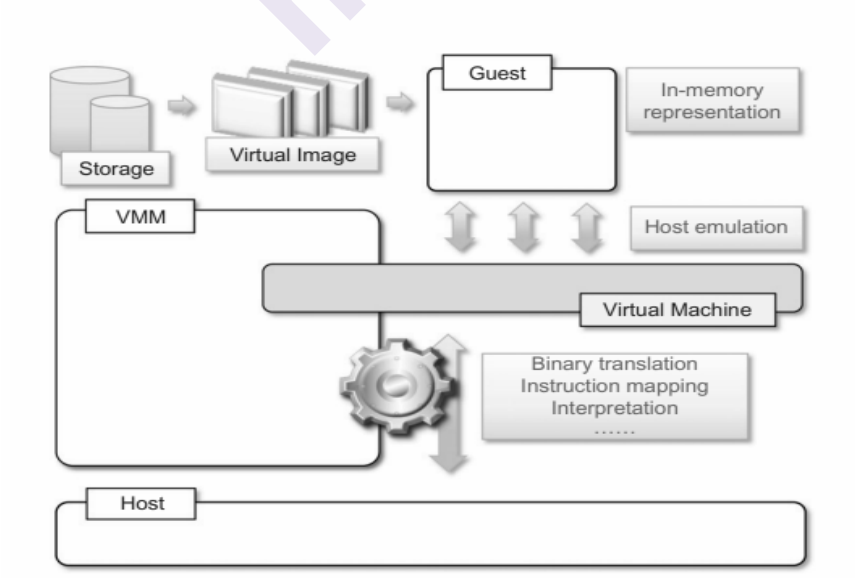


Figure 5. A hardware virtualization reference model.

- Hardware-level virtualization is also called system virtualization, since it supply ISA to virtual machines, which is the characterization of the hardware interface of a system.
- This is to differentiate it from process virtual machines, which expose ABI to virtual machines.

Hypervisors

- An elementary element of hardware virtualization is the hypervisor. It re-form a hardware environment in which guest operating systems are installed.
- Types of hypervisor:
  - Type I hypervisors execute on top of the hardware. Therefore, they grasp the place of the OS and communicate directly with the ISA interface exposed by the underlying hardware, and they imitate this interface in order to permit the management of guest operating systems. This hypervisor is also called a native virtual machine.
  - Type II hypervisors need the support of an operating system to enable virtualization services. Its mean programs controlled by the OS, which communicate with it through the ABI and emulate the ISA of virtual hardware for guest operating systems. Shown in figure 6.

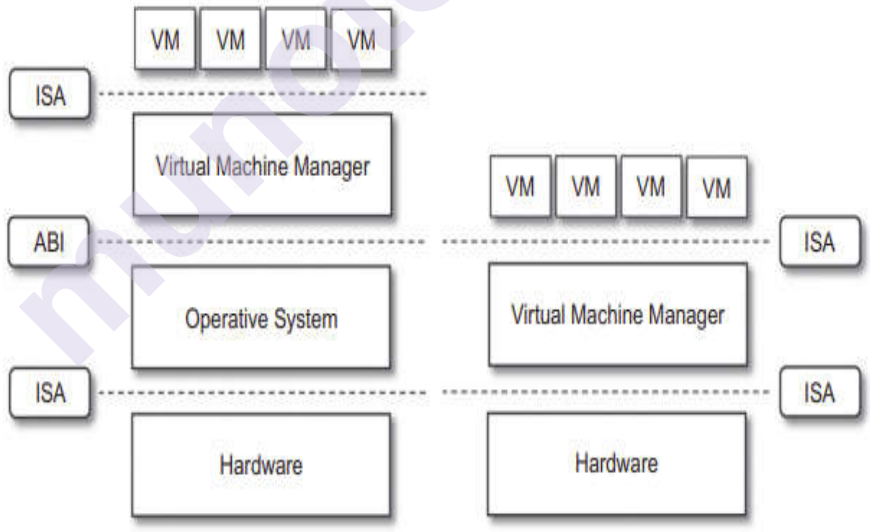


Figure 6 Hosted (left) and native (right) virtual machines.

Hardware virtualization techniques

- **Hardware-assisted virtualization.**
  - This term refers to a framework in which the hardware provides architectural aid for creating a virtual machine manager which is able to run a guest operating system in complete isolation.

- This technique was initially introduced in the IBM System 370. Examples is the extensions to the x86-64 bit architecture introduced with Intel VT and AMD V.

- **Full virtualization.**

- Full virtualization is the ability to run a program and application, such as an operating system, which resides directly on top of a virtual machine and without any alteration, as though it were run on the raw hardware.

- **Paravirtualization.**

- This is a not-transparent virtualization solution that enables execution of thin virtual machine managers.

- Paravirtualization methods expose a software interface to the virtual machine that is moderately changed or up-to- date from the host and, as a consequence, guests need to be modified. .

- **Partial virtualization.**

- Partial virtualization enables a partial emulation of the primary hardware, thus not permitting the complete execution of the guest operating system in complete isolation.

- Partial virtualization enables many applications to run translucency, but not all the features of the OS can be supported.

Operating system-level virtualization

- Operating system-level virtualization creates different and separated execution environments for applications that are controlled and handled concurrently.

- Operating systems supporting this type of virtualization are general-purpose, time shared operating systems with the capability to provide stronger namespace and resource isolation.

Programming language-level virtualization

- Programming language level virtualization is mainly used to attain ease of deployment of applications, managed implementation and portability across different platforms and operating systems.

3. Application-level virtualization

- Application-level virtualization allows applications to be executed in runtime environments that do not natively help all the features required by such applications.

4.4 SUMMARY

- The term virtualization is a large umbrella under which a variety of technologies and concepts are classified.
- The common root of all forms of virtualization is the ability to provide the illusion of a specific environment, whether a runtime environment, a storage facility, a network connection, or a remote desktop, by using some kind of emulation or abstraction layer.

4.5 REFERENCE FOR FURTHER READING

- Enterprise Cloud Computing Technology, Architecture, Applications, GautamShroff, Cambridge University Press, 2010
- Mastering In Cloud Computing, RajkumarBuyya, Christian Vecchiola And ThamariSelvi S, Tata Mcgraw-Hill Education, 2013
- Cloud Computing: A Practical Approach, Anthony T Velte, Tata Mcgraw Hill, 2009
- https://www.instructables.com/How-to-Create-a-Virtual-Machine/
- https://www.redhat.com/en/topics/virtualization/what-is-KVM#:~:text=Kernel%2Dbased%20Virtual%20Machine%20(KVM,KVM%20is%20part%20of%20Linux.
- https://u-next.com/blogs/cloud-computing/challenges-of-cloud-computing/

4.6 UNIT END EXERCISES

1. What is virtualization and what are its benefits?
2. What are the characteristics of virtualized environments?
3. Discuss classification or taxonomy of virtualization at different levels.

5

VIRTUALIZATION & CLOUD COMPUTING

Unit Structure

5.0 OBJECTIVE

- Understand the concept of virtualization and cloud computing.
- To study the pros and cons of virtualization.
- To study virtualization using KVM.
- To understand the open challenges of cloud computing.

5.1 INTRODUCTION

- Virtualization is the "creation of a virtual version of something, such as a server, a desktop, a storage device, an operating system or network resources".
- Another Way, 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 this by assigning a logical name to a physical storage and providing a pointer to that physical resource when demanded.

5.2 VIRTUALIZATION AND CLOUD COMPUTING

- Virtualization plays an important role in cloud computing since it allows for the appropriate degree of customization, security, isolation, and manageability that are fundamental for delivering IT services on demand.
- Virtualization technologies are primarily used to offer configurable computing environments and storage.
- Network virtualization is less popular and, in most cases, is a complementary feature, which is naturally needed in building virtual computing systems.
- Particularly important is the role of the virtual computing environment and execution virtualization techniques. Among these, hardware and programming language virtualization are the techniques adopted in cloud computing systems.
- Hardware virtualization is an enabling factor for solutions in the Infrastructure-as-a-Service (IaaS) market segment, while programming language virtualization is a technology leveraged in Platform-as-a-Service (PaaS) offerings.
- In both cases, the capability of offering a customizable and sandboxed environment constituted an attractive business opportunity for companies featuring a large computing infrastructure that was able to sustain and process huge workloads.
- Moreover, virtualization also allows isolation and a finer control, thus simplifying the leasing of services and their accountability on the vendor side.
- Besides being an enabler for computation on demand, virtualization also gives the opportunity to design more efficient computing systems by means of consolidation, which is performed transparently to cloud computing service users.
- Since virtualization allows us to create isolated and controllable environments, it is possible to serve these environments with the same resource without them interfering with each other.
- This opportunity is especially attractive when resources are not effectively used, because it decreases the number of active resources by aggregating virtual machines over a smaller number of resources that become fully utilized.
- This exercise is also known as server consolidation, while the movement of virtual machine instances is called virtual machine migration (Figure 1).

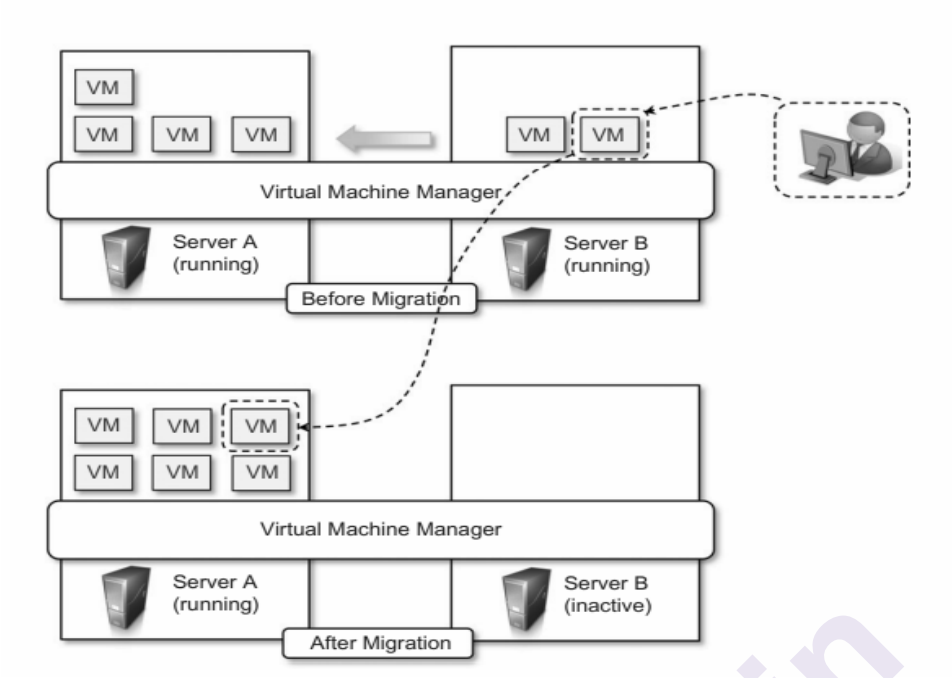


Figure 1 Live migration and server consolidation

5.3 PROS AND CONS OF VIRTUALIZATION

- Virtualization has become very popular and extensively used, especially in cloud computing.
- The primary reason for its wide success is the elimination of technology barriers that prevented virtualization from being an effective and viable solution in the past.
- The most relevant barrier has been performance.
- Nowadays virtualization is a compulsive opportunity to deliver on-demand IT infrastructure and services. Notwithstanding its renewed popularity, this technology has benefits and also drawbacks.

5.3.1 Advantages of virtualization

- Managed execution and isolation are maybe the most important advantages of virtualization.
- The creation of virtualized execution environments are the characteristics that allow building secure and controllable computing environments.
- A virtual execution environment can be designed as a sandbox, thus avert any harmful operation to cross the borders of the virtual host.
- This allows fine tuning of resources, which is very prime in a server consolidation scenario and is also a requirement for effective quality of service.

- Portability is one more advantage of virtualization, especially for execution virtualization techniques.
- Virtual machine instances are normally represented by one or more files that can be easily carried with respect to physical systems.
- Portability and self-containment simplify their administration. Java code is compiled once and runs everywhere. This needs the Java virtual machine to be installed on the host.
- Portability and self-containment helps to reduce the costs of maintenance.
- Multiple systems can securely coincide and share the resources of the underlying host, without interfering with each other.
- This is essential for server strengthening, which allows adjusting the number of active physical resources dynamically according to the current load of the system, thus creating the opportunity to save in terms of energy consumption and to be less impacting on the environment.

5.3.2 Disadvantages of virtualization

- Virtualization also has drawbacks. The most evident is represented by a performance decrease of guest systems as a result of the intermediation performed by the virtualization layer.
- The abstraction layer established by virtualization management software can lead to a very inefficient utilization of the host.

5.3.2.1 Performance degradation

- Performance is definitely one of the crucial concerns in using virtualization technology. Since virtualization insinuates an abstraction layer between the guest and the host, the guest can experience increased latencies.
- The causes of performance degradation can be traced back to the overhead introduced by the following pursuit.
  - Maintaining the status of virtual processors
  - Support of privileged instructions
  - Support of paging within VM
  - Console functions

5.3.2.2 Inefficiency and degraded user experience

- Virtualization can sometimes lead to an unsuitable use of the host. Specially, some of the certain features of the host cannot be exposed by the abstraction layer and then become inaccessible.

- In hardware virtualization, the virtual machine can from time to time simply provide a default graphic card that maps only a subset of the features available in the host.
- In the course of programming level virtual machines, some of the features of the underlying operating systems may become inaccessible unless specific libraries are used.
- Example is the first version of Java the support for graphic programming was very finite and the look and feel of applications was very needy compared to native applications.
- These problems have been resolved by providing a new framework called java swing for designing the user interface, and further development has been done by integrating support for the OpenGL libraries in the software development kit.

5.3.2.3 Security holes and new threats

- Virtualization opens the door to a new and unpredicted form of phishing.
- The potential of emulating a host in a completely transparent manner led the way to malicious programs that are designed to extract sensitive information from the guest.
- In hardware virtualization, malicious programs can preload one self before the OS and act as a thin virtual machine manager toward it.
- The operating system is then managed and can be altered to extract importatnt information of interest to third parties.
- Examples of malware are BluePill and SubVirt.

5.4 VIRTUALIZATION USING KVM

- Kernel Virtual Machine (KVM) is an open source virtualization technology developed into Linux.
- KVM become Linux into a hypervisor that enables a host machine to run multiple isolated virtual environments called guests / virtual machines (VMs).
- KVM belongs to Linux.
- KVM was first declared in 2006 and merged into the mainline Linux kernel version a year later.
- KVM is part of existing Linux code, it abruptly benefits from each new Linux feature, fix, and advancement except additional engineering.

Working of KVM

- KVM converts Linux into a type-1 hypervisor.
- All hypervisors require some OS-level part like a memory manager, process scheduler, input or output (I/O) stack, device drivers, security manager, a network stack, and more to run VMs.
- KVM consists of all these components because it's part of the Linux kernel.
- Every VM is implemented as a regular Linux process, arranged by the standard Linux scheduler, with fixed virtual hardware like a network card, graphics adapter, CPU(s), memory, and disks.

KVM features

KVM is part of Linux. Linux is part of KVM. Everything Linux has, KVM has too. But there are certain features that form KVM, an enterprise's preferred hypervisor.

- **Security**

KVM employs a combination of security enhanced Linux and secure virtualization (sVirt) for enhanced VM security and isolation.

- **Storage**

KVM is capable of using any storage supported by Linux, including some local disks and network-attached storage (NAS).

- **Hardware support**

KVM can use a broad variation of certified Linux supported hardware platforms.

- **Memory management**

Kernal VM inherits the memory management features of Linux, including non-uniform memory access and kernel same-page merging.

- **Live migration**

Kernel VM helps live migration, which is the ability to proceed a running VM between physical hosts with no service onstrution.

- **Performance and scalability**

Kernel VM inherits the performance of Linux, scaling to match demand load if the number of guest machines and requests increases.

- **Scheduling and resource control**

In the KVM model, a VM is a Linux process, scheduled and managed by the kernel.

- **Lower latency and higher prioritization**

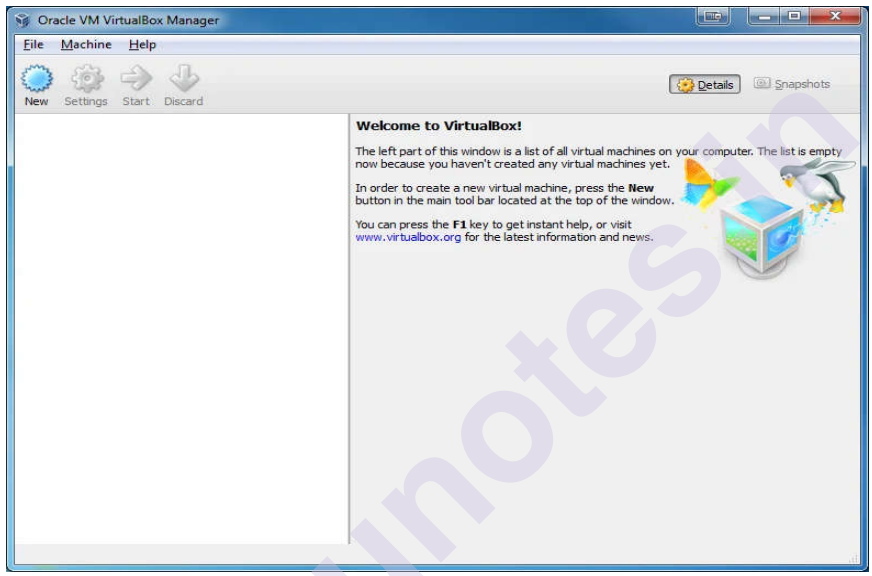
The Linux kernel features real-time extensions that allow VM-based apps to run at lower latency with better prioritization (compared to bare metal).

- **Managing KVM**

It's possible to manually manage a handful of VM fires on a single workstation without a management tool.

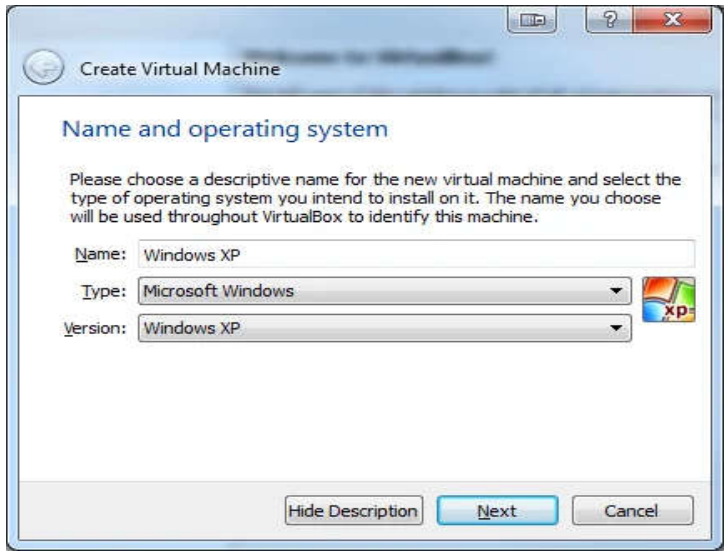
5.5 CREATING VIRTUAL MACHINES

Step 1: Download and Install VirtualBox



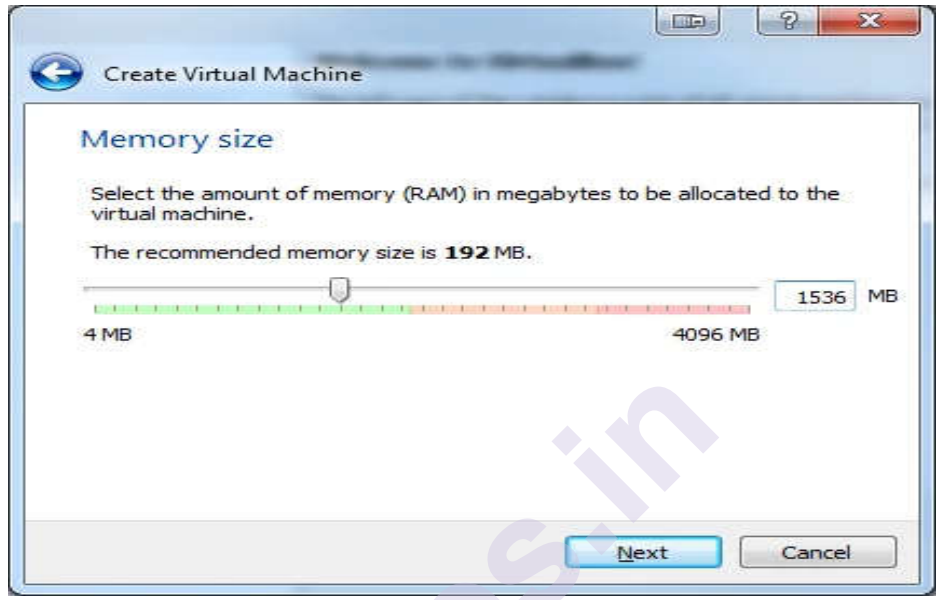
Keep all of the default settings. You will be prompted to install several Oracle components. Install all of them.

Step 2: Create a Virtual Machine



Start VirtualBox and Click on 'New' in the menu. Enter the Name of your VM. This is how you will identify it in VirtualBox so name it something meaningful to you. Select Type and Version. This depends on what OS you are installing.

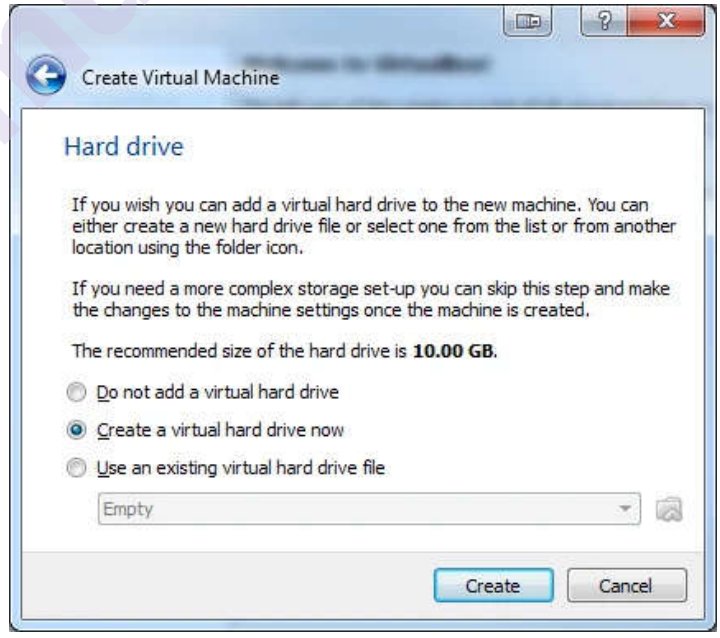
Step 3: Allocate Memory



This depends on how much memory you have on your host computer. Never allocate more than half of your available RAM. If you are creating a Windows VM I recommend at least (1-2 GB)

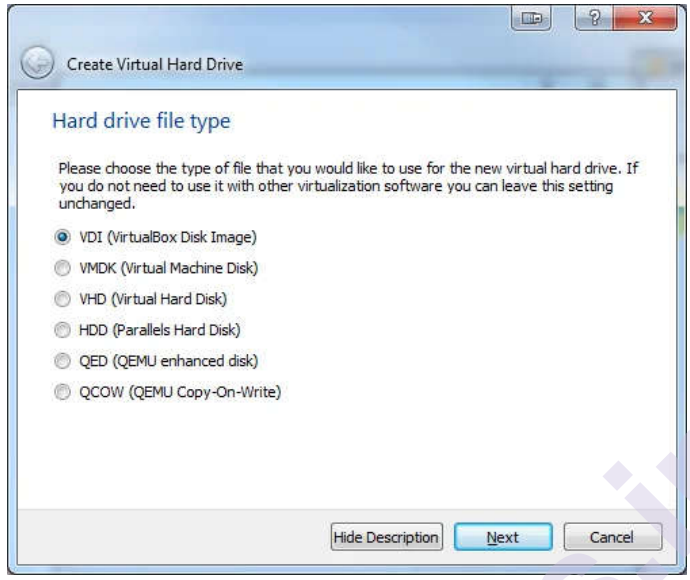
If you are creating a Linux VM I recommend at least (512 MB)

Step 4: Setup the Hard Drive



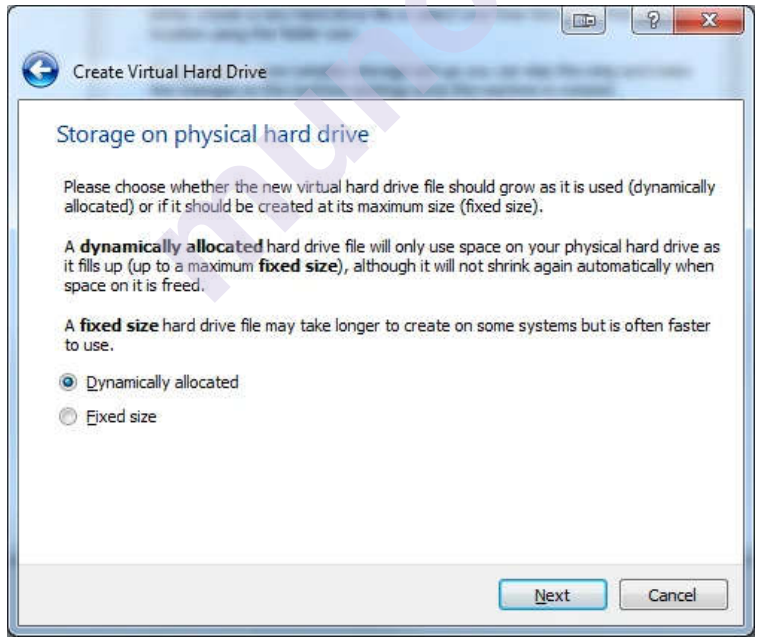
If you already have an existing VM that you want to add select "Use an existing Virtual hard drive file." Otherwise select "Create a virtual hard drive now."

Step 5: Select Hard Drive File Type



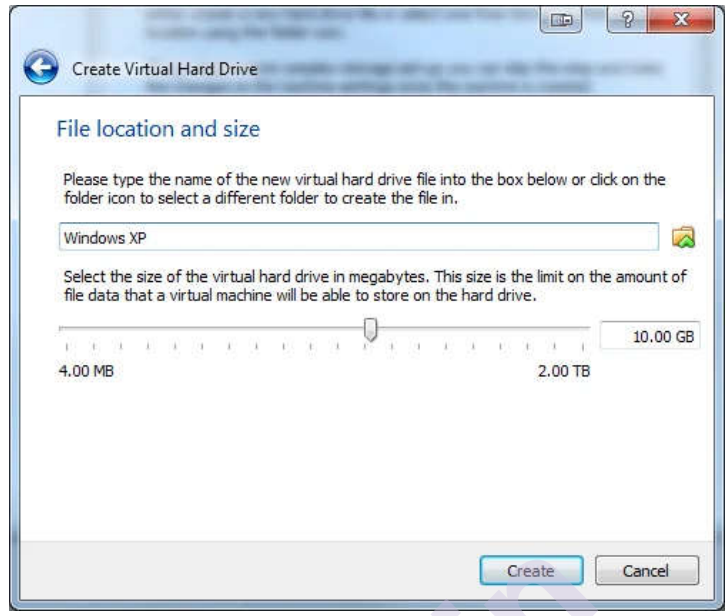
Select 'VDI.' This is usually the best option. The VM will be stored in a single file on your computer with the .vdi extension.

Step 6: Select Storage on Physical Hard Drive



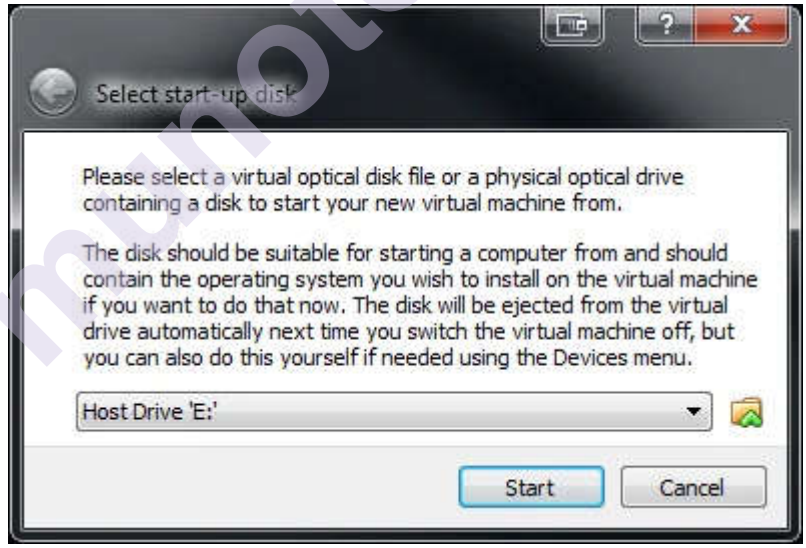
I recommend you choose "Dynamically allocated." This will save space on your computer.

Step 7: Setup File Location and Size



By default, Virtualbox selects the minimum size you should choose. Depending on what you want to do with the VM you may want to select a bigger size.

Step 8: Install the Operating System



Double click on your newly created VM (It will be on the left hand side and will have the name you gave it in Step 2). Browse to your installation media or .iso file. Finish installation.

5.6 OVIRT - MANAGEMENT TOOL FOR VIRTUALIZATION ENVIRONMENT

- OVirt is an open source data center virtualization platform developed and encouraged by Red Hat. OVirt, which provides large-scale, centralized management for server and desktop virtualization, was planned as an open source alternative to VMware vCenter.
- OVirt gives kernel based virtual machine management for multi-node virtualization. Kernel-based Virtual Machines (KVMs) are a virtualization infrastructure that changes the Linux kernel into a hypervisor.
- **Features of oVirt**

○ OVirt enables centralized management of VMs, networking configurations, hosts, and compute and storage resources from the web based front end.

○ OVirt also provides features for disaster recovery (DR) and hyper converged infrastructure deployments.

○ Features for the management of compute resources include:

■ CPU pinning,

■ same-page merging and

■ memory over commitment.

○ VM management features include

■ live migrations,

■ live snapshots,

■ the creation of VM templates and VMs,

■ automated configuration

○ DR features consist of inputting storage domains to different types.

○ OVirt utilizes both self-hosted and Gluster Storage domains for centralized management.

Components of oVirt

1. oVirt engine

a. The oVirt engine acts as the control center for oVirt environments.

b. The engine enables admins to define hosts and networks, as well as to add storage, create VMs and manage user permissions.

c. Included in the oVirt engine is a graphical user interface (GUI), which manages oVirt infrastructure resources.

d. The oVirt engine can be installed on a stand-alone server or in a node cluster in a VM.

2. oVirt node

a. The oVirt node is a server that runs on CentOS, Fedora or Red Hat Enterprise Linux with a virtual desktop and server manager (VDSM) daemon and KVM hypervisor.

b. The VDSM controls the resources available to the node, including compute, networking and storage resources.

5.7 OPEN CHALLENGES OF CLOUD COMPUTING

1. Security

- The main concern in investing in cloud services is security issues in cloud computing.

  It is because your data gets stored and processed by a third-party vendor and we cannot see it.

- We listen about broken authentication, compromised credentials, account hacking, data breaches, etc. in a particular organization. It makes you a little more doubtful.

2. Password Security

- As large numbers of people access cloud accounts, it sometimes becomes vulnerable. Anybody who knows the password or hacks into the cloud will be able to access confidential information.
- Nowadays organizations should use multiple level authentications and make sure that the passwords remain secured. Also, the passwords should be updated regularly, especially when a particular employee leaves the job or leaves the organization.

3. Cost Management

- Cloud computing allows access to application software over a fast internet connection and lets save on investing in costly computer hardware, software, management, and maintenance.

4. Lack of expertise

- With the increasing workload on cloud technologies and regularly improving cloud tools, management has become very difficult.
- There has been a consistent order for a trained workforce who can coordinate with cloud computing tools and services.
- Firms required training their IT staff to minimize this challenge.

5. Internet Connectivity

- Cloud services are mainly dependent on a high-speed internet connection.
- Hence businesses that are small and face connectivity problems should perfectly first invest in a good internet connection so that no downtime happens.
- It is because internet downtime might incur infinite business losses.

6. Control or Governance

- One more ethical issue in cloud computing is maintaining proper control over asset management and maintenance.
- There should be an individual team to make sure that the assets used to implement cloud services are used according to concur policies and dedicated procedures.

7. Compliance

- Another major risk of cloud computing is maintaining compliance.
- By compliance using mean, a set of rules about what data is permitted to be moved and what should be kept in house to maintain compliance.
- The organizations hence follow and respect the compliance rules set by various government bodies.

8. Multiple Cloud Management

- Companies have begun to invest in multiple public clouds, multiple private clouds or a combination of both is called the hybrid cloud.
- This has expanded rapidly in recent times.
- So it has become important to list the various challenges faced by such types of organizations and find solutions to grow with the trend.

9. Creating a private cloud

- Implementing an internal cloud is beneficial. This is because all the data remains secure in house.
- But the challenge here is that the IT team should build and fix everything by themselves. Also, the team is required to ensure the smooth working of the cloud.
- They are required to automate maximum manual tasks to be dynamic. The execution of tasks should be in the proper order.

10. Performance

- When business applications migrate to a cloud or a third-party vendor, the business performance starts to depend on the service provider as well.
- Another major issue in cloud computing is investing in the right cloud service provider.

11. Migration

- Migration is nothing but updating an application and a new application or an existing application to a cloud. In the case of a new application, the process is good and straightforward.

5.8 SUMMARY

- Virtualization has become very popular and extensively used, especially in cloud computing.
- All these concepts play a fundamental role in building cloud computing infrastructure and services in which hardware; IT infrastructure, applications, and services are delivered on demand through the Internet or more generally via a network connection.
- OVirt is an open source data center virtualization platform developed and encouraged by Red Hat. OVirt, which provides large-scale, centralized management for server and desktop virtualization, was planned as an open source alternative to VMware vCenter.

5.9 REFERENCE FOR FURTHER READING

- Enterprise Cloud Computing Technology, Architecture, Applications, GautamShroff, Cambridge University Press, 2010
- Mastering In Cloud Computing, RajkumarBuyya, Christian Vecchiola And ThamariSelvi S, Tata Mcgraw-Hill Education, 2013
- Cloud Computing: A Practical Approach, Anthony T Velte, Tata Mcgraw Hill, 2009
- https://www.instructables.com/How-to-Create-a-Virtual-Machine/
- https://www.redhat.com/en/topics/virtualization/what-is-KVM#:~:text=Kernel%2Dbased%20Virtual%20Machine%20(KVM,KVM%20is%20part%20of%20Linux.
- https://u-next.com/blogs/cloud-computing/challenges-of-cloud-computing/

5.10 UNIT END EXERCISES

1. What are the pros and cons of cloud computing?
2. Discuss Virtualization using KVM?
3. Explain how to create a virtual machine?
4. What are the open challenges of cloud computing?

❖❖❖❖

6

OPEN STACK

Unit Structure

6.1 OBJECTIVES

At the end of this unit, the student will be able to

- Understand the open stack and its components.
- Describe various parameters of openstack

6.2 INTRODUCTION TO OPENSTACK

1. OpenStack is an open-source cloud computing platform that provides a set of software tools for building and managing public, private, and hybrid clouds. It was initially released in 2010 as a joint project of NASA and Rackspace, and it has since become one of the most popular cloud computing platforms in the world.

2. OpenStack consists of a set of interrelated components, each of which provides a specific function in the cloud computing environment. These components include:

- Compute (Nova): This component provides virtualization services and manages the creation and scheduling of virtual machines.
- Networking (Neutron): This component provides network connectivity services and manages the creation and configuration of virtual networks.
- Storage (Cinder and Swift): These components provide storage services and manage the creation and management of block and object storage.
- Identity (Keystone): This component provides authentication and authorization services and manages the creation and management of user accounts.
- Dashboard (Horizon): This component provides a web-based user interface for managing the cloud computing environment.
- Orchestration (Heat): This component provides automated deployment and management of cloud applications.
- Telemetry (Ceilometer): This component provides monitoring and metering services for the cloud computing environment.

3. OpenStack is designed to be highly scalable and flexible, allowing it to be used in a wide range of cloud computing environments. It is also modular, which means that it can be easily customized and extended to meet the specific needs of individual organizations.

4. OpenStack has a large and active community of developers, users, and vendors, who contribute to the development and support of the platform. This community has created a wide range of tools and plugins that can be used with OpenStack to extend its functionality and make it easier to use.

6.3 OPENSTACK TEST-DRIVE

1. To test-drive OpenStack, you can use one of the available OpenStack distributions or deploy your own OpenStack environment. Here are the general steps to follow:

2. Choose an OpenStack distribution: Several vendors offer pre-packaged OpenStack distributions, such as Red Hat OpenStack, Canonical OpenStack, and MirantisOpenStack. You can also choose to deploy OpenStack directly from the source code.

3. Set up a testing environment: You will need a testing environment with adequate resources to deploy OpenStack, such as a dedicated server or a virtual machine. You can use a cloud provider like AWS or Google Cloud to set up the testing environment.

4. Install OpenStack: Follow the installation instructions provided by the OpenStack distribution or the official OpenStack documentation to install the necessary components.

5. Configure OpenStack: Once OpenStack is installed, you will need to configure it to suit your needs. This includes setting up users, roles, and permissions; configuring network settings; and defining storage policies.

6. Use OpenStack: Once OpenStack is up and running, you can use it to create virtual machines, configure networks, and store data. You can use the OpenStack dashboard or the command-line interface to manage the cloud computing environment.

7. Test OpenStack: To test OpenStack, you can create and deploy virtual machines, test network connectivity, and simulate workload scenarios to test the performance and scalability of the environment.

8. Keep in mind that deploying and configuring OpenStack can be a complex process, and it requires some level of expertise in cloud computing and networking. You may want to seek assistance from the OpenStack community or a professional services provider to ensure a smooth and successful deployment.

9. Pre-built OpenStack environment: Several companies offer pre-built OpenStack environments that you can use to test-drive the platform. These environments are typically available as virtual appliances or cloud images that you can download and run on your own computer or cloud environment. Some examples of pre-built OpenStack environments include:

- MirantisOpenStack Express
- Canonical's Ubuntu OpenStack
- Red Hat OpenStack Platform

10. Deploy your own OpenStack environment: If you want to deploy your own OpenStack environment, you can use tools such as DevStack or Packstack. DevStack is a script that automates the installation of OpenStack on a single machine, while Packstack is a similar tool that can be used to deploy OpenStack on multiple machines. To deploy OpenStack on your own, you will need to have a server or virtual machine that meets the hardware and software requirements for OpenStack.

11. Once you have a test environment set up, you can use the OpenStack web interface (Horizon) or command-line interface (CLI) to create virtual machines, networks, and storage resources. You can also explore the different OpenStack components and their functionality, such as the compute (Nova) and networking (Neutron) components.

6.4 BASIC OPENSTACK OPERATIONS

Some basic OpenStack operations include:

1. Creating instances: OpenStack instances are virtual machines that can be used to run applications and services. To create an instance, you can use the OpenStack web interface (Horizon) or command-line interface (CLI). You will need to specify the flavor (CPU, memory, and disk configuration), image (operating system and software), network, and security settings for the instance.

2. Managing networks: OpenStack provides a networking service (Neutron) that allows you to create and manage virtual networks. To create a network, you can use the OpenStack web interface or CLI and specify the network name, subnet, and IP address range. You can also attach instances to networks and configure security settings for the network.

3. Managing storage: OpenStack provides two storage services, Cinder (block storage) and Swift (object storage). To create a block storage volume, you can use the OpenStack web interface or CLI and specify the volume size, type, and other settings. To create an object storage container, you can use the Swift CLI and specify the container name and access settings.

4. Managing users and projects: OpenStack provides an identity service (Keystone) that allows you to create and manage users, projects, and roles. Users are granted roles that define their level of access to OpenStack resources. To create a user, you can use the OpenStack web interface or CLI and specify the user name, password, and other settings. To create a project, you can also use the OpenStack web interface or CLI and specify the project name and description.

5. Monitoring and logging: OpenStack provides several tools for monitoring and logging OpenStack resources. The telemetry service (Ceilometer) provides monitoring and metering of OpenStack resources, while the logging service (Logstash) provides centralized logging for OpenStack components. You can use these tools to monitor performance, detect errors, and troubleshoot issues in your OpenStack environment.

6. These are just a few examples of basic OpenStack operations. OpenStack provides a wide range of functionality, and the operations required will depend on your specific use case and requirements.

7. Here are some basic OpenStack operations that you can perform using the Horizon web interface or the OpenStack command-line interface (CLI):

- Launch an instance: You can create a new virtual machine instance by selecting the Compute tab in the Horizon dashboard and clicking on "Launch Instance." You will be prompted to select a flavor (virtual machine size), an image (operating system), and other configuration options.

- To launch an instance using the CLI, you can use the openstack server create command and specify the necessary parameters.

- Create a network: You can create a new network for your instances by selecting the Network tab in the Horizon dashboard and clicking on "Create Network." You will be prompted to specify the network type, subnet details, and other configuration options.

- To create a network using the CLI, you can use the openstack network create command and specify the necessary parameters.

- Attach a volume: You can attach a volume to an instance to provide additional storage by selecting the Compute tab in the Horizon dashboard, clicking on the instance, and selecting "Attach Volume." You will be prompted to select the volume and specify the device name.

- To attach a volume using the CLI, you can use the openstack server add volume command and specify the necessary parameters.

- Manage security groups: You can manage security groups to control incoming and outgoing traffic to your instances by selecting the Compute tab in the Horizon dashboard and clicking on "Access & Security." You can create new security groups, add rules, and associate them with instances.

- To manage security groups using the CLI, you can use the openstack security group create and openstack security group rule create commands.

- Resize an instance: You can resize an instance to a different flavor (virtual machine size) by selecting the Compute tab in the Horizon dashboard, clicking on the instance, and selecting "Resize Instance." You will be prompted to select the new flavor.

- To resize an instance using the CLI, you can use the openstack server resize command and specify the necessary parameters.

6.5 OPENSTACK CLI AND APIS

1. OpenStack provides a command-line interface (CLI) and APIs that allow you to manage and automate your cloud resources.

2. The OpenStack CLI is a tool that allows you to interact with OpenStack services from the command line. It is a powerful tool that allows you to automate tasks and perform complex operations.

3. The CLI uses the OpenStack API to communicate with the OpenStack services. The CLI is available for all OpenStack services, including Compute, Networking, Identity, Image, and Block Storage.

4. To use the OpenStack CLI, you need to install the OpenStack client on your local machine. The client is available for Linux, macOS, and Windows. Once you have installed the client, you can use the openstack command to interact with OpenStack services.

5. The OpenStack APIs are a set of RESTful APIs that allow you to programmatically interact with OpenStack services. The APIs provide a standardized way of accessing OpenStack services and can be used by developers to create custom applications and tools that interact with OpenStack services.

6. The OpenStack APIs are available for all OpenStack services, including Compute, Networking, Identity, Image, and Block Storage. The APIs are based on industry standards such as JSON, XML, and HTTP. You can use any programming language that supports HTTP to interact with the OpenStack APIs.

7. OpenStack also provides software development kits (SDKs) for popular programming languages such as Python, Java, and Ruby. The SDKs provide a higher-level interface to the OpenStack APIs and make it easier to build applications that interact with OpenStack services.

8. Overall, the OpenStack CLI and APIs are powerful tools that allow you to manage and automate your cloud resources. Whether you prefer to use the CLI or APIs, OpenStack provides a flexible and extensible platform for building and managing cloud infrastructure.

6.6 TENANT MODEL OPERATIONS

1. The tenant model is a core concept in OpenStack that enables multi-tenancy within a cloud environment. Tenants are logical groups of resources that are isolated from each other, allowing multiple organizations to share the same infrastructure without interfering with each other.

2. Here are some basic OpenStack tenant model operations:

- Creating a tenant: You can create a new tenant using the OpenStack CLI or APIs. When you create a tenant, you specify a name and an optional description.

- Creating users: Once you have created a tenant, you can create one or more users within that tenant. Users are granted access to the resources associated with their tenant.

- Assigning roles: You can assign roles to users within a tenant. Roles are used to define what actions a user can perform on a specific resource. For example, you might assign a user the role of "admin" for a particular project, giving them full access to all resources within that project.

- Creating projects: Projects are containers for resources within a tenant. You can create one or more projects within a tenant, and assign users and roles to those projects.

- Managing quotas: OpenStack allows you to set quotas for resources within a tenant, such as the number of instances or the amount of storage that can be used. You can set and manage quotas using the OpenStack CLI or APIs.

- Monitoring usage: OpenStack provides tools for monitoring resource usage within a tenant, allowing you to track usage and ensure that tenants are not exceeding their quotas.

- Managing user access to tenants: You can control which users have access to a tenant's resources by assigning roles to those users. You can assign multiple roles to a user, and you can assign roles to users at the tenant level or the project level.

- Managing resources: Once you have created a tenant, you can create and manage resources within that tenant. You can create instances, volumes, networks, and images, and you can assign those resources to the tenant.

3. Overall, the tenant model is a powerful feature of OpenStack that allows you to create a secure and scalable multi-tenant cloud environment. By using tenants, users, roles, and projects, you can create a flexible and customizable cloud infrastructure that meets the needs of your organization.

4. In OpenStack, a tenant is a logical grouping of resources that allows you to isolate and control access to those resources. A tenant can be a department, a project, or any other organizational unit. Each tenant in

OpenStack has its own set of resources, including instances, volumes, networks, and images.

5. Overall, the tenant model in OpenStack provides a flexible and powerful way to manage and control access to cloud resources. By using tenants, you can isolate resources and control access to those resources, which makes it easier to manage and secure your cloud infrastructure.

6.7 QUOTAS, PRIVATE CLOUD BUILDING BLOCKS

1. Quotas in OpenStack are limits that are imposed on the amount of resources that a tenant can use. Quotas are used to prevent a tenant from using too many resources and degrading the performance of the cloud infrastructure. Quotas can be set for individual resources, such as instances, volumes, and networks, as well as for aggregate resources, such as CPU cores and RAM.

2. Here are some examples of quotas that can be set in OpenStack:

- Instance quotas: This sets a limit on the number of instances that a tenant can create.

- Volume quotas: This sets a limit on the amount of storage that a tenant can allocate to volumes.

- Network quotas: This sets a limit on the number of networks and subnets that a tenant can create.

- Floating IP quotas: This sets a limit on the number of floating IPs that a tenant can allocate.

3. Private cloud building blocks are the components that are used to build a private cloud infrastructure. These components include the physical hardware, such as servers and storage devices, as well as the software, such as OpenStack, that is used to manage the cloud infrastructure.

4. Here are some examples of private cloud building blocks

- Compute nodes: These are the physical servers that are used to host virtual machines.

- Storage nodes: These are the physical servers that are used to provide storage for the cloud infrastructure.

- Networking hardware: This includes switches and routers that are used to connect the cloud infrastructure to the external network.

- Virtualization software: This is the software that is used to create and manage virtual machines.

- Cloud management software: This includes OpenStack and other software tools that are used to manage the cloud infrastructure.

5. Overall, the private cloud building blocks are the foundation of a private cloud infrastructure, and they must be carefully selected and configured to ensure that the cloud infrastructure is reliable, scalable, and secure. Quotas play an important role in ensuring that tenants use resources responsibly and that the cloud infrastructure performs well.

6. Quotas in OpenStack are used to limit the amount of resources that a tenant or user can consume. You can set quotas for different resources such as CPU, memory, storage, and network usage. Quotas can be set at the tenant level or the user level, and you can customize the quotas based on your specific needs.

7. Setting quotas helps to prevent overutilization of resources and ensure fair resource allocation among different tenants and users. Quotas can be managed using the OpenStack CLI or APIs.

8. A private cloud is a cloud infrastructure that is dedicated to a single organization. Private clouds offer several benefits, including enhanced security, greater control over resources, and increased flexibility.

9. Here are some common building blocks of a private cloud infrastructure:

- Hypervisor: A hypervisor is the software that enables virtualization of physical servers. It allows multiple virtual machines to run on a single physical server.

- Storage: Storage is a critical component of a private cloud infrastructure. You can use different types of storage, such as block storage, object storage, and file storage, depending on your specific needs.

- Networking: Networking is essential for connecting different components of a private cloud infrastructure. You can use software-defined networking (SDN) to create and manage virtual networks, subnets, and routers.

- Orchestration: Orchestration is the automation of the deployment and management of cloud resources. You can use tools such as OpenStack Heat or AWS CloudFormation to automate the creation and management of resources in your private cloud.

- Identity and Access Management (IAM): IAM is used to manage user access to cloud resources. You can use IAM tools such as OpenStack Keystone to authenticate and authorize users and assign roles and permissions.
- Monitoring and Management: Monitoring and management tools are used to ensure that your private cloud infrastructure is running smoothly. You can use tools such as Nagios or Zabbix to monitor system performance and detect issues before they become problems.

10. Overall, these building blocks are the foundation of a private cloud infrastructure, and they enable you to create a flexible, scalable, and secure cloud environment that meets your organization's needs.

6.8 CONTROLLER DEPLOYMENT

1. In OpenStack, the controller node is a critical component of the infrastructure. It is responsible for managing and coordinating all the other nodes in the OpenStack deployment, such as compute, storage, and network nodes.

2. The controller node typically runs the following OpenStack services:

- Identity (Keystone): Provides authentication and authorization services for OpenStack services and tenants.
- Image (Glance): Stores and retrieves virtual machine images.
- Dashboard (Horizon): Provides a web-based user interface for OpenStack services.
- Orchestration (Heat): Provides an API for defining and automating the deployment of infrastructure resources.
- Telemetry (Ceilometer): Collects metrics and data on OpenStack services.
- Networking (Neutron): Provides networking services for virtual machines and other OpenStack components.

3. Here are some general steps for deploying the controller node:

- Install the base operating system: The first step is to install the operating system on the server that will become the controller node. Many OpenStack distributions provide pre-configured images that you can use.

- Install OpenStack packages: Next, you need to install the OpenStack packages on the controller node. This can be done using package managers like yum or apt-get.

- Configure the services: Once the OpenStack packages are installed, you need to configure the services on the controller node. This includes setting up the database for each service, configuring the messaging system, and configuring the service endpoints.

- Verify the installation: After the services are configured, you can verify that they are working properly by running various tests and checks. For example, you can use the OpenStack CLI to check that you can authenticate and access OpenStack services.

- Configure additional services: Finally, you may want to configure additional services on the controller node, such as load balancing or firewall services.

4. It's important to note that the controller node deployment process can vary depending on the specific OpenStack distribution and version you are using, as well as the requirements of your environment. It's always a good idea to consult the documentation and follow best practices for your particular deployment.

5. In OpenStack, the controller node is the central management component of the cloud infrastructure. It provides the API services that enable users to interact with the cloud infrastructure, as well as the scheduling and orchestration of resources.

6. Here are the steps for deploying an OpenStack controller node:

- Prepare the controller node: The first step is to prepare the controller node by installing the operating system and configuring the network interfaces. You should also configure the hostname, domain name, and time zone.

- Install the OpenStack packages: The next step is to install the OpenStack packages, including the identity service (Keystone), the image service (Glance), the compute service (Nova), the network service (Neutron), and the dashboard (Horizon). You can use the package manager of your operating system (e.g., apt-get or yum) to install the packages.

- Configure the services: After installing the packages, you need to configure the services. For example, you need to configure Keystone to authenticate users and Neutron to provide network connectivity. You can use the OpenStack CLI or API to configure the services.

- Configure the database: OpenStack uses a database to store configuration information and metadata about resources. You need to configure the database service (e.g., MySQL or MariaDB) and create the necessary databases and users.

- Configure the message queue: OpenStack uses a message queue (e.g., RabbitMQ or Qpid) to communicate between the different components of the infrastructure. You need to configure the message queue service and create the necessary users and permissions.

- Start the services: After configuring the services, you can start them on the controller node using the service manager of your operating system (e.g., systemctl or service).

- Verify the installation: Finally, you should verify that the OpenStack services are running correctly by using the OpenStack CLI or API to create and manage resources.

7. Overall, deploying an OpenStack controller node requires careful planning and configuration, but it is a critical step in creating a functional and scalable cloud infrastructure.

6.9 NETWORKING DEPLOYMENT

1. In OpenStack, the networking component is responsible for providing network connectivity to instances (virtual machines). The networking deployment process involves configuring the networking services and components, such as the OpenStack Networking (Neutron) service, the Open vSwitch (OVS) agent, and network nodes.

2. Here are the steps for deploying networking in OpenStack:

- Install and configure the Neutron service: The first step is to install the Neutron service on the controller node and configure it to provide network connectivity. This involves configuring the Neutron server, the Neutron API, and the Neutron plugin (e.g., ML2). You also need to configure the Neutron database and the message queue service.

- Configure the OVS agent: The next step is to configure the Open vSwitch (OVS) agent, which provides virtual network connectivity to instances. This involves configuring the OVS service, creating the necessary bridges and ports, and configuring the OVS firewall.

- Configure network nodes: If you have multiple network nodes, you need to configure them to provide network connectivity to instances. This involves configuring the network node, the OVS agent, and the Neutron agent.

- Create networks, subnets, and routers: Once the Neutron service and OVS agent are configured, you can create networks, subnets, and routers. A network is a logical abstraction that provides connectivity between instances, while a subnet is a range of IP addresses that can be used by instances in a network. A router is a virtual device that connects two or more networks.

- Create security groups and rules: To control network access and security, you can create security groups and rules. A security group is a set of firewall rules that define the allowed inbound and outbound traffic for instances.

- Launch instances: Finally, you can launch instances and attach them to the networks and security groups you created. The instances should be able to communicate with each other and with the external network through the router.

3. Overall, deploying networking in OpenStack requires careful planning and configuration, but it is a critical step in creating a functional and scalable cloud infrastructure.

6.10 BLOCK STORAGE DEPLOYMENT

1. In OpenStack, block storage provides persistent storage for instances (virtual machines) that requires data to persist beyond the lifetime of the instance.

2. The block storage deployment process involves configuring and deploying the Cinder service, which manages the creation, deletion, and management of block storage volumes.

3. Here are the steps for deploying block storage in OpenStack:

- Install and configure the Cinder service: The first step is to install the Cinder service on the controller node and configure it to provide block storage. This involves configuring the Cinder server, the Cinder API, the Cinder scheduler, and the Cinder volume service.

- Configure the storage backends: Once the Cinder service is installed and configured, you need to configure the storage backends that will provide the physical storage for block volumes. OpenStack supports a variety of storage backends, including local disks, iSCSI, NFS, and Ceph. You need to configure the appropriate drivers for your storage backend.

- Create storage pools: A storage pool is a group of storage devices that are managed together. You need to create storage pools for each storage backend you configured.

- Create volume types: A volume type is a way to define the characteristics of a block volume, such as the size, performance, and availability. You need to create volume types that reflect the different needs of your applications.

- Create block volumes: Once the storage backends and volume types are configured, you can create block volumes. A block volume is a persistent block storage device that can be attached to an instance.

- Attach block volumes to instances: Finally, you can attach block volumes to instances. This allows the instance to access the persistent storage provided by the block volume.

- Configure the storage backend: The next step is to configure the storage backend, which is the physical storage device that provides the block storage. There are different types of storage backends that can be used with Cinder, such as LVM, Ceph, and NFS. You need to configure the backend based on the type of storage you are using.

- Create storage pools and volumes: Once the Cinder service and storage backend are configured, you can create storage pools and volumes. A storage pool is a group of storage devices that are used to create volumes, while a volume is a block-level storage device that can be attached to an instance.

- Create volume types and encryption: To provide advanced features and security, you can create volume types and enable encryption. A volume type is a set of characteristics that define a volume, such as its size, performance, and availability. Encryption is a feature that encrypts the data stored on the volume to provide additional security.

- Attach volumes to instances: Finally, you can attach the volumes to instances and use them as persistent storage. The volumes can be attached and detached from instances as needed, and they can be used to store data that needs to persist across instance reboots or terminations

4. Overall, deploying block storage in OpenStack requires careful planning and configuration, but it is a critical step in creating a functional and scalable cloud infrastructure.

5. Overall, deploying Block Storage in OpenStack requires careful planning and configuration, but it is a critical step in providing persistent storage to instances and enabling advanced features and security.

6.11 COMPUTE DEPLOYMENT

1. Compute deployment, deploying and utilizing OpenStack in production environments

2. Compute deployment is a critical component of deploying and utilizing OpenStack in production environments. Here are the steps involved in deploying and utilizing OpenStack Compute in a production environment:

- Install and configure the Nova compute service: The first step is to install and configure the Nova compute service on each compute node. This involves installing the necessary packages, configuring the Nova compute service, and setting up the networking.

- Configure the hypervisor: Next, you need to configure the hypervisor, which is the software layer that enables virtualization on the compute nodes. There are different types of hypervisors that can be used with OpenStack Compute, such as KVM, Xen, and VMware. You need to configure the hypervisor based on the type of virtualization you are using.

- Create and manage instances: Once the Nova compute service and hypervisor are configured, you can create and manage instances. An instance is a virtual machine that runs on the compute node and provides computing resources to users. You can create instances using the OpenStack dashboard, CLI, or API.

- Configure security and networking: To ensure security and connectivity, you need to configure security groups, firewalls, and networking. Security groups are sets of firewall rules that define which traffic is allowed in and out of an instance. Networking involves configuring the network interfaces and IP addresses of the instances.

- Monitor and troubleshoot: Finally, you need to monitor and troubleshoot the compute environment to ensure it is running smoothly. This involves monitoring the performance of the instances and the compute nodes, as well as identifying and resolving any issues that arise.

6.12 DEPLOYING AND UTILIZING OPENSTACK IN PRODUCTION ENVIRONMENTS

1. Deploying and utilizing OpenStack Compute in a production environment requires careful planning, configuration, and management. It is important to follow best practices and security guidelines to ensure the compute environment is secure, reliable, and scalable.

2. Deploying OpenStack Compute (Nova) involves installing and configuring the Nova services and components, including the Nova API, Nova compute nodes, Nova scheduler, and the Nova database. Here are the steps involved in deploying Nova:

- Install and configure the Nova services: The first step in deploying Nova is to install and configure the Nova services on the controller node. This involves configuring the Nova API, the Nova conductor, and the Nova database.

- Install and configure the Nova compute nodes: The next step is to install and configure the Nova compute nodes. This involves configuring the Nova compute service, setting up networking, and configuring the hypervisor.

- Configure the Nova scheduler: The Nova scheduler is responsible for selecting the appropriate compute node for a given instance. You can configure the scheduler to use different algorithms to select the node based on various criteria, such as available resources, affinity, and anti-affinity.

- Create flavors: Flavors are predefined templates that define the size, CPU, memory, and disk specifications of an instance. You can create different flavors based on the requirements of your applications and workloads.

- Create and manage instances: Once Nova is deployed, you can create and manage instances using the Nova API or the OpenStack dashboard. You can select the appropriate flavor for your instances, attach storage volumes, and configure networking.

3. To deploy and utilize OpenStack in production environments, it is essential to follow best practices and ensure high availability, scalability, and security. Here are some tips for deploying and utilizing OpenStack in production:

- Use high availability (HA) deployment: To ensure high availability and minimize downtime, it is recommended to deploy OpenStack in an HA configuration. This involves setting up multiple instances of the services, load balancers, and databases.

- Configure security: OpenStack provides several security features, such as role-based access control (RBAC), network security groups (NSGs), and encryption. It is essential to configure these features based on your security requirements to protect your cloud environment from unauthorized access.

- Monitor and manage resources: To ensure optimal performance and utilization of resources, it is recommended to monitor and manage

your OpenStack resources regularly. You can use OpenStack monitoring tools such as Ceilometer, Aodh, and Gnocchi to monitor the performance of your resources.

- Plan for scalability: OpenStack is designed to scale horizontally, which means you can add more nodes to handle increased workloads. It is essential to plan for scalability and ensure that your deployment can handle future growth.

- Ensure backup and disaster recovery: It is essential to ensure backup and disaster recovery of your OpenStack environment to protect against data loss and minimize downtime in the event of a disaster. You can use OpenStack backup and recovery tools such as Freezer and Barman to backup and recover your OpenStack environment.

4. Overall, deploying and utilizing OpenStack in production environments requires careful planning, configuration, and management. By following best practices and utilizing OpenStack's features and tools, you can build a robust, scalable, and secure cloud infrastructure that meets your business requirements.

6.13 BUILDING A PRODUCTION ENVIRONMENT

1. Building a production environment for OpenStack involves several steps and considerations. Here are some key steps:

- Planning: Before starting the deployment, plan the architecture of the production environment. This includes defining the number of nodes required for each component, network topology, storage requirements, and security considerations.

- Hardware Requirements: Ensure that the hardware meets the minimum requirements for running OpenStack. Consider using hardware that is scalable and can be easily upgraded.

- Operating System: Choose the operating system for the nodes, and ensure that it is compatible with the OpenStack version.

- Networking: Set up the networking infrastructure for the environment. Consider using redundant network paths, VLANs, and subnets.

- Storage: Choose the storage solution for the environment, and ensure that it is compatible with OpenStack. Consider using redundant storage systems for high availability.

- Install OpenStack: Install the OpenStack components on the nodes. Follow the documentation for the version of OpenStack being installed.

- Configure OpenStack: After installation, configure OpenStack according to the requirements of the production environment. This includes configuring compute, networking, and storage.

- Testing: After configuration, test the environment thoroughly to ensure that all components are working correctly.

- Monitoring: Set up monitoring for the production environment to ensure that it is running smoothly. Consider using monitoring tools such as Nagios, Zabbix, or Prometheus.

- Maintenance: Perform regular maintenance tasks such as patching, upgrades, and backups.

2. Building a production environment for OpenStack can be complex and time-consuming, but proper planning and execution can result in a highly scalable, flexible, and reliable cloud platform.

6.14 APPLICATION ORCHESTRATION USING OPENSTACK HEAT

1. OpenStack Heat is a service that provides orchestration capabilities to OpenStack. Heat enables automated provisioning of infrastructure and applications on top of OpenStack, by defining templates that describe the desired configuration of the resources.

2. Application orchestration using OpenStack Heat involves the following steps:

- Create a Heat template: A Heat template is a text file that defines the resources required to deploy an application. The template is written in YAML or JSON format and includes a description of the resources, their dependencies, and their configuration.

- Upload the template to Heat: Once the template is created, it needs to be uploaded to Heat. This can be done through the OpenStack CLI or web interface.

- Launch the stack: After the template is uploaded, it can be used to launch a stack. A stack is a collection of resources that are created and managed by Heat. The stack creation process involves validating the template, creating the necessary resources, and configuring them according to the template.

- Monitor the stack: After the stack is launched, it can be monitored through the OpenStack web interface or CLI. Heat provides visibility into the status of the resources, and the ability to perform actions on them, such as scaling or deleting.

- Update the stack: If changes need to be made to the stack, the Heat template can be modified and uploaded to Heat. Heat will then update the stack by making the necessary changes to the resources.

3. By using Heat for application orchestration, it is possible to automate the deployment and management of complex applications on OpenStack. Heat provides a standardized way of defining infrastructure and application resources, making it easier to manage and scale deployments.

3.15 SUMMARY

In this chapter we learned about openstack and its components. The summing of all points as follows

- OpenStack has a large and active community of developers, users, and vendors, who contribute to the development and support of the platform.

- The OpenStack APIs are available for all OpenStack services, including Compute, Networking, Identity, Image, and Block Storage. The APIs are based on industry standards such as JSON, XML, and HTTP.

- Deploying block storage in OpenStack requires careful planning and configuration, but it is a critical step in creating a functional and scalable cloud infrastructure.

- Building a production environment for OpenStack can be complex and time-consuming, but proper planning and execution can result in a highly scalable, flexible, and reliable cloud platform.

3.16 QUESTIONS

1. What is openstack?
2. Write a short note on
   i. OpenStack test-drive
   ii. Basic OpenStack operations
   iii. OpenStack CLI and APIs
   iv. Tenant model operations
   v. Quotas, Private cloud building blocks
3. Explain the following concepts in detail.
   i. Controller deployment
   ii. Networking deployment

iii. Block Storage deployment

iv. Compute deployment

v. deploying and utilizing OpenStack in production environments

4. Illustrate the concept of Building a production environment and Application orchestration using OpenStack Heat?

6.17 REFERENCES

1. OpenStack Essentials, Dan Radez, PACKT Publishing, 2015

2. OpenStack Operations Guide, Tom Fifield, Diane Fleming, Anne Gentle, Lorin Hochstein, Jonathan Proulx, Everett Toews, and Joe Topjian, O'Reilly Media, Inc., 2014

3. https://www.openstack.org