ANNEXURE 2.2.1 - A

K.S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura main Road, Bengaluru – 560 109

Department of Computer Science & Engineering

QUESTION BANK WITH ANSWERS

FOR SLOW LEARNERS

CLOUD COMPUTING AND ITS APPLICATIONS

18CS643

PREPARED BY,

Mrs. SUPREETHA GANESH
ASSISTANT PROFESSOR
DEPT. OF CSE

Ms. NAMYAPRIYA D
ASSISTANT PROFESSOR
DEPT. OF CSE

With Guidance from: Dr. PRADEEP V DESAI
K S INSTITUTE OF TECHNOLOGY
#14, Raghuvanahalli, Kanakapura main Road,
Bengaluru – 560 109

CONTENTS

SL NO	MODULES	PAGE NO
1	**MODULE 1** Chapter 1: Introduction ,Cloud Computing at a Glance, The Vision of Cloud Computing, Defining a Cloud, A Closer Look, Cloud Computing Reference Model, Characteristics and Benefits, Challenges Ahead, Historical Developments, Distributed Systems, Virtualization, Web 2.0, Service-Oriented Computing, Utility-Oriented Computing, Building Cloud Computing Environments, Application Development, Infrastructure and System Development, Computing Platforms and Technologies, Amazon Web Services (AWS), Google AppEngine, Microsoft Azure, Hadoop, Force.com and Salesforce.com, Manjrasoft Aneka Chapter 2: Virtualization, Introduction, Characteristics of Virtualized, Environments Taxonomy of Virtualization Techniques, Execution Virtualization, Other Types of Virtualization, Virtualization and Cloud Computing, Pros and Cons of Virtualization, Technology Examples Xen: Paravirtualization, VMware: Full Virtualization, Microsoft Hyper-V	1
2	**MODULE 2** Cloud Computing Architecture, Introduction, Cloud Reference Model, Architecture, Infrastructure / Hardware as a Service, Platform as a Service, Software as a Service, Types of Clouds, Public Clouds, Private Clouds, Hybrid Clouds, Community Clouds, Economics of the Cloud, Open Challenges, Cloud Definition, Cloud Interoperability and Standards Scalability and Fault Tolerance Security, Trust, and Privacy Organizational Aspects Aneka: Cloud Application Platform, Framework Overview, Anatomy of the Aneka Container, From the Ground Up: Platform Abstraction Layer, Fabric Services, foundation Services, Application Services, Building Aneka Clouds, Infrastructure Organization, Logical Organization, Private Cloud Deployment Mode, Public Cloud Deployment Mode, Hybrid Cloud Deployment Mode, Cloud Programming and Management, Aneka SDK, Management Tools	21
3	**MODULE 3** Concurrent Computing: Thread Programming, Introducing Parallelism for Single Machine Computation, Programming Applications with Threads, What is a Thread?, Thread APIs, Techniques for Parallel Computation with Threads, Multithreading with Aneka, Introducing the Thread Programming Model, Aneka Thread vs. Common Threads, Programming Applications with Aneka Threads, Aneka Threads Application Model, Domain Decomposition: Matrix Multiplication, Functional Decomposition: Sine, Cosine, and Tangent. High-Throughput Computing: Task Programming, Task Computing, characterizing a Task, Computing Categories, Frameworks for Task Computing, Task-based Application Models, Embarrassingly Parallel Applications, Parameter Sweep Applications, MPI Applications, Workflow Applications with Task Dependencies, Aneka Task-Based	46

	Programming, Task Programming Model, Developing Applications with the Task Model, Developing Parameter Sweep Application, Managing Workflows.	
4	**MODULE 4** Data Intensive Computing: Map-Reduce Programming, what is Data-Intensive Computing? Characterizing Data-Intensive Computations, Challenges Ahead, Historical Perspective, Technologies for Data-Intensive Computing, Storage Systems, Programming Platforms, Aneka MapReduce Programming, Introducing the MapReduce Programming Model, Example Application	**67**
5	**MODULE 5** Cloud Platforms in Industry, Amazon Web Services, Compute Services, Storage Services, Communication Services, Additional Services, Google AppEngine, Architecture and Core Concepts, Application Life-Cycle, Cost Model, Observations, Microsoft Azure, Azure Core Concepts, SQL Azure, Windows Azure Platform Appliance. Cloud Applications Scientific Applications, Healthcare: ECG Analysis in the Cloud, Biology: Protein Structure Prediction, Biology: Gene Expression Data Analysis for Cancer Diagnosis, Geoscience: Satellite Image Processing, Business and Consumer Applications, CRM and ERP, Productivity, Social Networking, Media Applications, Multiplayer Online Gaming.	**83**

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
MODEL QUESTION BANK 2022-23 EVEN SEMESTER

Degree:	**B. E**	**Semester:**	**VI**
Branch:	**COMPUTER SCIENCE & ENGINEERING**	**Course Code:**	**18CS643**
Course Title:	**CLOUD COMPUTING AND ITS APPLICATIONS Q&A from 2018 to 2023**		

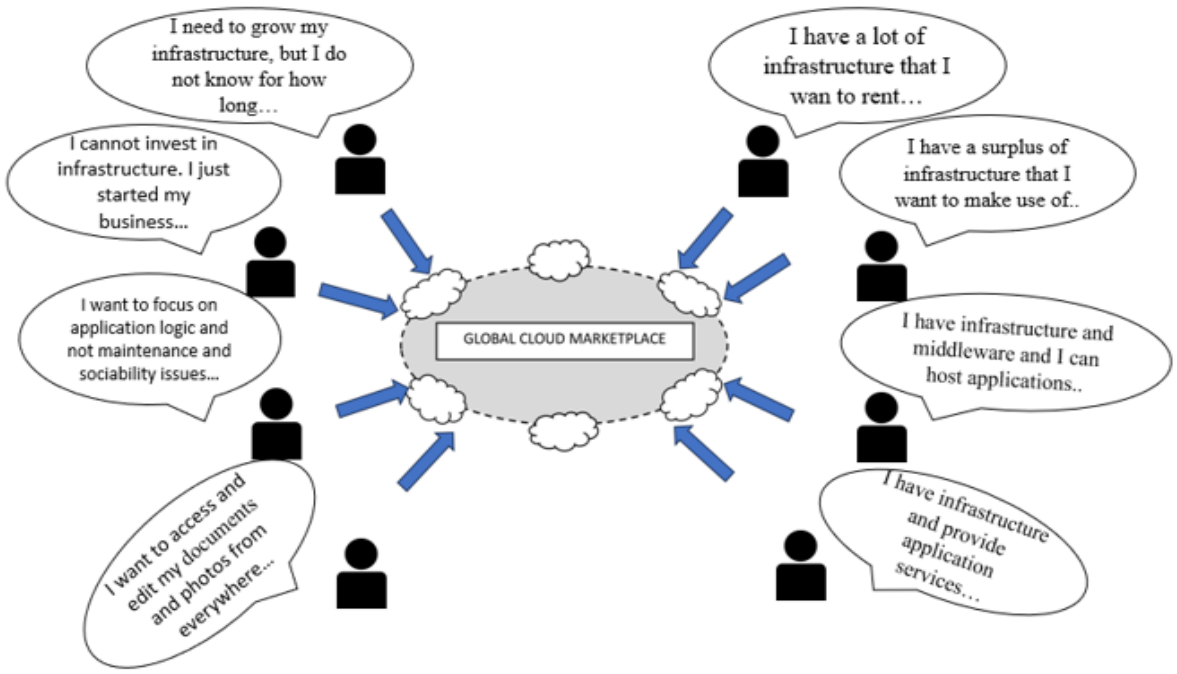
MODULE 1

1. What is Cloud? What is the vision of Cloud Computing? (6M , Dec-2018)

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

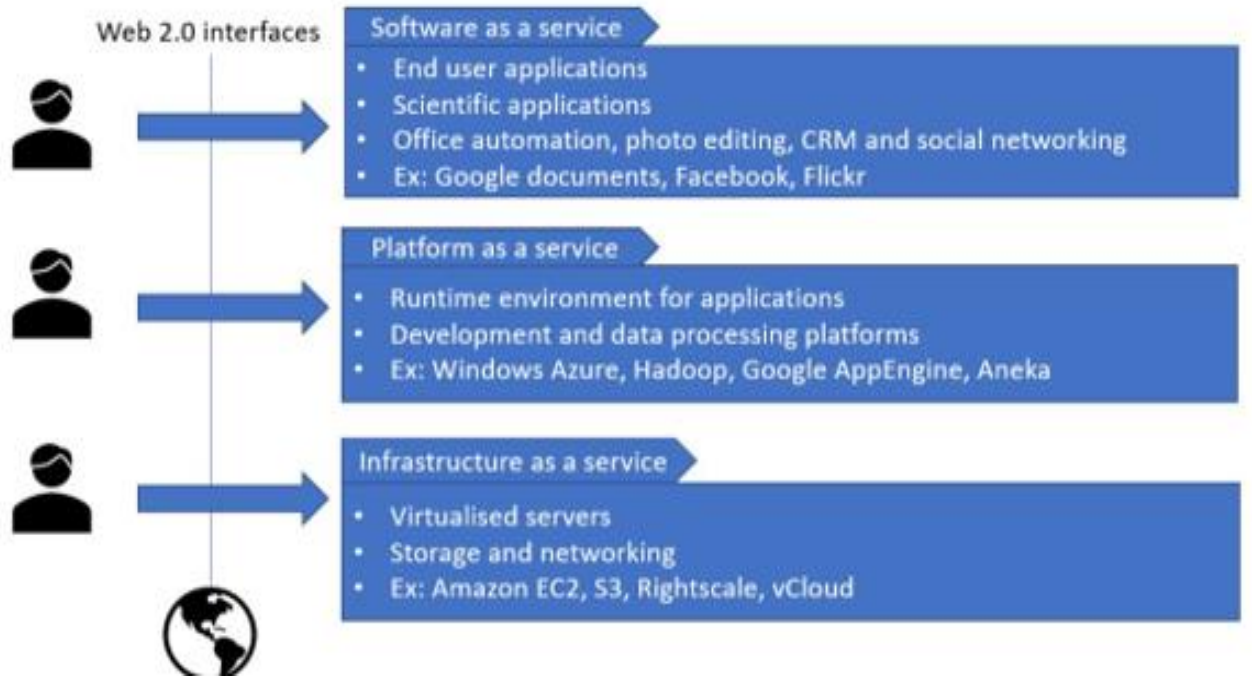
The vision of cloud computing

- Vision of cloud computing is that in the near future it will be possible to find the solution that matches the customer needs by simply entering our request in a global digital market that trades cloud computing services.
- Cloud computing allows anyone with a credit card to provision virtual hardware, runtime environments, and services. These are used for as long as needed, with no up-front commitments required.
- The entire stack of a computing system is transformed into a collection of utilities, which can be provisioned and composed together to deploy systems in hours rather than days and with virtually no maintenance costs.



2. What is Cloud Computing? With the neat diagram explain Cloud Computing reference model. (10M,Jan-2018, Jan-2023, July-2022, Feb-2022)

Solution: Cloud computing refers to both the applications delivered as services over the Internet and the hardware and system software in the datacenters that provide those services. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.



Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). These categories are related to each other as described in the model organizes the wide range of cloud computing services into a layered view that walks the computing stack from bottom to top.

- **Infrastructure-as-a-Service**:
  - solutions deliver infrastructure on demand in the form of virtual hardware, storage, and networking.
  - Virtual hardware is utilized to provide compute on demand in the form of virtual machine instances. These are created at users' request on the provider's infrastructure, and users are given tools and interfaces to configure the software stack installed in the virtual machine. The pricing model is usually defined in terms of some amount per hour.
  - Virtual storage is delivered in the form of raw disk space or object store. Virtual networking identifies the collection of services that manage the networking among virtual instances and their connectivity to the Internet or private networks.

Platform-as-a-Service

- solutions are the next step in the stack. They deliver scalable and elastic runtime environments on demand and host the execution of applications. These services are backed by a core middleware platform that is responsible for creating the abstract environment where applications are deployed and executed.
- It is the responsibility of the service provider to provide scalability and to manage fault tolerance, while users are requested to focus on the logic of the application developed by leveraging the provider's APIs and libraries.
- This approach increases the level of abstraction at which cloud computing is leveraged but also constrains the user in a more controlled environment.

Software-as-a-Service:

- solutions provide applications and services on demand. Most of the common functionalities of desktop applications—such as office automation, document management, photo editing, and customer relationship management (CRM) software—are replicated on the provider's infrastructure and made more scalable and accessible through a browser on demand. These applications are shared across multiple users whose interaction is isolated from the other users.
- The SaaS layer is also the area of social networking Websites, which leverage cloud-based infrastructures to sustain the load generated by their popularity. Each layer provides a different service to users. IaaS solutions are sought by users who want to leverage cloud computing from building dynamically scalable computing systems requiring a specific software stack. IaaS services are therefore used to develop scalable Websites or for back- ground processing.
- PaaS solutions provide scalable programming platforms for developing applications and are more appropriate when new systems have to be developed. SaaS solutions target mostly end users who want to benefit from the elastic scalability of the cloud without doing any software development, installation, configuration, and maintenance.

3. Brief the Characteristics, Benefits and Challenges of Cloud Computing.
(8M,9 M,Dec2018, July-2022)

Solution:

Characteristics and Benefits of Cloud Computing

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

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

Challenges Ahead

1. Data Security and Privacy

- Data security is a major concern when switching to cloud computing. User or organizational data stored in the cloud is critical and private. Even if the cloud service provider assures data integrity, it is your responsibility to carry out user authentication and authorization, identity management, data encryption, and access control.
- Security issues on the cloud include identity theft, data breaches, malware infections, and a lot more which eventually decrease the trust amongst the users of your applications. This can in turn lead to potential loss in revenue alongside reputation and stature. Also, dealing with cloud computing requires sending and receiving huge amounts of data at high speed, and therefore is susceptible to data leaks.

2. Cost Management

- Even as almost all cloud service providers have a "Pay As You Go" model, which reduces the overall cost of the resources being used, there are times when there are huge costs incurred to the enterprise using cloud computing. When there is under optimization of the resources, let's say that the servers are not being used to their full potential, add up to the hidden costs. If there is a degraded application performance or sudden spikes or overages in the usage, it adds up to the overall cost.
- Unused resources are one of the other main reasons why the costs go up. If you turn on the services or an instance of cloud and forget to turn it off during the weekend or when there is no current use of it, it will increase the cost without even using the resources.

3. Multi-Cloud Environments

- Due to an increase in the options available to the companies, enterprises not only use a single cloud but depend on multiple cloud service providers. Most of these companies use hybrid cloud tactics and close to 84% are dependent on multiple clouds.
- This often ends up being hindered and difficult to manage for the infrastructure team. The process most of the time ends up being highly complex for the IT team due to the differences between multiple cloud providers.

4. Performance Challenges

- Performance is an important factor while considering cloud-based solutions. If the performance of the cloud is not satisfactory, it can drive away users and decrease profits. Even a little latency while loading an app or a web page can result in a huge drop in the percentage of users.
- This latency can be a product of inefficient load balancing, which means that the server cannot efficiently split the incoming traffic so as to provide the best user experience. Challenges also arise in the case of fault tolerance, which means the operations continue as required even when one or more of the components fail.

5. Interoperability and Flexibility

- When an organization uses a specific cloud service provider and wants to switch to another cloud-based solution, it often turns up to be a tedious procedure since applications written for one cloud with the application stack are required to be re-written for the other cloud.
- There is a lack of flexibility from switching from one cloud to another due to the complexities involved. Handling data movement, setting up the security from scratch and network also add up to the issues encountered when changing cloud solutions, thereby reducing flexibility.

6. High Dependence on Network

- Since cloud computing deals with provisioning resources in real-time, it deals with enormous amounts of data transfer to and from the servers. This is only made possible due to the availability of the high-speed network.
- Even when the enterprises can cut their hardware costs, they need to ensure that the internet bandwidth is high as well there are zero network outages, or else it can result in a potential business loss. It is therefore a major challenge for smaller enterprises that have to maintain network bandwidth that comes with a high cost.

7. Lack of Knowledge and Expertise

- Due to the complex nature and the high demand for research working with the cloud often ends up being a highly tedious task. It requires immense knowledge and wide expertise on the subject. .
- Cloud computing is a highly paid job due to the extensive gap between demand and supply. There are a lot of vacancies but very few talented cloud engineers, developers, and professionals. Therefore, there is a need for upskilling so these professionals can actively understand, manage and develop cloud-based applications with minimum issues and maximum reliability.

4. Briefly explain the core technologies and platform that play an important role in Cloud Computing. 8M ,Jan-2018, Jan-2023, July-2022, Feb-2022)

Solution:

Development of a cloud computing application happens by leveraging platforms and frameworks that provide different types of services, from the bare-metal infrastructure to customizable applications serving specific purposes.

1. Amazon web services (AWS)
2. Google AppEngine
3. Microsoft Azure
4. Hadoop
5. Force.com and Salesforce.com
6. Manjrasoft Aneka

- **Amazon web services (AWS)**: AWS offers comprehensive cloud IaaS services ranging from virtual compute, storage, and networking to complete computing stacks. AWS is mostly known for its compute and storage on- demand services, namely Elastic Compute Cloud (EC2) and Simple Storage Service (S3).
- EC2 provides users with customizable virtual hardware that can be used as the base infrastructure for deploying computing systems on the cloud. It is possible to choose from a large variety of virtual hardware configurations, including GPU and cluster instances. S3 is organized into buckets; these are containers of objects that are stored in binary form. Users can store objects of any size, from simple files to entire disk images, and have them accessible from everywhere.

1. **Google AppEngine:** Google AppEngine is a scalable runtime environment mostly devoted to executing Web applications. AppEngine provides both a secure execution environment and a collection of services that simplify the development of scalable and high-performance Web applications. These services include in-memory caching, scalable data store, job queues, messaging, and cron tasks. Developers can build and test applications on their own machines using the AppEngine software development kit (SDK), which replicates the production runtime environment and helps test and profile applications. Once development is complete, developers can easily migrate their application to AppEngine, and make the application available to the world. The languages currently supported are Python, Java.
2. **Microsoft Azure:** Microsoft Azure is a cloud operating system and a platform for developing applications in the cloud. It provides a scalable runtime environment for Web applications and distributed applications in general. Applications in Azure are organized around the concept of roles. Currently, there are three types of role: Web role, worker role, and virtual machine role. The Web role is designed to host a Web application, the worker role is a more generic container of applications and can be used to perform workload processing, and the virtual machine role provides a virtual environment in which the computing stack can be fully customized, including the operating systems.
3. **Hadoop Apache:** Hadoop is an open-source framework that is suited for processing large data sets on commodity hardware. Yahoo!, the sponsor of the Apache Hadoop project, has put considerable effort into transforming the project into an enterprise-ready cloud computing platform for data processing. Hadoop is an integral part of the Yahoo! cloud infrastructure and supports several business processes of the company. Currently, Yahoo! manages the largest Hadoop cluster in the world.
4. **Force.com and Salesforce.com:** Force.com is a cloud computing platform for developing social enterprise applications. Force.com allows developers to create applications by composing ready-to-use blocks; a complete set of components supporting all the activities of an enterprise are available. The Force.com platform is completely hosted on the cloud and provides complete access to its functionalities and those implemented in the hosted applications through Web services technologies.
5. **Manjrasoft Aneka:** Manjrasoft Aneka is a cloud application platform for rapid creation of scalable applications and their deployment on various types of clouds in a seamless and elastic manner. It supports a collection of programming abstractions for developing applications and a distributed runtime environment that can be deployed on heterogeneous hardware (clusters,

networked desktop computers, and cloud resources). These platforms are key examples of technologies available for cloud computing. They mostly fall into the three major market segments identified in the reference model: Infrastructure-as-aService, Platform-as-a-Service, and Software-as-a-Service.

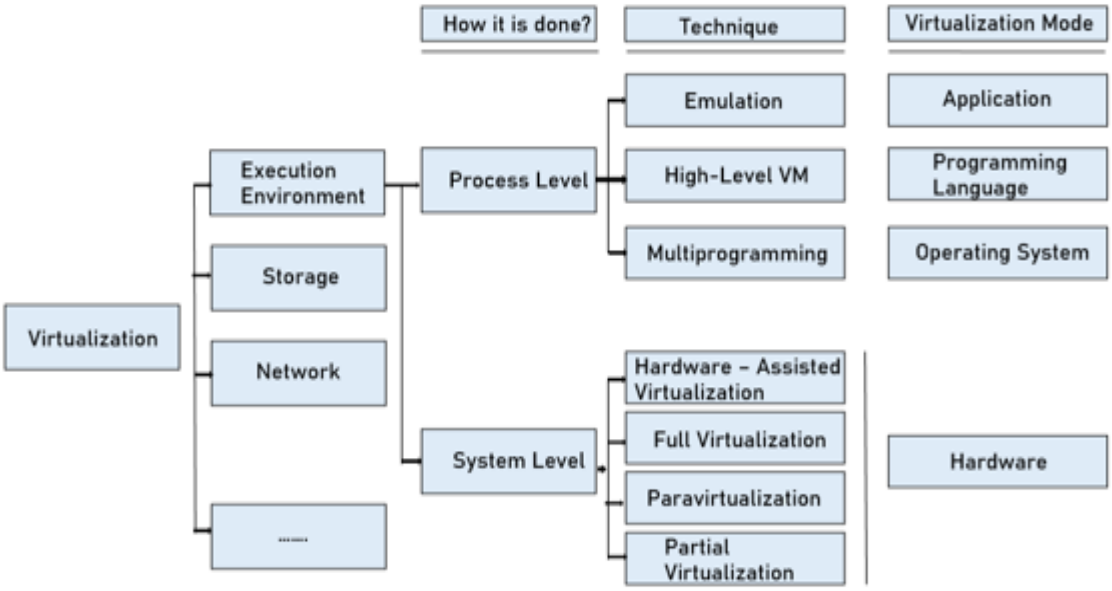
5. What is Virtualization? Explain Taxonomy of Virtualization? (8M,Jan-2023, July-2022)

Solution:

- Virtualization is a large umbrella of technologies and concepts that are meant to provide an abstract environment—whether virtual hardware or an operating system—to run applications.
- The term virtualization is often synonymous with hardware virtualization, which plays a fundamental role in efficiently delivering Infrastructure-as-a-Service (IaaS) solutions for cloud computing.
- In fact, virtualization technologies are available in many flavors by providing virtual environments at the operating system level, the programming language level, and the application level.

Taxonomy of virtualization techniques

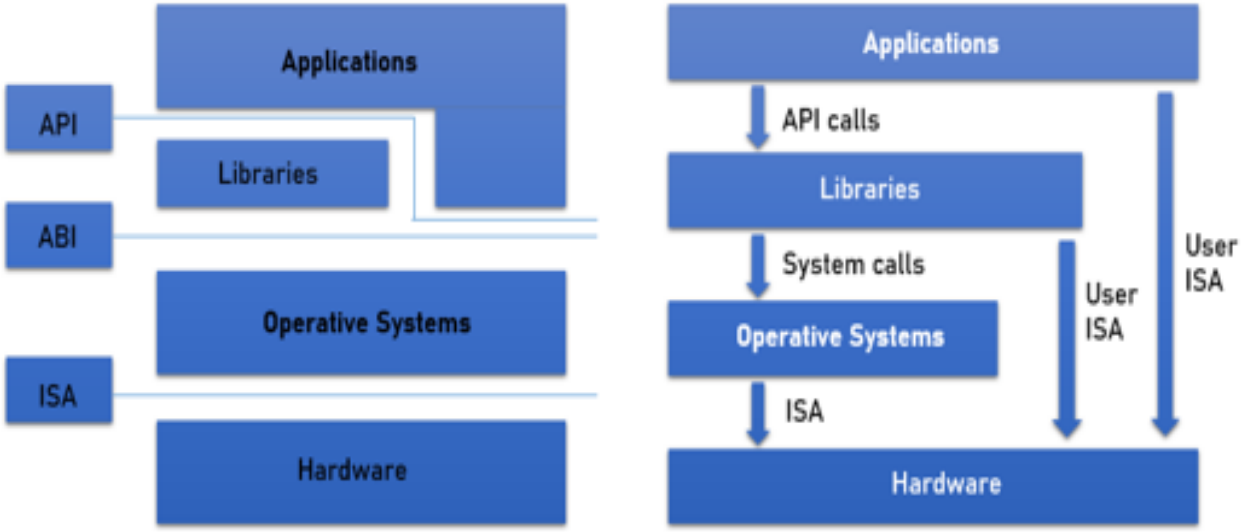
Virtualization is mainly used to emulate execution environments, storage, and networks. Among these categories, execution virtualization constitutes the oldest, most popular, and most developed area.



In particular we can divide these execution virtualization techniques 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 control of the hardware. System-level techniques are implemented directly on hardware and do not require—or require a minimum of support from—an existing operating system. Within these two categories we can list various techniques that offer the guest a different type of virtual computation environment: bare hardware, operating system resources, low-level programming language, and application libraries.

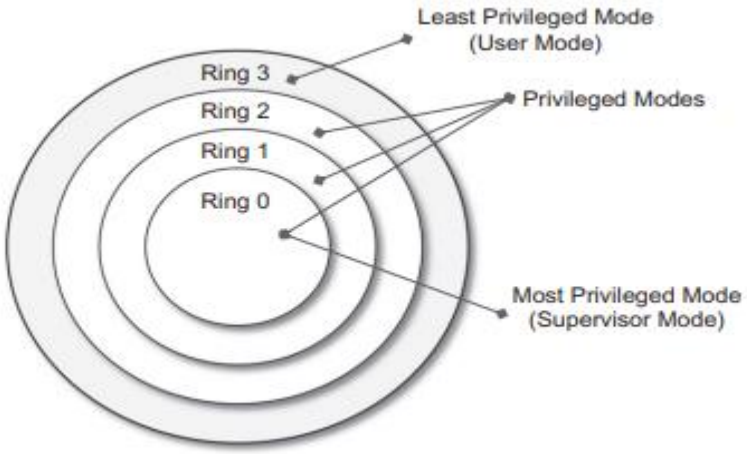
Execution virtualization: Execution virtualization includes all techniques that aim to emulate an execution environment that is separate from the one hosting the virtualization layer. All these techniques concentrate their interest on providing support for the execution of programs, whether these are the operating system, a binary specification of a program compiled against an abstract machine model, or an application. Therefore, execution virtualization can be implemented directly on top of the hardware by the operating system, an application, or libraries dynamically or statically linked to an application image.

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 perspective, virtualization techniques actually replace one of the layers and intercept the calls that are directed toward it.
Modern computing systems can be expressed in terms of the reference model. At the bottom layer, the model for the hardware is expressed in terms of the Instruction Set Architecture (ISA), which defines the instruction set for the processor, registers, memory, and interrupts management. ISA is the interface between hardware and software, and it is important to the operating system (OS) developer (System ISA) and developers of applications that directly manage the underlying hardware (User ISA).



Application Binary Interface (ABI) separates the operating system layer from the applications and libraries, which are managed by the OS. ABI covers details such as low-level data types, alignment, and call conventions and defines a format for executable programs. System calls are defined at this level. This interface allows portability of applications and libraries across operating systems that implement the same ABI. The highest level of abstraction is represented by the application programming interface (API), which interfaces applications to libraries and/or the underlying operating system.
The instruction set exposed by the hardware has been divided into different security classes that define who can operate with them. The first distinction can be made between privileged and no nprivileged instructions. Non privileged instructions are those instructions that can be used without

interfering with other tasks because they do not access shared resources. This category contains, for example, all the floating, fixed-point, and arithmetic instructions. Privileged instructions are those that are executed under specific restrictions and are mostly used for sensitive operations, which expose (behavior-sensitive) or modify (control-sensitive) the privileged state. For instance, behavior-sensitive instructions are those that operate on the I/O, whereas control-sensitive instructions alter the state of the CPU registers.



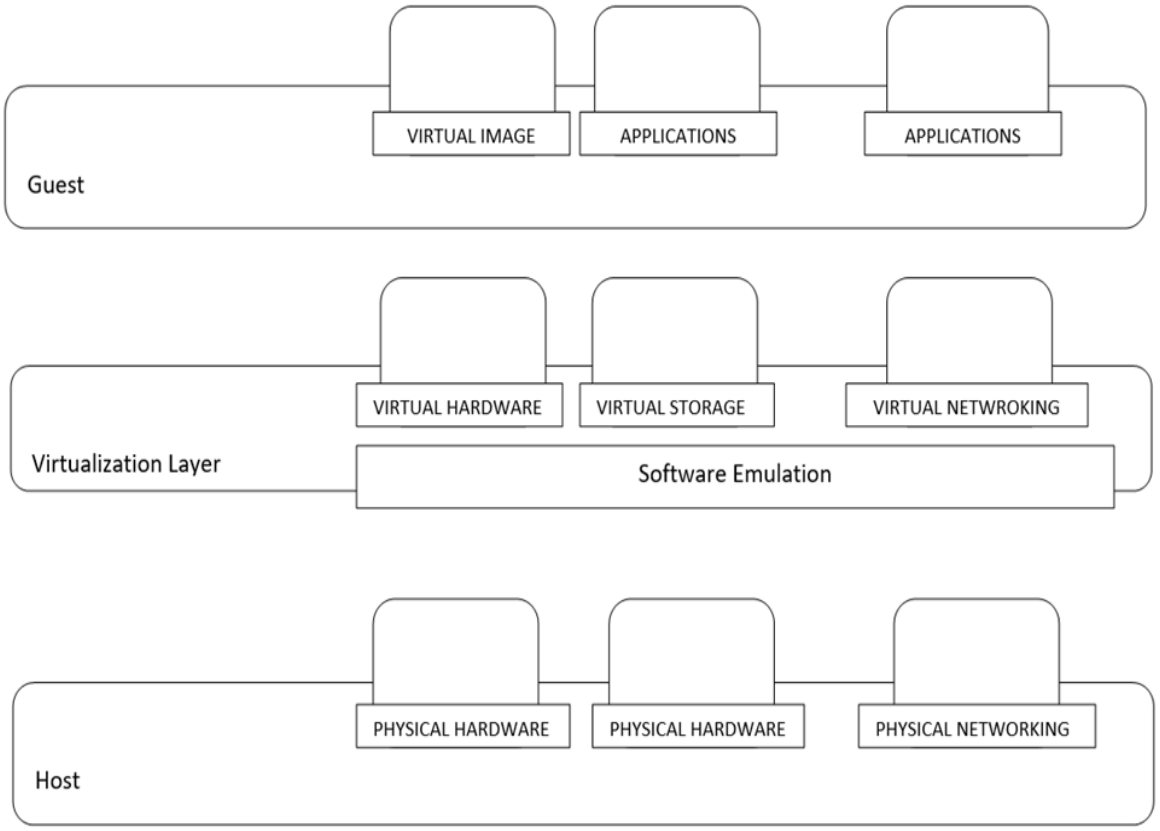
All the current systems support at least two different execution modes: supervisor mode and user mode. The first mode denotes an execution mode in which all the instructions (privileged and nonprivileged) can be executed without any restriction. This mode, also called master mode or kernel mode, is generally used by the operating system (or the hypervisor) to perform sensitive operations on hardware level resources. In user mode, there are restrictions to control the machine-level resources. If code running in user mode invokes the privileged instructions, hardware interrupts occur and trap the potentially harmful execution of the instruction. Conceptually, the hypervisor runs above the supervisor mode.

6. Explain the characteristics of Virtualized Environment. (8M,July-2022, Feb-2022)

Solution:

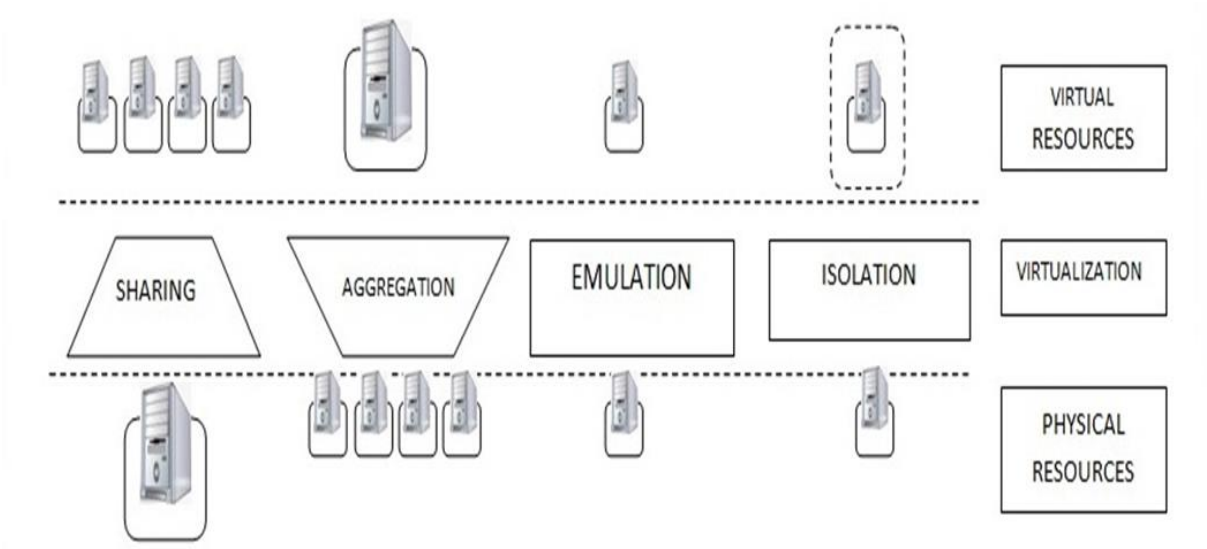
Characteristics of virtualized environments:

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



1. **Increased security:** The ability to control the execution of a guest in a completely transparent manner opens new possibilities for delivering a secure, controlled execution environment. The virtual machine represents an emulated environment in which the guest is executed. All the operations of the guest are generally performed against the virtual machine, which then translates and applies them to the host. Resources exposed by the host can then be hidden or simply protected from the guest. Sensitive information that is contained in the host can be naturally hidden without the need to install complex security policies.

2. **Managed execution:** Virtualization of the execution environment not only allows increased security, but a wider range of features also can be implemented. In particular, sharing, aggregation, emulation, and isolation are the most relevant features



• **Aggregation:** Not only is it possible to share physical resources among several guests, but virtualization also allows aggregation, which is the opposite process. A group of separate hosts can be tied together and represented to guests as a single virtual host. 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.

• **Emulation:** Guest programs are executed within an environment that is controlled by the virtualization layer, which ultimately is a program. This allows for controlling and tuning the environment that is exposed to guests. For instance, a completely different environment with respect to the host can be emulated, thus allowing the execution of guest programs requiring specific characteristics that are not present in the physical host. Hardware virtualization solutions are able to provide virtual hardware and emulate a particular kind of device such as Small Computer System Interface (SCSI) devices for file I/O, without the hosting machine having such hardware installed.

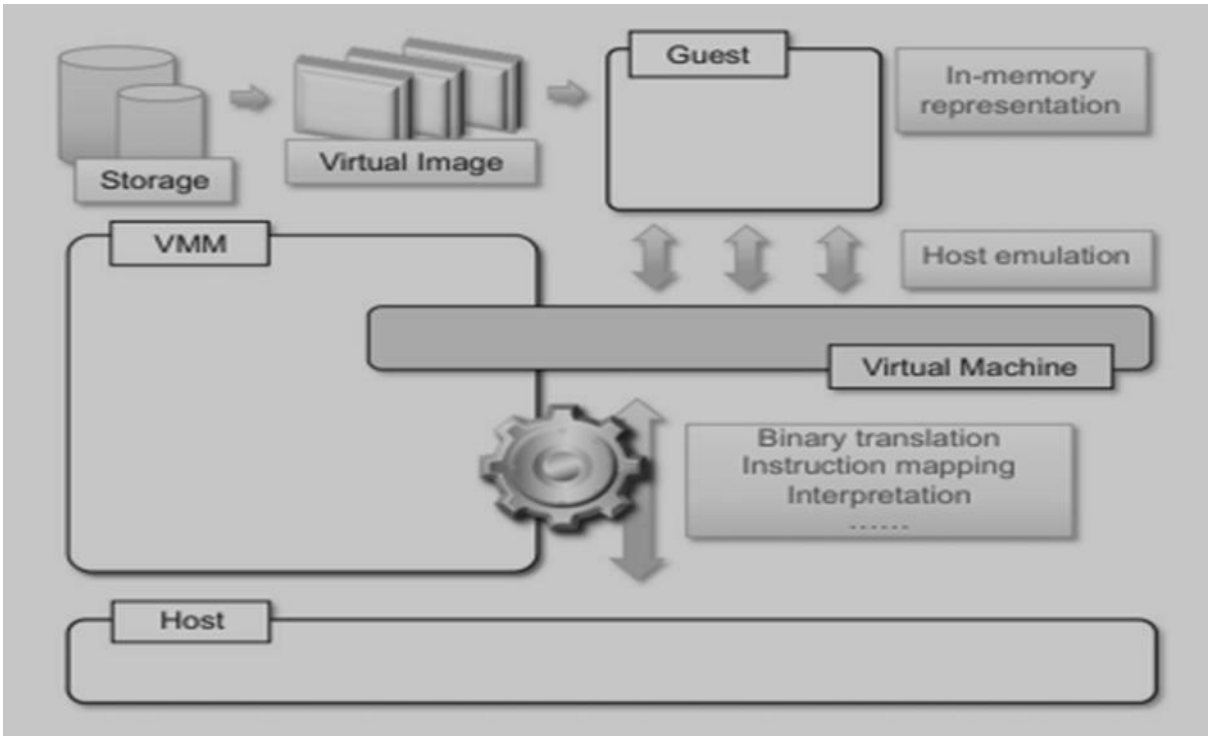
• **Isolation:** Virtualization allows providing guests—whether they are operating systems, applications, or other entities—with a completely separate environment, in which they are executed. The guest program performs its activity by interacting with an abstraction layer, which provides access to the underlying resources. The virtual machine can filter the activity of the guest and prevent harmful operations against the host.

3. Portability: The concept of portability applies in different ways according to the specific type of virtualization considered. In the case of a hardware virtualization solution, the guest is packaged into a virtual image that, in most cases, can be safely moved and executed on top of different virtual machines. Except for the file size, this happens with the same simplicity with which we can display a picture image in different computers. Virtual images are generally proprietary formats that require a specific virtual machine manager to be executed.

7. Explain different levels of Virtualization (10M,Aug2022)

Solution:

Hardware-level virtualization:



- Hardware-level virtualization is a virtualization technique that provides an abstract execution environment in terms of computer hardware on top of which a guest operating system can be run. In this model, the guest is represented by the operating system, the host by the physical computer hardware, the virtual machine by its emulation, and the virtual machine manager by the hypervisor.
- The hypervisor is generally a program or a combination of software and hardware that allows the abstraction of the underlying physical hardware. Hardware-level virtualization is also called system virtualization, since it provides ISA to virtual machines, which is the representation of the hardware interface of a system.

Operating system-level virtualization:

- Operating system-level virtualization offers the opportunity to create different and separated execution environments for applications that are managed concurrently. Differently from hardware virtualization, there is no virtual machine manager or hypervisor, and the virtualization is done within a single operating system, where the OS kernel allows for multiple isolated user space instances.
- The kernel is also responsible for sharing the system resources among instances and for limiting the impact of instances on each other. A user space instance in general contains a proper view of the file system, which is completely isolated, and separate IP addresses, software configurations, and access to devices.
- Operating system-level virtualization aims to provide separated and multiple execution containers for running applications. Compared to hardware virtualization, this strategy imposes little or no overhead because applications directly use OS system calls and there is no need for emulation. Examples of operating system-level virtualizations are FreeBSD Jails,

IBM Logical Partition (LPAR), SolarisZones and Containers, Parallels Virtuozzo Containers, OpenVZ, iCore Virtual Accounts, Free Virtual Private Server (FreeVPS).

Programming language-level virtualization:

- Programming language-level virtualization is mostly used to achieve ease of deployment of applications, managed execution, and portability across different platforms and operating systems. It consists of a virtual machine executing the byte code of a program, which is the result of the compilation process. Compilers implemented and used this technology to produce a binary format representing the machine code for an abstract architecture. The characteristics of this architecture vary from implementation to implementation.
- Programming language-level virtualization has a long trail in computer science history and originally was used in 1966 for the implementation of Basic Combined Programming Language (BCPL), a language for writing compilers and one of the ancestors of the C programming language. Other important examples of the use of this technology have been the UCSD Pascal and Smalltalk. Virtual machine programming languages become popular again with Sun's introduction of the Java platform in 1996.
- Currently, the Java platform and .NET Framework represent the most popular technologies for enterprise application development. The main advantage of programming-level virtual machines, also called process virtual machines, is the ability to provide a uniform execution environment across different platforms. Programs compiled into byte code can be executed on any operating system and platform for which a virtual machine able to execute that code has been provided.

Application-level virtualization:

Application-level virtualization is a technique allowing applications to be run in runtime environments that do not natively support all the features required by such applications. In this scenario, applications are not installed in the expected runtime environment but are run as though they were.

In general, these techniques are mostly concerned with partial file systems, libraries, and operating system component emulation. Such emulation is performed by a thin layer—a program or an operating system component—that is in charge of executing the application. Emulation can also be used to execute program binaries compiled for different hardware architectures. In this case, one of the following strategies can be implemented:

• **Interpretation:** In this technique every source instruction is interpreted by an emulator for executing native ISA instructions, leading to poor performance. Interpretation has a minimal startup cost but a huge overhead, since each instruction is emulated.

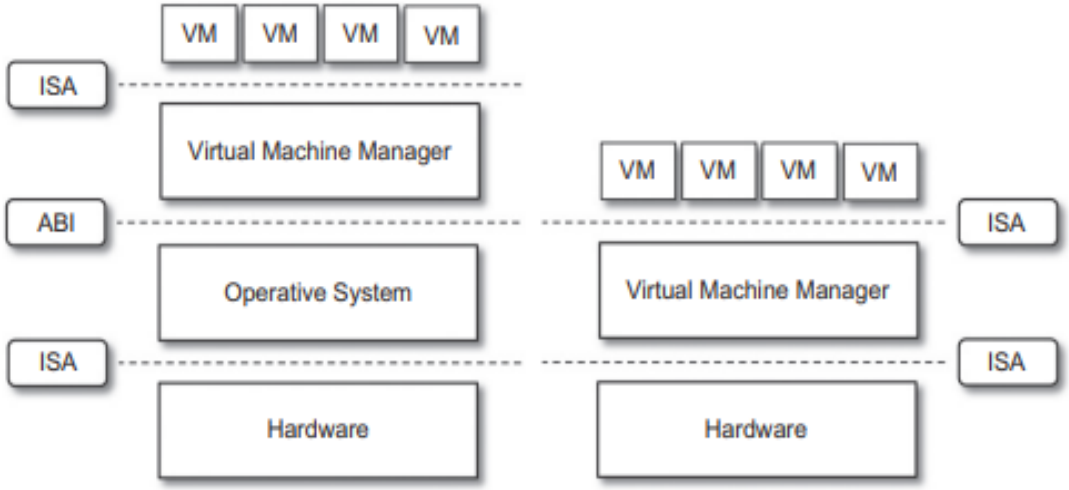
• **Binary translation:** In this technique every source instruction is converted to native instructions with equivalent functions. After a block of instructions is translated, it is cached and reused. Binary translation has a large initial overhead cost, but over time it is subject to better performance, since previously translated instruction blocks are directly executed. Application virtualization is a good solution in the case of missing libraries in the host operating system. One of the most popular solutions implementing application virtualization is Wine, which is a software application allowing Unix-like operating systems to execute programs written for the Microsoft Windows platform.

8. Explain Virtual machine manager and Hypervisors (8M,Jan-2023)

Solution:

Hypervisors: A fundamental element of hardware virtualization is the hypervisor, or virtual machine manager (VMM). It recreates a hardware environment in which guest operating systems are installed. There are two major types of hypervisors: Type I and Type II

- Type I hypervisors run directly on top of the hardware. Therefore, they take the place of the operating systems and interact directly with the ISA interface exposed by the underlying hardware, and they emulate this interface in order to allow the management of guest operating systems. This type of hypervisor is also called a native virtual machine since it runs natively on hardware.
- Type II hypervisors require the support of an operating system to provide virtualization services. This means that they are programs managed by the operating system, which interact with it through the ABI and emulate the ISA of virtual hardware for guest operating systems.



This type of hypervisor is also called a hosted virtual machine since it is hosted within an operating system.

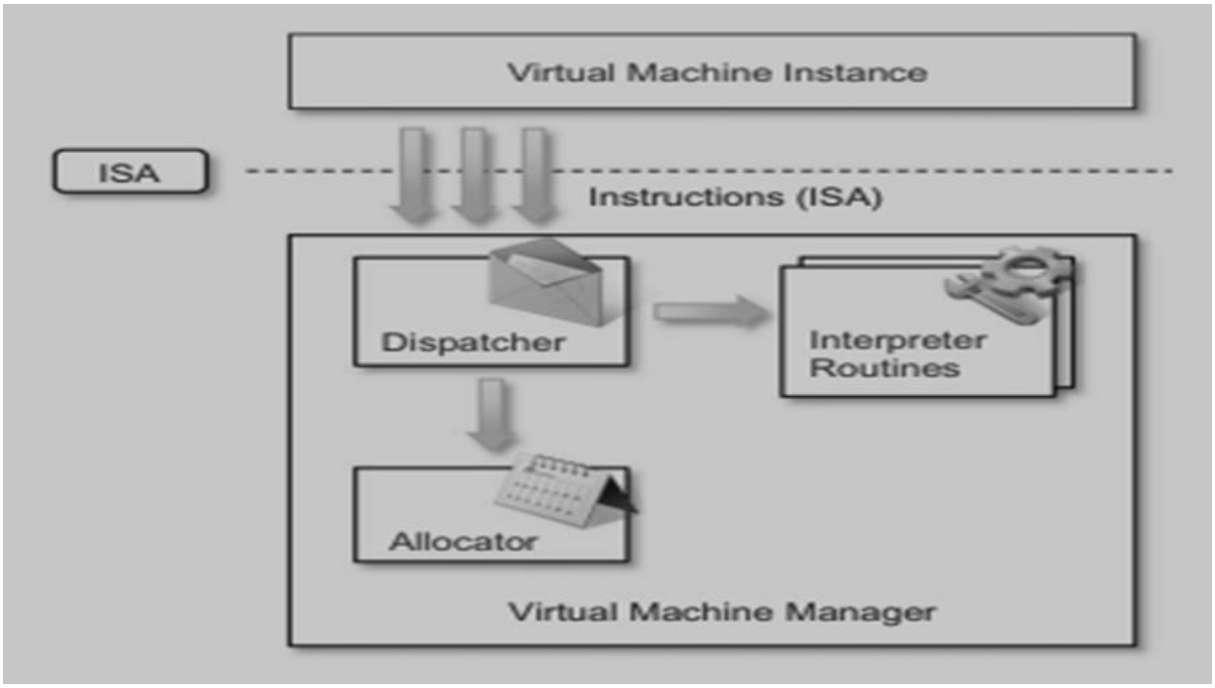
- A virtual machine manager is internally organized as described in Figure 2.8. Three main modules, dispatcher, allocator, and interpreter, coordinate their activity in order to emulate the underlying hardware. The dispatcher constitutes the entry point of the monitor and reroutes the instructions issued by the virtual machine instance to one of the two other modules.
- The allocator is responsible for deciding the system resources to be provided to the VM: whenever a virtual machine tries to execute an instruction that results in changing the machine resources associated with that VM, the allocator is invoked by the dispatcher. The interpreter module consists of interpreter routines. These are executed whenever a virtual machine executes a privileged instruction: a trap is triggered and the corresponding routine is executed.

Virtual Machine Manager: Three properties of Virtual Machine Manager that have to be satisfied:

• **Equivalence:** A guest running under the control of a virtual machine manager should exhibit the same behavior as when it is executed directly on the physical host.

• **Resource control:** The virtual machine manager should be in complete control of virtualized resources.

• **Efficiency:** A statistically dominant fraction of the machine instructions should be executed without intervention from the virtual machine manager. Popek and Goldberg provided a classification of the instruction set and proposed three theorems that define the properties that hardware instructions need to satisfy in order to efficiently support virtualization



9. Explain Theorem I, Theorem II and Theorem III (6M,Mar2022)

Solution:

THEOREM 1: For any conventional third-generation computer, a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

This theorem establishes that all the instructions that change the configuration of the system resources should generate a trap in user mode and be executed under the control of the virtual machine manager. This allows hypervisors to efficiently control only those instructions that would reveal the presence of an abstraction layer while executing all the rest of the instructions without considerable performance loss.

THEOREM 2: A conventional third-generation computer is recursively virtualizable if:

• It is virtualizable and

• A VMM without any timing dependencies can be constructed for it. Recursive virtualization is the ability to run a virtual machine manager on top of another virtual machine manager. This allows nesting hypervisors as long as the capacity of the underlying resources can accommodate that. Virtualizable hardware is a prerequisite to recursive virtualization.

THEOREM 3: A hybrid VMM may be constructed for any conventional third-generation machine in which the set of user-sensitive instructions is a subset of the set of privileged instructions. There is another term, hybrid virtual machine (HVM), which is less efficient than the virtual machine system.

In the case of an HVM, more instructions are interpreted rather than being executed directly. All instructions in virtual supervisor mode are interpreted. Whenever there is an attempt to execute a behavior-sensitive or control-sensitive instruction, HVM controls the execution directly or gains the control via a trap.

10. What are Hardware virtualization techniques (8M,Jan-2023)

Solution:

Hardware-assisted virtualization:

This term refers to a scenario in which the hardware provides architectural support for building a virtual machine manager able to run a guest operating system in complete isolation. This technique was originally introduced in the IBM System/370. At present, examples of hardware-assisted virtualization are the extensions to the x86-64 bit architecture introduced with Intel VT (formerly known as Vanderpool) and AMD V (formerly known as Pacifica).

Products such as VMware Virtual Platform, introduced in 1999 by VMware, which pioneered the field of x86 virtualization, were based on this technique. After 2006, Intel and AMD introduced processor extensions, and a wide range of virtualization solutions took advantage of them: Kernel-based Virtual Machine (KVM), VirtualBox, Xen, VMware, Hyper-V, Sun xVM, Parallels, and others.

Full virtualization:

Full virtualization refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and without any modification, as though it were run on the raw hardware. To make this possible, virtual machine manager are required to provide a complete emulation of the entire underlying hardware.

The principal advantage of full virtualization is complete isolation, which leads to enhanced security, ease of emulation of different architectures, and coexistence of different systems on the same platform. A simple solution to achieve full virtualization is to provide a virtual environment for all the instructions, thus posing some limits on performance.

Paravirtualization:

This is a not-transparent virtualization solution that allows implementing thin virtual machine managers. Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified.

The aim of paravirtualization is to provide the capability to demand the execution of performance-critical operations directly on the host, thus preventing performance losses that would otherwise be experienced in managed execution. This technique has been successfully used by Xen for providing virtualization solutions for Linux-based operating systems specifically ported to run on Xen hypervisors.

Partial virtualization:

Partial virtualization provides a partial emulation of the underlying hardware, thus not allowing the complete execution of the guest operating system in complete isolation.

Partial virtualization allows many applications to run transparently, but not all the features of the operating system can be supported, as happens with full virtualization. Partial virtualization was implemented on the experimental IBM M44/44X. Address space virtualization is a common feature of contemporary operating systems.

11. Explain Pros and Cons of Virtualization. (8M ,Jan-2023, July-2022, Feb-2022)
Solution:
Advantages of virtualization:
Managed execution and isolation are perhaps the most important advantages of virtualization. In the case of techniques supporting the creation of virtualized execution environments, these two characteristics allow building secure and controllable computing environments. A virtual execution environment can be configured as a sandbox, thus preventing any harmful operation to cross the borders of the virtual host. Moreover, allocation of resources and their partitioning among different guests is simplified, being the virtual host controlled by a program. This enables fine-tuning of resources, which is very important in a server consolidation scenario and is also a requirement for effective quality of service.

Portability and self-containment also contribute to reducing the costs of maintenance, since the number of hosts is expected to be lower than the number of virtual machine instances. By means of virtualization it is possible to achieve a more efficient use of resources. Multiple systems can securely coexist and share the resources of the underlying host, without interfering with each other.

Performance degradation: Performance is definitely one of the major concerns in using virtualization technology. Since virtualization interposes an abstraction layer between the guest and the host, the guest can experience increased latencies.

For instance, in the case of hardware virtualization, where the intermediate emulates a bare machine on top of which an entire system can be installed, the causes of performance degradation can be traced back to the overhead introduced by the following activities:
• Maintaining the status of virtual processors
• Support of privileged instructions (trap and simulate privileged instructions)
• Support of paging within VM
• Console functions

Furthermore, when hardware virtualization is realized through a program that is installed or executed on top of the host operating systems, a major source of performance degradation is represented by the fact that the virtual machine manager is executed and scheduled together with other applications, thus sharing with them the resources of the host.

- Similar consideration can be made in the case of virtualization technologies at higher levels, such as in the case of programming language virtual machines (Java, .NET, and others). Binary translation and interpretation can slow down the execution of managed applications. Moreover, because their execution is filtered by the runtime environment, access to memory and other physical resources can represent sources of performance degradation.

- These concerns are becoming less and less important thanks to technology advancements and the ever-increasing computational power available today. For example, specific techniques for hardware virtualization such as paravirtualization can increase the performance of the guest program by offloading most of its execution to the host without any change. In programming level virtual machines such as the JVM or .NET, compilation to native code is offered as an option when performance is a serious concern.

Disadvantages:

Inefficiency and degraded user experience: Virtualization can sometime lead to an inefficient use of the host. In particular, some of the specific features of the host cannot be exposed by the abstraction layer and then become inaccessible. In the case of hardware virtualization, this could happen for device drivers: The virtual machine can sometime simply provide a default graphic card that maps only a subset of the features available in the host. In the case of programming-level virtual machines, some of the features of the underlying operating systems may become inaccessible unless specific libraries are used.

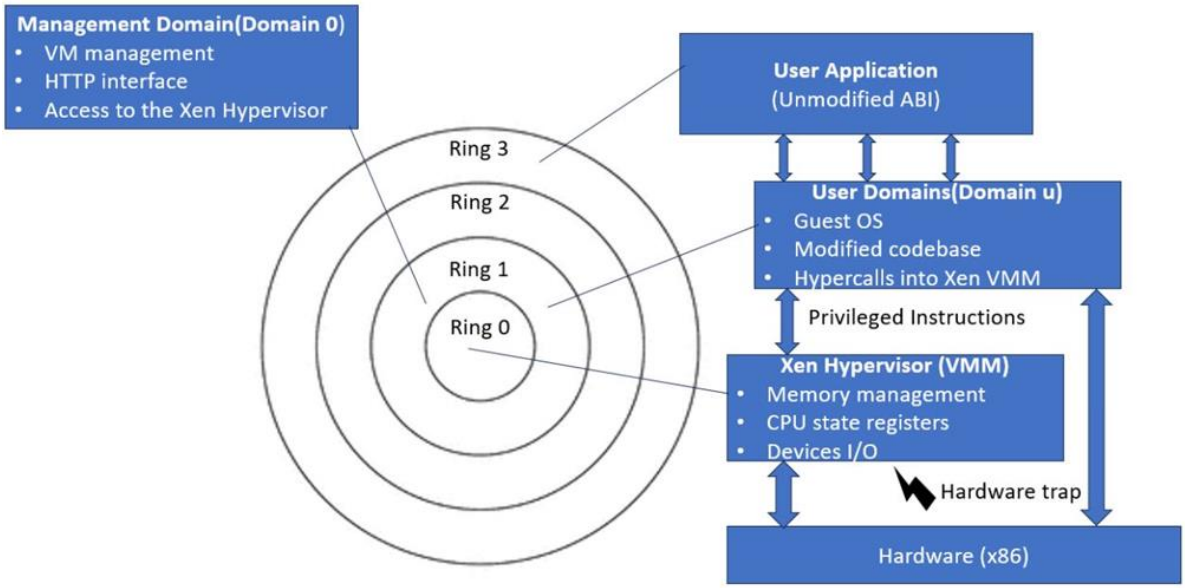
Security holes and new threats: Virtualization opens the door to a new and unexpected form of phishing. The capability of emulating a host in a completely transparent manner led the way to malicious programs that are designed to extract sensitive information from the guest. In the case of hardware virtualization, malicious programs can preload themselves before the operating system and act as a thin virtual machine manager toward it. The operating system is then controlled and can be manipulated to extract sensitive information of interest to third parties.

12. Explain Xen Paravirtualization (8M,Dec 2018)

Solution:

Xen paravirtualization: Xen is an open-source initiative implementing a virtualization platform based on paravirtualization. Initially developed by a group of researchers at the University of Cambridge in the United Kingdom, Xen now has a large open-source community backing it. Xen-based technology is used for either desktop virtualization or server virtualization, and recently it has also been used to provide cloud computing solutions by means of Xen Cloud Platform (XCP).

- Figure describes the architecture of Xen and its mapping onto a classic x86 privilege model. A Xen-based system is managed by the Xen hypervisor, which runs in the highest privileged mode and controls the access of the guest operating system to the underlying hardware.



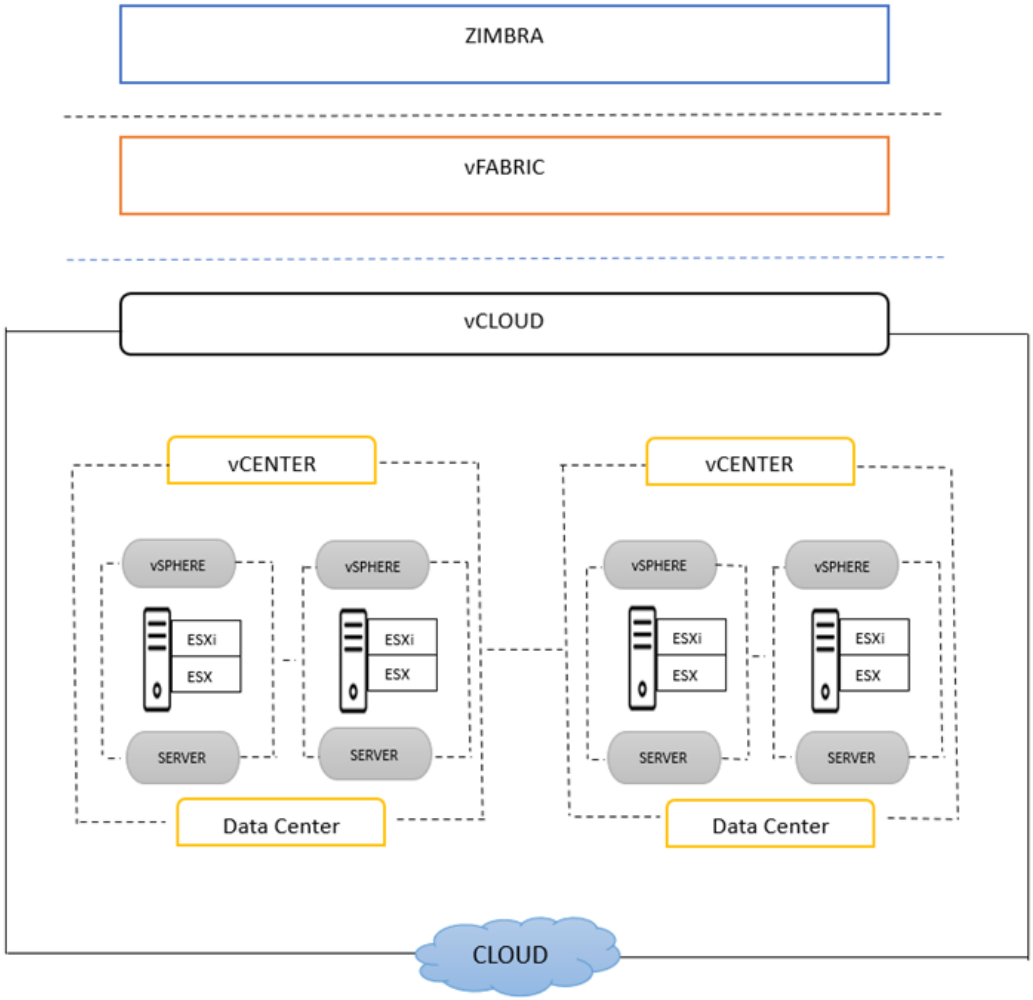
Guest operating systems are executed within domains, which represent virtual machine instances. Moreover, specific control software, which has privileged access to the host and controls all the other guest operating systems, is executed in a special domain called Domain 0. This is the first one that is loaded once the virtual machine manager has completely booted, and it hosts a HyperText Transfer Protocol (HTTP) server that serves requests for virtual machine creation, con- figuration, and termination. This component constitutes the embryonic version of a distributed virtual machine manager, which is an essential component of cloud computing systems providing Infrastructure-as-a-Service (IaaS) solutions.

- Many of the x86 implementations support four different security levels, called rings, where Ring 0 represent the level with the highest privileges and Ring 3 the level with the lowest ones. Because of the structure of the x86 instruction set, some instructions allow code executing in Ring 3 to jump into Ring 0 (kernel mode). Such operation is performed at the hardware level and therefore within a virtualized environment will result in a trap or silent fault, thus preventing the normal operations of the guest operating system, since this is now running in Ring 1. This condition is generally triggered by a subset of the system calls. To avoid this situation, operating systems need to be changed in their implementation, and the sensitive system calls need to be reimplemented with hypercalls, which are specific calls exposed by the virtual machine interface of Xen. With the use of hypercalls, the Xen hypervisor is able to catch the execution of all the sensitive instructions, manage them, and return the control to the guest operating system by means of a supplied handler.
- Paravirtualization needs the operating system codebase to be modified, and hence not all operating systems can be used as guests in a Xen-based environment. Open-source operating systems such as Linux can be easily modified, since their code is publicly available and Xen provides full support for their virtualization, whereas components of the Windows family are generally not supported by Xen unless hardware-assisted virtualization is available.

13. With a neat diagram Explain VMWare Solution stack. (10M, Mar 2022)

Solution:

VMware: full virtualization: VMware's technology is based on the concept of full virtualization, where the underlying hardware is replicated and made available to the guest operating system, which runs unaware of such abstrac- tion layers and does not need to be modified.



- VMware implements full virtualization either in the desktop environment, by means of Type II hypervisors, or in the server environment, by means of Type I hypervisors. In both cases, full virtualization is made possible by means of direct execution (for non-sensitive instructions) and binary translation (for sensitive instructions), thus allowing the virtualization of architecture such as x86.

- VMware provides a set of products covering the entire stack of cloud computing, from infrastructure management to Software-as-a-Service solutions hosted in the cloud. Figure 3.16 gives an overview of the different solutions offered and how they relate to each other. ESX and ESXi constitute the building blocks of the solution for virtual infrastructure management: A pool of virtualized servers is tied together and remotely managed as a whole by VMware vSphere.
- As a virtualization platform it provides a set of basic services besides virtual compute services: Virtual file system, virtual storage, and virtual network constitute the core of the infrastructure;

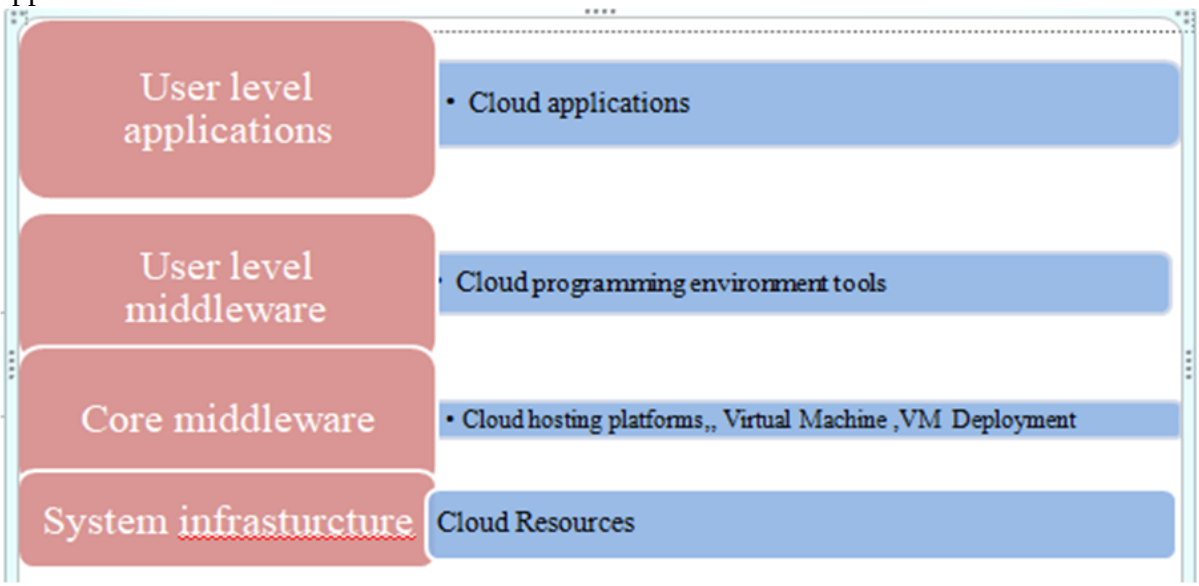
application services, such as virtual machine migration, storage migration, data recovery, and security zones, complete the services offered by vSphere. The management of the infrastructure is operated by VMware vCenter, which provides centralized administration and management of vSphere installations in a data center environment.

- A collection of virtualized data centers is turned into an Infrastructure-as-a-Service cloud by VMware vCloud, which allows service providers to make available to end users virtual computing environments on demand on a pay-per-use basis. A Web portal provides access to the provisioning services of vCloud, and end users can self provision virtual machines by choosing from available templates and setting up virtual networks among virtual instances. VMware also provides a solution for application development in the cloud with VMware vFabric, which is a set of components that facilitate the development of scalable Web applications on top of a virtualized infrastructure. vFabric is a collection of components for application monitoring, scalable data management, and scalable execution and provisioning of Java Web applications.
- Finally, at the top of the cloud computing stack, VMware provides Zimbra, a solution for office automation, messaging, and collaboration that is completely hosted in the cloud and accessible from anywhere. This is a SaaS solution that integrates various features into a single software platform providing email and collaboration management.

MODULE 2

1. What is Cloud Computing? Explain the architecture of Cloud Computing? 10M (FEB 2023, DEC 2018)

Solution: 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.



It is possible to organize all the concrete realizations of cloud computing into a layered view covering the entire stack (see Figure 3.1), from hardware appliances to software systems. Cloud resources are harnessed to offer "computing horsepower" required for providing services.

Cloud infrastructure can be heterogeneous in nature because a variety of resources, such as clusters and even networked PCs, can be used to build it. Moreover, database systems and other storage services can also be part of the infrastructure.

The physical infrastructure is managed by the core middleware, the objectives of which are to provide an appropriate runtime environment for applications and to best utilize resources. At the bottom of the stack, virtualization technologies are used to guarantee runtime environment customization, application isolation, sandboxing, and quality of service.

- Hardware virtualization is most commonly used at this level. Hypervisors manage the pool of resources and expose the distributed infrastructure as a collection of virtual machines.
- Infrastructure management is the key function of core middleware, which supports capabilities such as negotiation of the quality of service, admission control, execution management and monitoring, accounting, and billing. The combination of cloud hosting platforms and resources is generally classified as a Infrastructure-as-a-Service (IaaS) solution. The IaaS has two categories: Some of them provide both the management layer and the physical infrastructure; others provide only the management layer (IaaS (M)). In this second case, the management layer is often integrated with other IaaS solutions that provide physical infrastructure and adds value to them. IaaS solutions are suitable for designing the system infrastructure but provide limited services to build applications.

- PaaS solutions generally include the infrastructure as well, which is bundled as part of the service provided to users. In the case of Pure PaaS, only the user-level middleware is offered, and it has to be complemented with a virtual or physical infrastructure. The top layer of the reference model depicted in Figure contains services delivered at the application level. These are mostly referred to as Software-as-a-Service (SaaS).

Table summarizes the characteristics of the three major categories used to classify cloud computing solutions.

Table 4.1 Cloud Computing Services Classification

Category	Characteristics	Product Type	Vendors and Products
SaaS	Customers are provided with applications that are accessible anytime and from anywhere.	Web applications and services (Web 2.0)	SalesForce.com (CRM) Clarizen.com (project management) Google Apps
PaaS	Customers are provided with a platform for developing applications hosted in the cloud.	Programming APIs and frameworks Deployment systems	Google AppEngine Microsoft Azure Manjrasoft Aneka Data Synapse
IaaS/HaaS	Customers are provided with virtualized hardware and storage on top of which they can build their infrastructure.	Virtual machine management infrastructure Storage management Network management	Amazon EC2 and S3 GoGrid Nirvanix

2.Explain the following with a neat diagram.

i. **Infrastructure and hardware-as-a-service (Iaas)**
ii. **Platform as a Service (Paas)**
iii. **Software as a service(Saas)** **(10M,DEC 2019,JULY2022,FEB 2023)**

Solution:

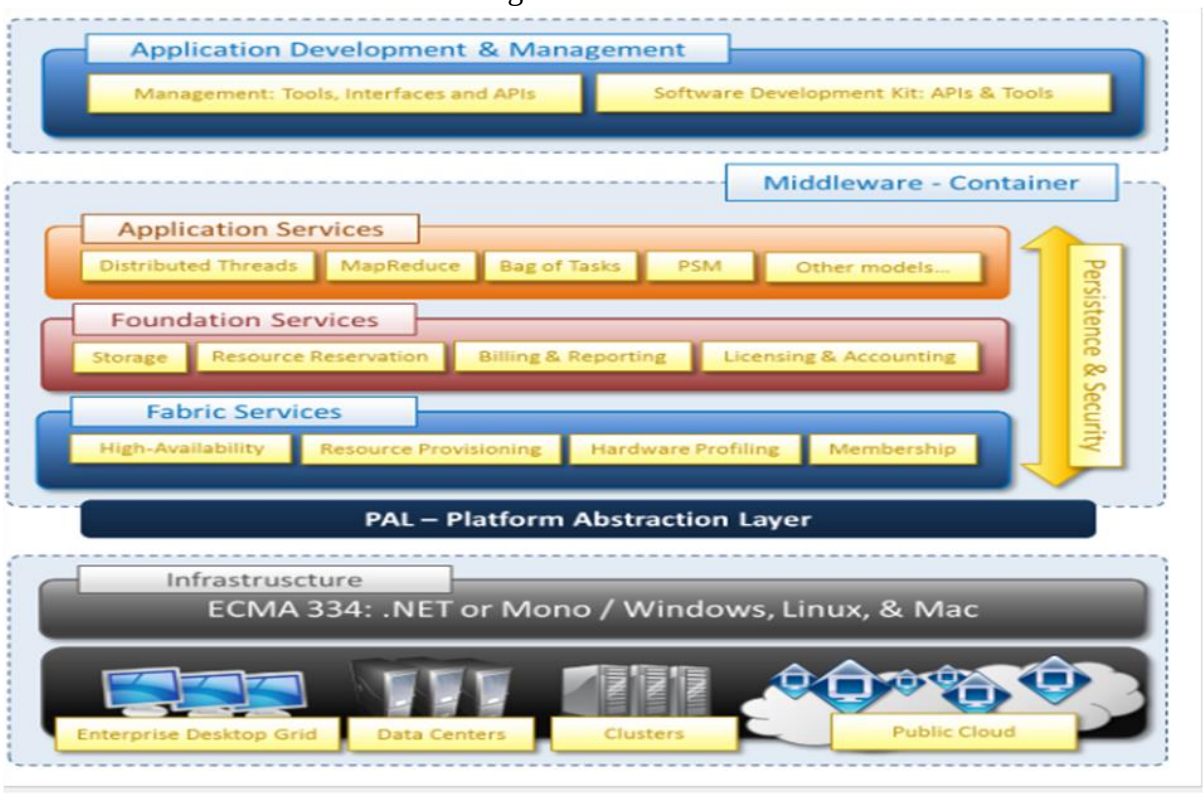
Infrastructure and hardware-as-a-service Infrastructure and Hardware-as-a-Service (IaaS/HaaS) solutions are the most popular and developed market segment of cloud computing. They deliver customizable infrastructure on demand. The available options within the IaaS offering umbrella range

from single servers to entire infrastructures, including network devices, load balancers, and database and Web servers.

The main technology used to deliver and implement these solutions is hardware virtualization: one or more virtual machines opportunely configured and interconnected define the distributed system on top of which applications are installed and deployed. IaaS/HaaS solutions bring all the benefits of hardware virtualization: workload partitioning, application isolation, sandboxing, and hard- ware tuning. From the perspective of the service provider, IaaS/HaaS allows better exploiting the IT infrastructure and provides a more secure environment where executing third party applications. From the perspective of the customer, it reduces the administration and maintenance cost as well as the capital costs allocated to purchase hardware.

The below Figure provides an overall view of the components forming an Infrastructure-as-a-Service solution. It is possible to distinguish three principal layers: the physical infrastructure, the software management infrastructure, and the user interface. At the top layer the user interface provides access to the services exposed by the software management infrastructure.
The core features of an IaaS solution are implemented in the infrastructure management software layer. In particular, management of the virtual machines is the most important function performed by this layer. A central role is played by the scheduler, which is in charge of allocating the execution of virtual machine instances. The scheduler interacts with the other components that perform a variety of tasks:

• The **pricing and billing** component takes care of the cost of executing each virtual machine instance and maintains data that will be used to charge the user.



- The **monitoring** component tracks the execution of each virtual machine instance and maintains data required for reporting and analyzing the performance of the system.
- • The **reservation** component stores the information of all the virtual machine instances that have been executed or that will be executed in the future.
- • If support for QoS-based execution is provided, a **QoS/SLA management** component will maintain a repository of all the SLAs made with the users; together with the monitoring component, this component is used to ensure that a given virtual machine instance is executed with the desired quality of service.
- The **VM repository** component provides a catalog of virtual machine images that users can use to create virtual instances. Some implementations also allow users to upload their specific virtual machine images.
- A **VM pool manager** component is responsible for keeping track of all the live instances.
- A **provisioning** component interacts with the scheduler to provide a virtual machine instance that is external to the local physical infrastructure directly managed by the pool. The bottom layer is composed of the physical infrastructure, on top of which the management layer operates. From an architectural point of view, the physical layer also includes the virtual resources that are rented from external IaaS providers. In the case of complete IaaS solutions, all three levels are offered as service.

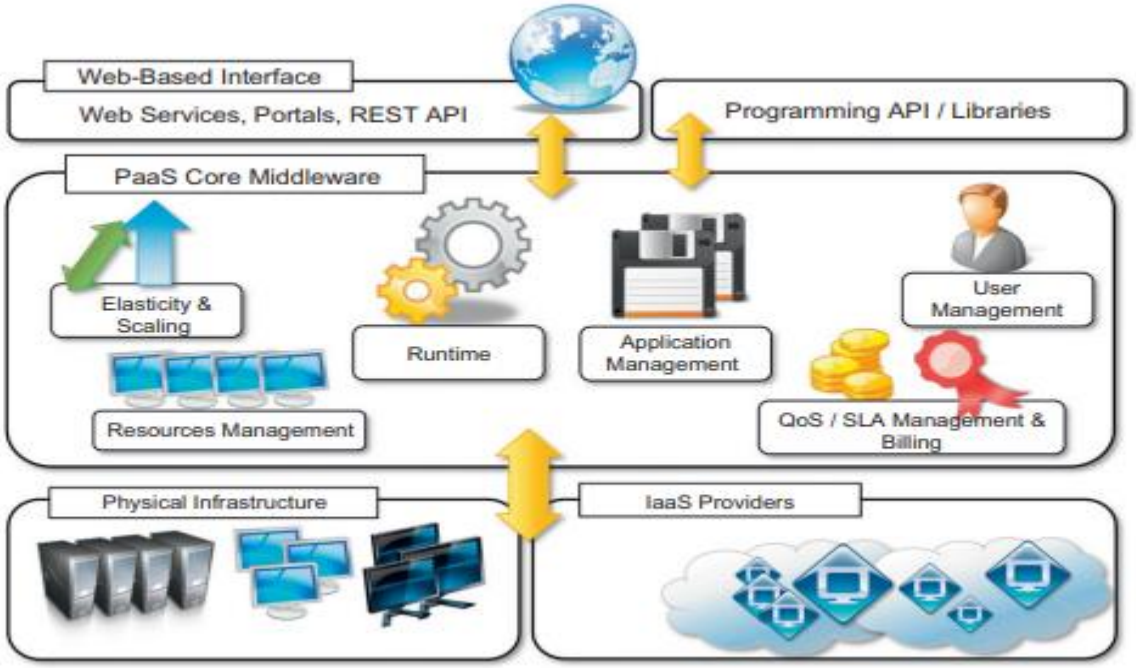
Platform as a Service

- Platform-as-a-Service (PaaS) solutions provide a development and deployment platform for running applications in the cloud. They constitute the middleware on top of which applications are built. A general overview of the features characterizing the PaaS approach is given in figure.

- Application management is the core functionality of the middleware. PaaS implementations provide applications with a runtime environment and do not expose 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 in charge of managing the resources and scaling applications on



demand or automatically, according to the commitments made with users. From a user point of view, the core middleware exposes interfaces that allow programming and deploying applications on the cloud. These can be in the form of a Web-based interface or in the form of programming APIs and libraries.

- Some implementations provide a completely Web-based interface hosted in the cloud and offering a variety of services. Other implementations of the PaaS model provide a complete object model for representing an application and provide a programming language-based approach.

- PaaS solutions can offer middleware for developing applications together with the infrastructure or simply provide users with the software that is installed on the user premises.

- Table provides a classification of the most popular PaaS implementations. It is possible to organize the various solutions into three wide categories: PaaS-I, PaaS-II, and PaaS-III.
- The first category identifies PaaS implementations that completely follow the cloud computing style for application development and deployment. They offer an integrated development environment hosted within the Web browser where applications are designed, developed, composed, and deployed. The second class gives solutions that are focused on providing a scalable infrastructure for Web applications, mostly websites. In this case, developers generally use the providers' APIs, which are built on top of industrial runtimes, to develop applications. The third category consists of all those solutions that provide a cloud programming platform for any kind of application, not only Web applications.

The essential characteristics that identify a PaaS solution:

- **Runtime framework.** The runtime framework executes end-user code according to the policies set by the user and the provider.
- **Abstraction.** PaaS solutions are distinguished by the higher level of abstraction that they provide. This means that PaaS solutions offer a way to deploy and manage applications on the cloud.
- **Automation.** PaaS environments automate the process of deploying applications to the infrastructure, scaling them by provisioning additional resources when needed. This process is performed automatically and according to the SLA made between the customers and the provider.
- **Cloud services.** PaaS offerings provide developers and architects with services and APIs, helping them to simplify the creation and delivery of elastic and highly available cloud applications.

- Another essential component for a PaaS-based approach is the ability to integrate third-party cloud services offered from other vendors by leveraging service-oriented architecture. One of the major concerns of leveraging PaaS solutions for implementing applications is vendor lock-in.
- PaaS solutions can cut the cost across development, deployment, and management of applications. It helps management reduce the risk of ever-changing technologies by offloading the cost of upgrading the technology to the PaaS provider. The PaaS approach, when bundled with underlying IaaS solutions, helps even small start-up companies quickly offer customers integrated solutions on a hosted platform at a very minimal cost.

Software as a service

- Software-as-a-Service (SaaS) is a software delivery model that provides access to applications through the Internet as a Web-based service. It provides a means to 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. In this scenario, customers neither need to install anything on their premises nor have to pay considerable up-front costs to purchase the software and the required licenses. On the provider side, the specific details and features of each customer's application are maintained in the infrastructure and made available on demand.

- SaaS applications are naturally multitenant. Multitenancy, which is a feature of SaaS compared to traditional packaged software, allows providers to centralize and sustain the effort of managing large hardware infrastructures, maintaining and upgrading applications transparently to the users, and optimizing resources by sharing the costs among the large user base. On the customer side, such costs constitute a minimal fraction of the usage fee paid for the software.

- In the software as a service model, the application, or service, is deployed from a centralized datacenter across a network—Internet, Intranet, LAN, or VPN—providing access and use on a recurring fee basis. Users "rent," "subscribe to," "are assigned," or "are granted access to" the applications from a central provider. Business models vary according to the level to which the software is streamlined, to lower price and increase efficiency, or value-added through customization to further improve digitized business processes.

- The analysis carried out by SIIA was mainly oriented to cover application service providers (ASPs) and all their variations, which capture the concept of software applications consumed as a service in a broader sense. ASPs already had some of the core characteristics of SaaS:
  • The product sold to customer is application access.
  • The application is centrally managed.
  • The service delivered is one-to-many.
  • The service delivered is an integrated solution delivered on the contract, which means provided as promised.

- The SaaS approach introduces a more flexible way of delivering application services that are fully customizable by the user by integrating new services, injecting their own components, and designing the application and information workflows. Such a new approach has also been possible with the support of Web 2.0 technologies, which allowed turning the Web browser into a full-featured interface, able even to support application composition and development.

- Initially the SaaS model was of interest only for lead users and early adopters. The benefits delivered at that stage were the following:
  •Software cost reduction and total cost of ownership (TCO) were paramount Service-level improvements
  •Rapid implementation
  •Standalone and configurable applications
  •Rudimentary application and data integration
  •Subscription and pay-as-you-go (PAYG) pricing
- With the advent of cloud computing there has been an increasing acceptance of SaaS as a viable software delivery model. This led to transition into SaaS 2.0, which does not introduce a new technology but transforms the way in which SaaS is used.

2. Explain the different types of Clouds. (10M , MAR 2022, FEB 2023)

Solution:

Types of clouds Clouds constitute the primary outcome of cloud computing. They are a type of parallel and distributed system harnessing physical and virtual computers presented as a unified computing resource. Clouds build the infrastructure on top of which services are implemented and delivered to customers. A more useful 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 adopted to support such services, and qualifies them. It is then possible to differentiate four different types of cloud:

• **Public clouds.** The cloud is open to the wider public.

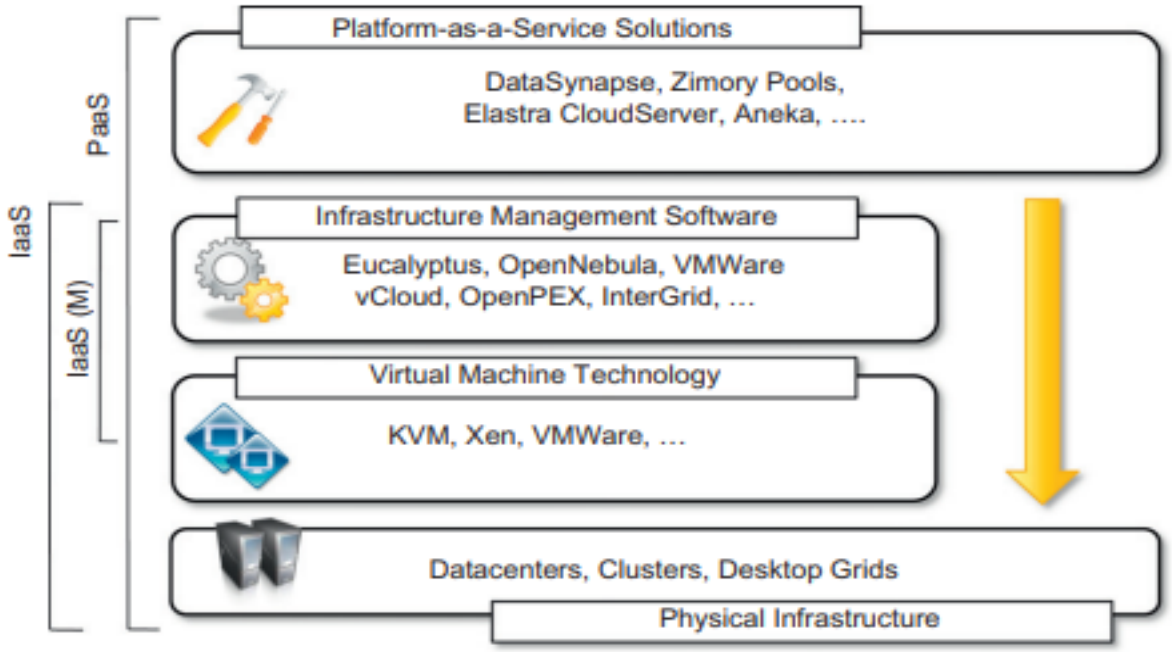
Private clouds. The cloud is implemented within the private premises of an institution and generally made accessible to the members of the institution or a subset of them.

• **Hybrid or heterogeneous clouds.** The cloud is a combination of the two previous solutions and most likely identifies a private cloud that has been augmented with resources or services hosted in a public cloud.

• **Community clouds.** The cloud is characterized by a multi-administrative domain involving different deployment models (public, private, and hybrid), and it is specifically designed to address the needs of a specific industry.

1. **Public clouds:**
   - Public clouds constitute the first expression of cloud computing. They are a realization of the canonical view of cloud computing in which the services offered are made available to anyone, from anywhere, and at any time through the Internet. 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 sign in with the cloud provider, enter their credential and billing details, and use the services offered. Public clouds are used both to completely replace the IT infrastructure of enterprises and to extend it when it is required.
   - A fundamental characteristic of public clouds is multi-tenancy. A public cloud is meant to serve a multitude of users, not a single customer. A public cloud is meant to serve a multitude of users, not a single customer. Any customer requires a virtual computing environment that is separated, and most likely isolated, from other users.
   - QoS management is a very important aspect of public clouds. significant portion of the software infrastructure is devoted to monitoring the cloud resources, to bill them according to the contract made with the user, and to keep a complete history of cloud usage for each customer.
   - A public cloud can offer any kind of service: infrastructure, platform, or applications. For example, Amazon EC2 is a public cloud that provides infrastructure as a service; Google AppEngine is a public cloud that provides an application development platform as a service; and SalesForce.com is a public cloud that provides software as a service.
   - From an architectural point of view there is no restriction concerning the type of distributed system implemented to support public clouds. Public clouds can be composed of geographically dispersed data centers to share the load of users and better serve them according to their locations

     Private clouds:
   - Private clouds are virtual distributed systems that rely on a private infrastructure and provide internal users with dynamic provisioning of computing resources. Instead of a pay-as-you-go model as in public clouds, there could be other schemes in place, taking into account the usage of the cloud and proportionally billing the different departments or sections of an enterprise.
   - Private clouds have the advantage of keeping the core business operations in-house by relying on the existing IT infrastructure and reducing the burden of maintaining it once the cloud has been set up.
   - In this scenario, security concerns are less critical, since sensitive information does not flow out of the private infrastructure. Existing IT resources can be better utilized because the private cloud can provide services to a different range of users. Another interesting opportunity that comes with private clouds is the possibility of testing applications and systems at a comparatively lower price rather than public clouds before deploying them on the public virtual infrastructure.



Private clouds hardware and software stack

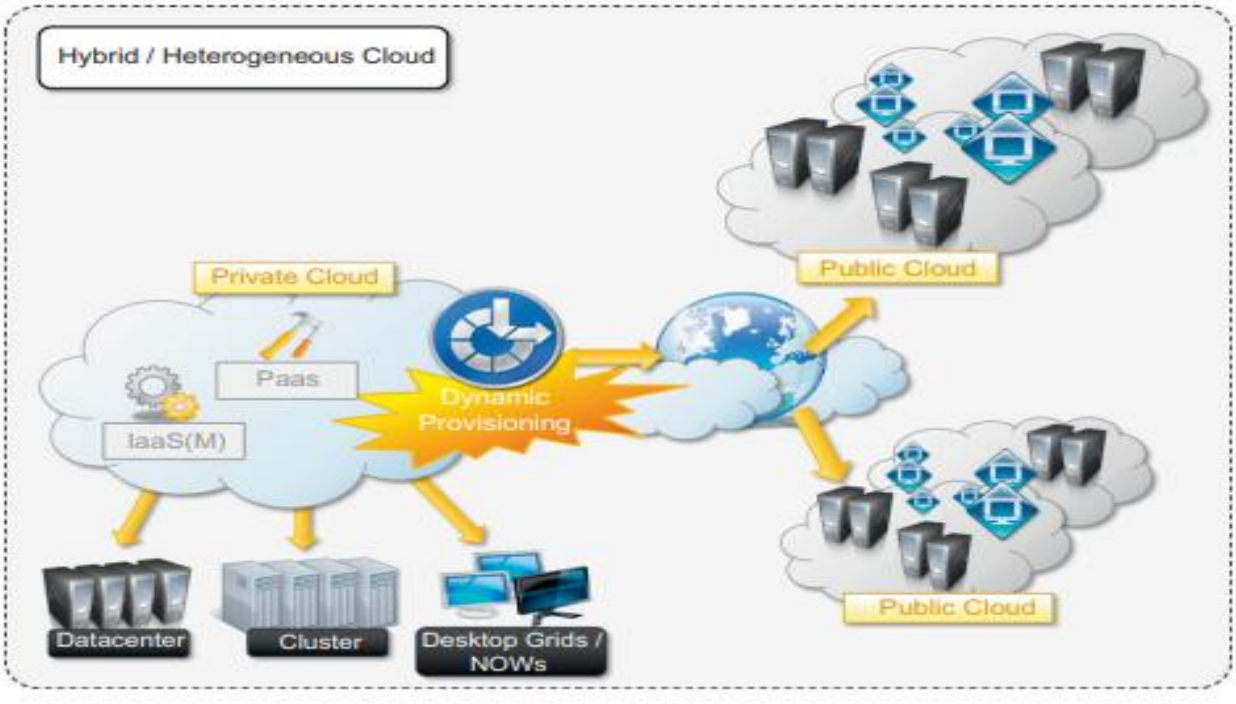
The **key advantages** of using a private cloud computing infrastructure:

• **Customer information protection.** Despite assurances by the public cloud leaders about security, few provide satisfactory disclosure or have long enough histories with their cloud offerings to provide warranties about the specific level of security put in place on their systems. In-house security is easier to maintain and rely on.

• **Infrastructure ensuring SLAs.** Quality of service implies specific operations such as appropriate clustering and failover, data replication, system monitoring and maintenance, and disaster recovery, and other uptime services can be commensurate to the application needs.

• **Compliance with standard procedures and operations**. If organizations are subject to thirdparty compliance standards, specific procedures have to be put in place when deploying and executing applications.

2. **Hybrid Cloud:**
   - Hybrid clouds allow enterprises to exploit existing IT infrastructures, maintain sensitive information within the premises, and naturally grow and shrink by provisioning external resources and releasing them when they're no longer needed. Security concerns are then only limited to the public portion of the cloud that can be used to perform operations with less stringent constraints but that are still part of the system workload.
   - Figure provides a general overview of a hybrid cloud: It is a heterogeneous distributed system



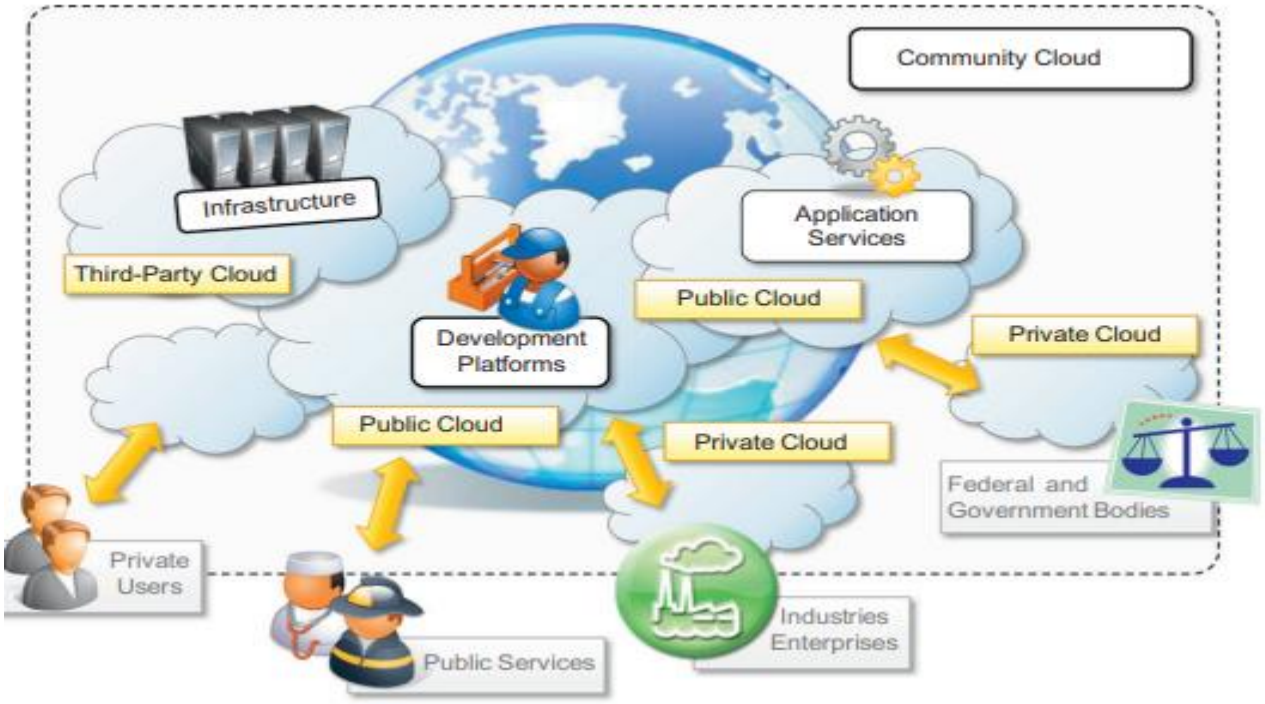
resulting from a private cloud that integrates additional services or resources from one or more public clouds. For this reason, they are also called heterogeneous clouds.

   - As depicted in the diagram, dynamic provisioning is a fundamental component in this scenario. Hybrid clouds address scalability issues by leveraging external resources for exceeding capacity demand. These resources or services are temporarily leased for the time required and then released. This practice is also known as cloud bursting.
   - Whereas the concept of hybrid cloud is general, it mostly applies to IT infrastructure rather than software services. Service-oriented computing already introduces the concept of integration of paid software services with existing application deployed in the private premises.

3. **Community clouds:**
   - Community clouds are distributed systems created by integrating the services of different clouds to address the specific needs of an industry, a community, or a business sector. The

National Institute of Standards and Technologies (NIST) characterize community clouds as follows:

- The infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations).
- Figure below provides a general view of the usage scenario of community clouds, together



with reference architecture. The users of a specific community cloud fall into a well-identified community, sharing the same concerns or needs; they can be government bodies, industries, or even simple users, but all of them focus on the same issues for their interaction with the cloud.

- Community clouds are also different from private clouds, where the services are generally delivered within the institution that owns the cloud.
- From an architectural point of view, a community cloud is most likely implemented over multiple administrative domains.

Candidate sectors for community clouds are as follows:

• **Media industry.** In the media industry, companies are looking for low-cost, agile, and simple solutions to improve the efficiency of content production. Most media productions involve an extended ecosystem of partners. In particular, the creation of digital content is the outcome of a collaborative process that includes movement of large data, massive compute-intensive rendering tasks, and complex workflow executions.

• **Healthcare industry.** In the healthcare industry, there are different scenarios in which community clouds could be of use. In particular, community clouds can provide a global platform on which to share information and knowledge without revealing sensitive data maintained within the private infrastructure.

• **Energy and other core industries.** In these sectors, community clouds can bundle the comprehensive set of solutions that together vertically address management, deployment, and orchestration of services and operations.
• **Public sector.** Legal and political restrictions in the public sector can limit the adoption of public cloud offerings. Moreover, governmental processes involve several institutions and agencies and are aimed at providing strategic solutions at local, national, and international administrative levels. They involve business-to-administration, citizen-to-administration, and possibly business-to-business processes.
• **Scientific research.** Science clouds are an interesting example of community clouds. In this case, the common interest driving different organizations sharing a large distributed infrastructure is scientific computing.

The **benefits of these community clouds** are the following:
• **Openness.** By removing the dependency on cloud vendors, community clouds are open systems in which fair competition between different solutions can happen.
• **Community.** Being based on a collective that provides resources and services, the infrastructure turns out to be more scalable because the system can grow simply by expanding its user base.
• **Graceful failures.** Since there is no single provider or vendor in control of the infrastructure, there is no single point of failure.
• **Convenience and control.** Within a community cloud there is no conflict between convenience and control because the cloud is shared and owned by the community, which makes all the decisions through a collective democratic process.
• **Environmental sustainability.** The community cloud is supposed to have a smaller carbon footprint because it harnesses underutilized resources. Moreover, these clouds tend to be more organic by growing and shrinking in a symbiotic relationship to support the demand of the community, which in turn sustains it.

3. Describe the fundamental features of the economic and business model behind Cloud Computing. (8M, DEC 2018)

Solution:

Economics of the cloud

The main drivers of cloud computing are economy of scale and simplicity of software delivery and its operation. In fact, the biggest benefit of this phenomenon is financial: the pay-as-you-go model offered by cloud providers. In particular, cloud computing allows:
• Reducing the capital costs associated to the IT infrastructure
• Eliminating the depreciation or lifetime costs associated with IT capital assets
• Replacing software licensing with subscriptions
• Cutting the maintenance and administrative costs of IT resources

- A capital cost is the cost occurred in purchasing an asset that is useful in the production of goods or the rendering of services. Capital costs are one-time expenses that are generally paid up front and that will contribute over the long term to generate profit.
- IT resources constitute a capital cost for any kind of enterprise. It is good practice to try to keep capital costs low because they introduce expenses that will generate profit over time;

more than that, since they are associated with material things they are subject to depreciation over time.

- These costs can be better controlled according to the business needs and prosperity of the enterprise. Cloud computing also introduces reductions in administrative and maintenance costs. That is, there is no or limited need for having administrative staff take care of the management of the cloud infrastructure. At the same time, the cost of IT support staff is also reduced. When it comes to depreciation costs, they simply disappear for the enterprise, since in a scenario where all the IT needs are served by the cloud there are no IT capital assets that depreciate over time.
- The amount of cost savings that cloud computing can introduce within an enterprise is related to the specific scenario in which cloud services are used and how they contribute to generate a profit for the enterprise.

• **Tiered pricing.** In this model, cloud services are offered in several tiers, each of which offers a fixed computing specification and SLA at a specific price per unit of time.

• **Per-unit pricing.** This model is more suitable to cases where the principal source of revenue for the cloud provider is determined in terms of units of specific services, such as data transfer and memory allocation.

• **Subscription-based pricing.** This is the model used mostly by SaaS providers in which users pay a periodic subscription fee for use of the software or the specific component services that are integrated in their applications.

4. Write a note on open challenges in cloud computing in industries.

(6M, FEB 2023, JULY 2022)

Solution:

Cloud computing presents many challenges for industry

Cloud interoperability and standards: To fully realize this goal, introducing standards and allowing interoperability between solutions offered by different vendors are objectives of fundamental importance.

Vendor lock-in constitutes one of the major strategic barriers against the seam- less adoption of cloud computing at all stages. Vendor lock-in can prevent a customer from switching to another competitor's solution, or when this is possible, it happens at considerable conversion cost and requires significant amounts of time.

Scalability and fault tolerance: Clouds allow scaling beyond the limits of the existing in-house IT resources. To implement such a capability, the cloud middleware has to be designed with the principle of scalability along different dimensions in mind—for example, performance, size, and load. the ability to tolerate failure becomes fundamental, sometimes even more important than providing an extremely efficient and optimized system. Hence, the challenge in this case is designing highly scalable and fault-tolerant systems.

Security, trust, and privacy: Security, trust, and privacy issues are major obstacles for massive adoption of cloud computing. The massive use of virtualization technologies exposes the existing system to new threats, which previously were not considered applicable. For example, it might be possible that applications hosted in the cloud can process sensitive information; such information can

be stored within a cloud storage facility using the most advanced technology in cryptography to protect data and then be considered safe from any attempt to access it without the required permissions.

Organizational aspects: Cloud computing introduces a significant change in the way IT services are consumed and managed. In particular, a wide acceptance of cloud computing will require a significant change to business processes and organizational boundaries.

From an organizational point of view, the lack of control over the management of data and processes poses not only security threats but also new problems that previously did not exist. The existing IT staff is required to have a different kind of competency and, in general, fewer skills, thus reducing their value. These are the challenges from an organizational point of view that must be faced.

5. Describe Aneka Framework / Aneka Containers with the diagram.

(10M ,Dec2018, Feb 2023)

Solution:

Aneka

• Aneka is a software platform for developing cloud computing applications.

• Aneka is a pure PaaS solution for cloud computing.

• Aneka is a cloud middleware product that can be deployed on a heterogeneous set of resources: Like: a network of computers, a multi core server, data centers, virtual cloud infrastructures, or a mixture of all

• The framework provides both middleware for managing and scaling distributed applications and an extensible set of APIs for developing them

• A collection of interconnected containers constitutes the Aneka Cloud: a single domain in which services are made available to users and developers.

• The container features three different classes of services:

- o Fabric Services,
- o Foundation Services,
- o Execution Services.

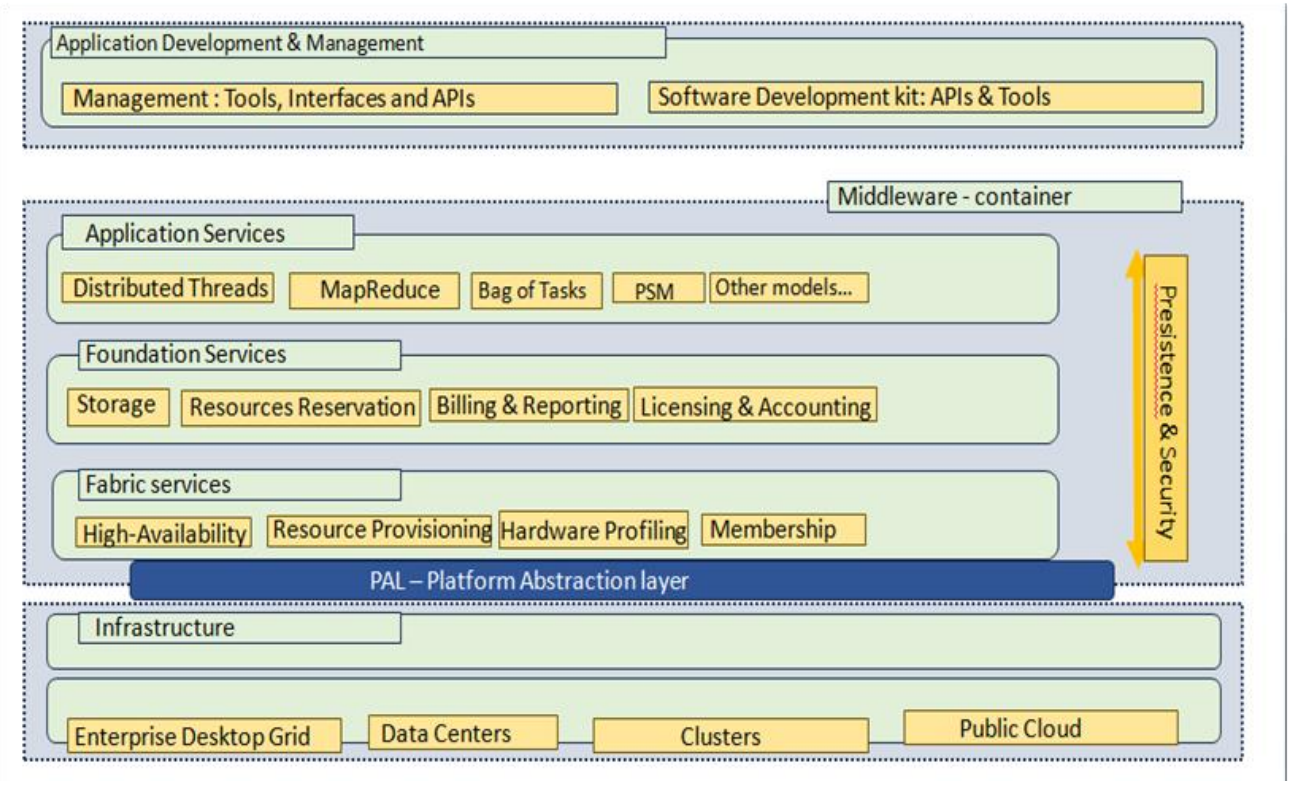
• Fabric services take care of infrastructure management for the Aneka Cloud

• Foundation Services take care of supporting services for the Aneka Cloud

• Application Services take care of application management and execution respectively

. Runtime management:

The run time machinery is responsible for keeping the infrastructure up and running and serves as a hosting environment for services. Resource management: Aneka is an elastic infrastructure in which resources are added and removed dynamically according to application needs and user requirements



Aneka Framework with Diagram

- **Application management:** A specific subset of services is devoted to managing applications. These services include schedling, execution, monitoring, and storage management. User management: Aneka is a multi-tenant distributed environment in which multiple applications, potentially belonging to different users, are executed.
- The framework provides an extensible user system via which it is possible to define users, groups, and permissions. QoS/SLA management and billing: Within a cloud environment, application execution is metered and billed. Aneka provides a collection of services that coordinate together to take into account the usage of resources by each application and to bill the owning user accordingly

Anatomy of the Aneka container

- The Aneka container constitutes the building blocks of Aneka Clouds and represents the runtime machinery available to services and applications
- The container is the unit of deployment in Aneka Clouds, and it is a lightweight software layer designed to host services and interact with the underlying operating system and hardware

The Aneka container can be classified into three major categories:
• Fabric Services
• Foundation Services
Application Services

- These services stack resides on top of the Platform Abstraction Layer (PAL) (Refer Diagram) ; it represents the interface to the underlying operating system and hardware.

- PAL provides a uniform view of the software and hardware environment in which the container is running

PAL –Platform Abstraction Layer

- In a cloud environment each operating system has a different file system organization and stores that information differently.
- It is The Platform Abstraction Layer (PAL) that addresses this heterogeneity problem with Operating systems and provides the container with a uniform interface for accessing the relevant information, thus the rest of the container can be operated without modification on any supported platform

The PAL provides the following features:
• Uniform and platform-independent implementation interface for accessing the hosting platform
• Uniform access to extended and additional properties of the hosting platform
• Uniform and platform-independent access to remote nodes
• Uniform and platform-independent management interfaces

Also, The PAL is a small layer of software that comprises a detection engine, which automatically configures the container at boot time, with the platform-specific component to access the above information and an implementation of the abstraction layer for the Windows, Linux, and Mac OS X operating systems.
Following are the collectible data that are exposed by the PAL:
• Number of cores, frequency, and CPU usage
• Memory size and usage
• Aggregate available disk space
• Network addresses and devices attached to the node

Fabric services

- FabricServices define the lowest level of the software stack representing the Aneka Container.
- They provide access to the Resource-provisioning subsystem and to the Monitoring facilities implemented in Aneka.
- Resource-provisioning services are in charge of dynamically providing new nodes on demand by relying on virtualization technologies
- Monitoring services allow for hardware profiling and implement a basic monitoring infrastructure that can be used by all the services installed in the container

The two services of Fabric class are:
o Profiling and monitoring
o Resource management

Profiling and monitoring

Profiling and monitoring services are mostly exposed through following services

- Heartbeat,
- Monitoring,
- Reporting

- The Heart Beat makes the information that is collected through the PAL available
- The Monitoring and Reporting implement a generic infrastructure for monitoring the activity of any service in the Aneka Cloud

Heartbeat Functions in detail

- The Heartbeat Service periodically collects the dynamic performance information about the node and publishes this information to the membership service in the Aneka Cloud
- It collects basic information about memory, disk space, CPU, and operating system
- These data are collected by the index node of the Cloud, and makes them available for reservations and scheduling services that optimizes them for heterogeneous infrastructure
- Heartbeat works with a specific component, called Node Resolver, which is in charge of collecting these data

Reporting & Monitoring Functions in detail

- The Reporting Service manages the store for monitored data and makes them accessible to other services for analysis purposes.
- On each node, an instance of the Monitoring Service acts as a gateway to the Reporting Service and forwards to it all the monitored data that has been collected on the node

Many Built-in services use this channel to provide information, important built-in services are:

- The Membership Catalog service tracks the performance information of nodes.
- The Execution Service monitors several time intervals for the execution of jobs.
- The Scheduling Service tracks the state transitions of jobs.
- The Storage Service monitors and obtains information about data transfer such as upload and download times, file names, and sizes.
- The Resource Provisioning Service tracks the provisioning and lifetime information of virtual nodes.

Aneka provides a collection of services that are in charge of managing resources.
These are

- Index Service (or Membership Catalogue)
- Resource Provisioning Service

Membership Catalogue features

- The Membership Catalogue is Aneka's fundamental component for resource management
- It keeps track of the basic node information for all the nodes that are connected or disconnected.
- It implements the basic services of a directory service, where services can be searched using attributes such as names and nodes

Resource Provisioning Service Features

- The resource provisioning infrastructure built into Aneka is mainly concentrated in the Resource Provisioning Service,

- It includes all the operations that are needed for provisioning virtual instances (Providing virtual instances as needed by users).
- The implementation of the service is based on the idea of resource pools.
- A resource pool abstracts the interaction with a specific IaaS provider by exposing a common interface so that all the pools can be managed uniformly.

Foundation services

Foundation Services are related to the logical management of the distributed system built on top of the infrastructure and provide supporting services for the execution of distributed applications. These services cover:

- Storage management for applications
- Accounting, billing and resource pricing
- Resource reservation

Storage management

Aneka offers two different facilities for storage management:

- A centralized file storage, which is mostly used for the execution of compute- intensive applications,
- A distributed file system, which is more suitable for the execution of data-intensive applications.
- As the requirements for the two types of applications are rather different. Compute intensive applications mostly require powerful processors and do not have high demands in terms of storage, which in many cases is used to store small files that are easily transferred from one node to another. Here, a centralized storage node is sufficient to store data
- Centralized storage is implemented through and managed by Aneka's Storage Service.
- The protocols for implementing centralized storage management are supported by a concept of File channel. It consists of a File Channel controller and File channel handler. File channel controller is a server component whereas File channel handler is a client component (that allows browsing, downloading and uploading of files).
- In contrast, data-intensive applications are characterized by large data files (gigabytes or terabytes, peta bytes) and here processing power required by tasks is not more.
- Here instead of centralized data storage a distributed file system is used for storing data by using all the nodes belonging to the cloud.
- Data intensive applications are implemented by means of a distributed file system. Google File system is best example for distributed file systems.
- Typical characteristics of Google File system are:
- Files are huge by traditional standards (multi-gigabytes).
- Files are modified by appending new data rather than rewriting existing data. o There are two kinds of major workloads: large streaming reads and small random reads.
- It is more important to have a sustained bandwidth than a low latency.

Accounting, billing, and resource pricing

- Accounting Services keep track of the status of applications in the Aneka Cloud

- The information collected for accounting is primarily related to infrastructure usage and application execution
- Billing is another important feature of accounting.
- Billing is important since Aneka is a multi tenant cloud programming platform in which the execution of applications can involve provisioning additional resources from commercial providers.
- Aneka Billing Service provides detailed information about each user's usage of resources, with the associated costs.
- Resource pricing is associated with the price fixed for different types of resources/nodes that are provided for the subscribers. Powerful resources are priced high and less featured resources are priced low
- Two internal services used by accounting and billing are Accounting service and Reporting Service

Resource reservation

Resource Reservation supports the execution of distributed applications and allows for reserving resources for exclusive use by specific applications.

Two types of services are used to build resource reservation:

- The Resource Reservation
- The Allocation Service
- Resource Reservation keeps track of all the reserved time slots in the Aneka Cloud and provides a unified view of the system. (provides overview of the system)
- The Allocation Service is installed on each node that features execution services and manages the database of information regarding the allocated slots on the local node.

Different Reservation Service Implementations supported by Aneka Cloud are:

Basic Reservation: Features the basic capability to reserve execution slots on nodes and implements the alternate offers protocol, which provides alternative options in case the initial reservation requests cannot be satisfied.

Libra Reservation: Represents a variation of the previous implementation that features the ability to price nodes differently according to their hardware capabilities.

Relay Reservation: This implementation is useful in integration scenarios in which Aneka operates in an inter cloud environment.

Application services

Application Services manage the execution of applications and constitute a layer that differentiates according to the specific programming model used for developing distributed applications on top of Aneka

Two types of services are:

1. The Scheduling Service : Scheduling Services are in charge of planning the execution of distributed applications on top of Aneka and governing the allocation of jobs composing an application to nodes. Common tasks that are performed by the scheduling component are the following:

- Job to node mapping
- Rescheduling of failed jobs
- Job status monitoring
- Application status monitoring

The various Programming Models Supported by Execution Services of Aneka Cloud are:

1. Task Model. This model provides the support for the independent "bag of tasks" applications and many computing tasks. In this model application is modeled as a collection of tasks that are independent from each other and whose execution can be sequenced in any order

2. Thread Model. This model provides an extension to the classical multithreaded programming to a distributed infrastructure and uses the abstraction of Thread to wrap a method that is executed remotely.

3. Map Reduce Model. This is an implementation of Map Reduce as proposed by Google on top of Aneka.

4. Parameter Sweep Model. This model is a specialized form of Task Model for applications that can be described by a template task whose instances are created by generating different combinations of parameters.

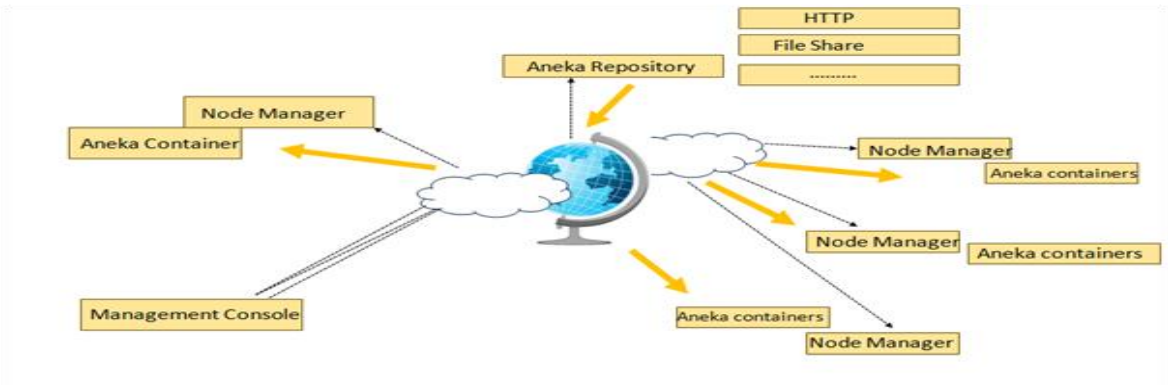
6. Describe how to build Aneka Cloud with the diagram. Or Two modes of Aneka Organization (8M ,Feb2023)

Solution: Building Aneka clouds Aneka Cloud can be realized by two methods:

1. Infrastructure Organization
2. Logical Organization

Infrastructure based organization of Aneka Cloud is given in the following figure The working mechanism of this model: It contains Aneka Repository, Administrative Console, Aneka Containers & Node Managers as major components.

The Management Console manages multiple repositories and select the one that best suits the specific deployment. A number of node managers and Aneka containers are deployed across the cloud platform to provision necessary services, The Aneka node manager are also known as AnekaDaemon The collection of resulting containers identifies the final AnekaCloud

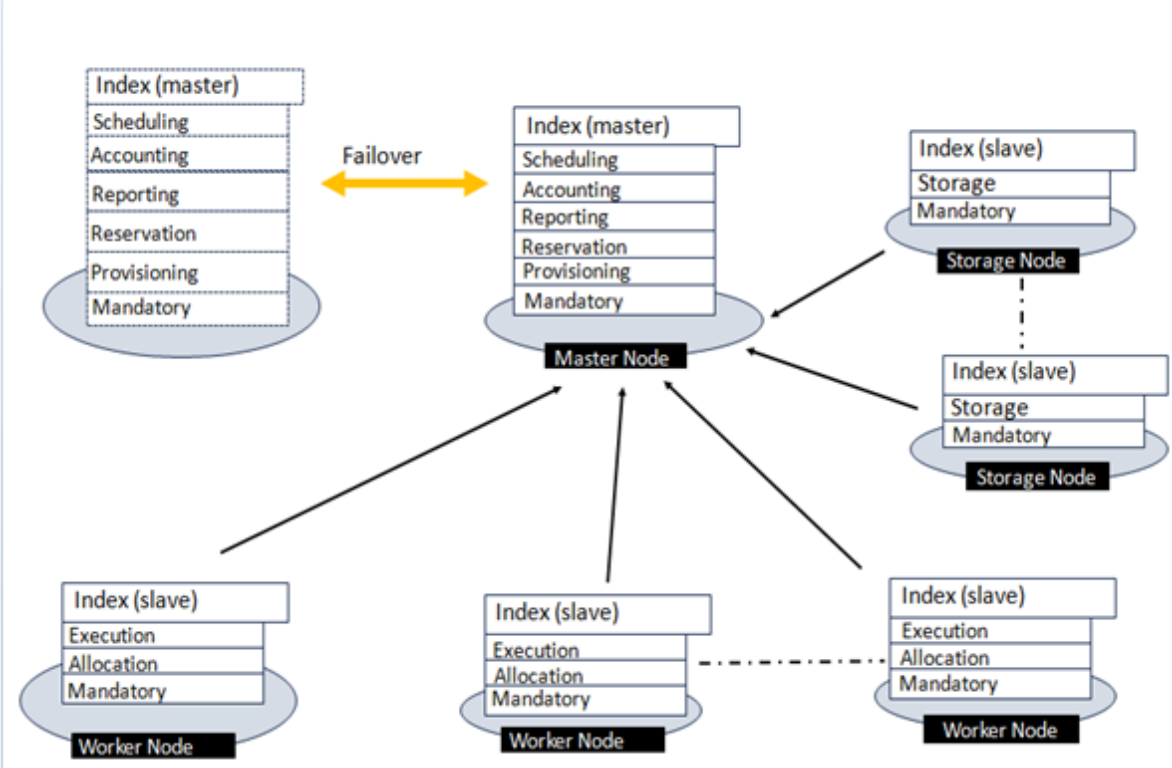


Logical organization

The logical organization of Aneka Clouds can be very diverse, since it strongly depends on the configuration selected for each of the container instances belonging to the Cloud. Here is a scenario that has master-worker configuration with separate nodes for storage, the Figure. Portray The master node comprises of following services:

- Index Service (master copy)
- Heartbeat Service
- Logging Service
- Reservation Service
- Resource Provisioning Service
- Accounting Service
- Reporting and Monitoring Service
- Scheduling Services for the supported programming models

Here Logging service and Heartbeat service and Monitoring service are considered as Mandatory services in all the block diagrams whereas other services are shown ditto.



Similarly the Worker Node comprises of following services:

- Index Service
- Execution service
- Allocation service
- And mandatory (Logging, Heartbeat and monitoring services)

The Storage Node comprises of :

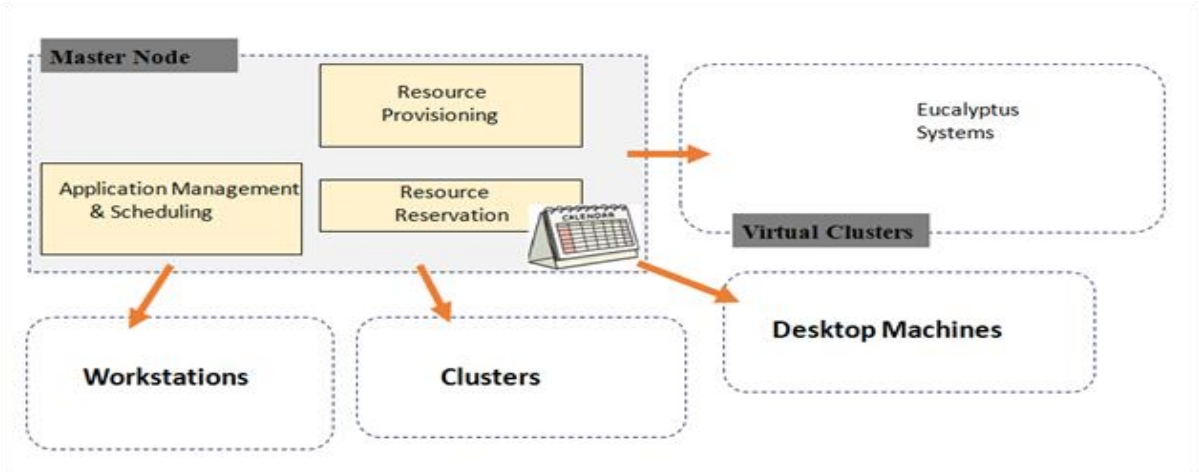
- Index service
- Storage Service
- And mandatory (Logging, Heartbeat and monitoring services)

In addition all nodes are registered with the master node and transparently refer to any failover partner in the case of a high-availability configuration

7. Describe how to build Aneka Cloud Deployment models with the diagram.
Note: 8M if they ask each type,10M for all the type (Dec 2018, July 2022)

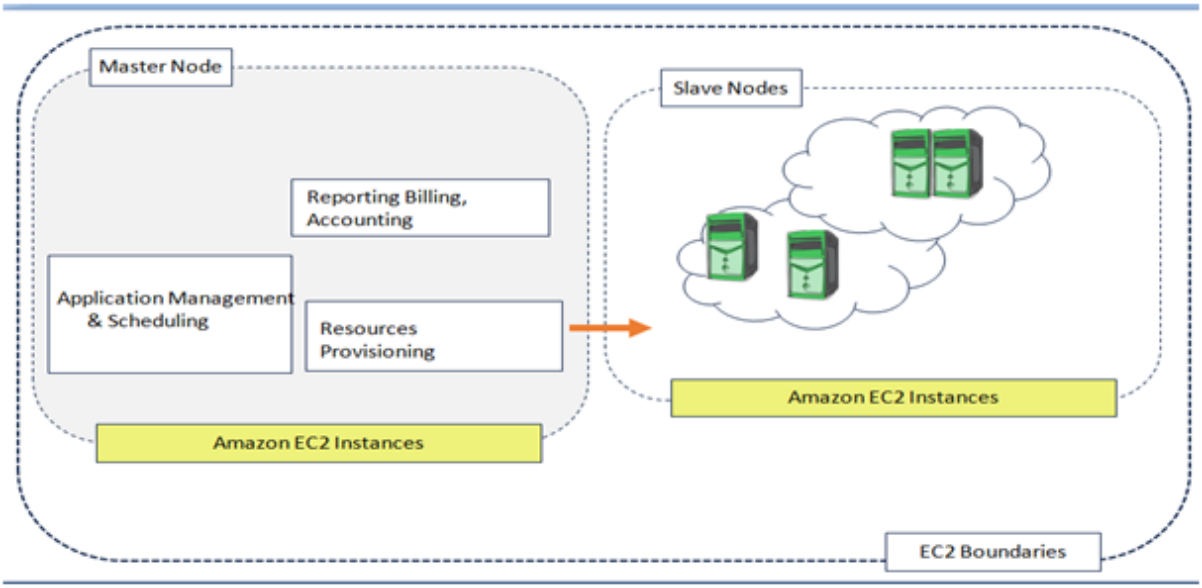
Solution:
Aneka Cloud Deployment Models
All the general cloud deployment models like
Private cloud deployment mode, Public cloud deployment mode and Hybrid Cloud deployment mode are applicable to Aneka Clouds also.

Private cloud deployment mode A private deployment mode is mostly constituted by local physical resources and infrastructure management software providing access to a local pool of nodes, which might be virtualized. Figure shows a common deployment for a private Aneka Cloud. This deployment is acceptable for a scenario in which the workload of the system is predictable and a local virtual machine manager can easily address excess capacity demand.
The different nature of the machines harnessed in a private environment allows for specific policies on resource management and usage that can be accomplished by means of the Reservation Service. For example, desktop machines that are used during the day for office automation can be exploited outside the standard working hours to execute distributed applications.
Note: In the master node: Resource Provisioning, Application Management & Scheduling and Resource Reservation are the primary services.



Public cloud deployment mode
Public Cloud deployment mode features the installation of Aneka master and worker nodes over a completely virtualized infrastructure that is hosted on the infrastructure of one or more resource providers such as Amazon EC2 or GoGrid.
Figure provides an overview of this scenario. The deployment is generally contained within the infrastructure boundaries of a single IaaS pro- vider. The reasons for this are to minimize the data transfer between different providers, which is generally priced at a higher cost, and to have better

network performance. In this scenario it is possible to deploy an Aneka Cloud composed of only one node and to completely leverage dynamic provisioning to elastically scale the infrastructure on demand. A fundamental role is played by the Resource Provisioning Service, which can be configured with different images and templates to instantiate. Other important services that have to be included in the master node are the Accounting and Reporting Services. These provide details about resource utilization by users and applications and are fundamental in a multitenant Cloud where users are billed according to their consumption of Cloud capabilities.



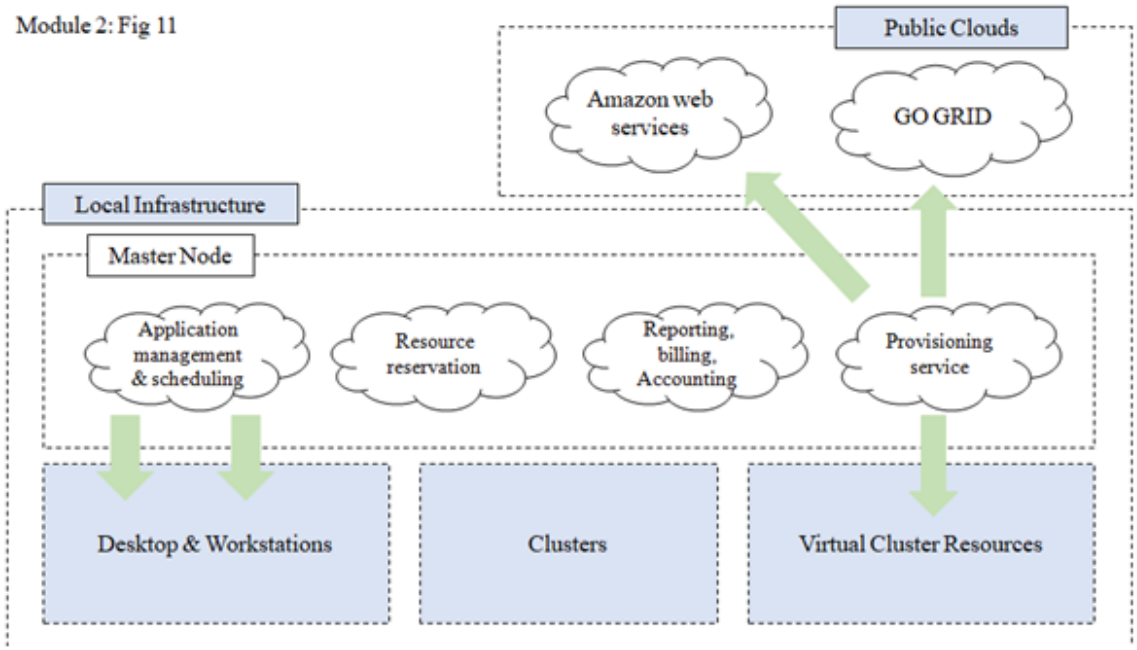
Hybrid cloud deployment mode

The hybrid deployment model constitutes the most common deployment of Aneka. In many cases, there is an existing computing infrastructure that can be leveraged to address the computing needs of applications. This infrastructure will constitute the static deployment of Aneka that can be elastically scaled on demand when additional resources are required

An overview of this deployment is presented in Figure. This scenario constitutes the most complete deployment for Aneka that is able to leverage all the capabilities of the framework:

• Dynamic Resource Provisioning
• Resource Reservation
• Workload Partitioning (Scheduling)
• Accounting, Monitoring, and Reporting

- In a hybrid scenario, heterogeneous resources can be used for different purposes. As we discussed in the case of a private cloud deployment, desktop machines can be reserved for low priority work- load outside the common working hours.
- The majority of the applications will be executed on work- stations and clusters, which are the nodes that are constantly connected to the Aneka Cloud. Any additional computing capability demand can be primarily addressed by the local virtualization facili- ties, and if more computing power is required, it is possible to leverage external IaaS providers.



8. Describe the features of the Aneka management tools in terms of infrastructure, platform, and applications. (10M, Feb 2022)

Solution:

Management tools

Aneka is a pure PaaS implementation and requires virtual or physical hardware to be deployed. Hence, infrastructure management, together with facilities for installing logical clouds on such infrastructure, is a fundamental feature of Aneka's management layer. This layer also includes capabilities for managing services and applications running in the Aneka Cloud.

Infrastructure management Aneka leverages virtual and physical hardware in order to deploy Aneka Clouds. Virtual hardware is generally managed by means of the Resource Provisioning Service, which acquires resources on demand according to the need of applications, while physical hardware is directly managed by the Administrative Console by leveraging the Aneka management API of the PAL. The management features are mostly concerned with the provisioning of physical hardware and the remote installation of Aneka on the hardware.

Platform management Infrastructure management provides the basic layer on top of which Aneka Clouds are deployed. The creation of Clouds is orchestrated by deploying a collection of services on the physical infrastructure that allows the installation and the management of containers. A collection of connected containers defines the platform on top of which applications are executed. The features available for platform management are mostly concerned with the logical organization and structure of Aneka Clouds. It is possible to partition the available hardware into several Clouds variably configured for different purposes. Services implement the core features of Aneka Clouds and the management layer exposes operations for some of them, such as Cloud monitoring, resource provisioning and reservation, user management, and application profiling.

Application management Applications identify the user contribution to the Cloud. The management APIs provide administrators with monitoring and profiling features that help them track the usage of resources and relate them to users and applications. This is an important feature in a cloud computing scenario in which Cloud programming and management users are billed for their resource usage. Aneka exposes capabilities for giving summary and detailed information about application execution

and resource utilization. All these features are made accessible through the Aneka Cloud Management Studio, which constitutes the main Administrative Console for the Cloud.

10. What are the major features of Aneka's application model? Explain Aneka Service model. (Dec 2018, July 2022)

Solution:

Application Model represents the minimum set of APIs that is common to all the programming models for representing and programming distributed applications on top of Aneka. This model is further specialized according to the needs and the particular features of each of the programming models.

- Table below summarizes the features that are available in the Aneka Application Model and the way they reflect into the supported programming model. The model has been designed to be extensible, and these classes can be used as a starting point to implement a new programming model. This can be done by augmenting the features (or specializing) an existing implementation of a programming model or by using the base classes to define new models and abstractions.
- For example, the Parameter Sweep Model is a specialization of the Task Model, and it has been implemented in the context of management of applications on Aneka. It is achieved by providing a different interface to end users who just need to define a template task and the parameters that customize it.

Table 5.1 Aneka's Application Model Features

Category	Description	Base Application Type	Work Units?	Programming Models
Manual	Units of work are generated by the user and submitted through the application.	*AnekaApplication<W,M>* *IManualApplicationManager<W>* *ManualApplicationManager<W>*	Yes	Task Model Thread Model Parameter Sweep Model
Auto	Units of work are generated by the runtime infrastructure and managed internally.	*ApplicationBase<M>* *IAutoApplicationManager*	No	*MapReduce* Model

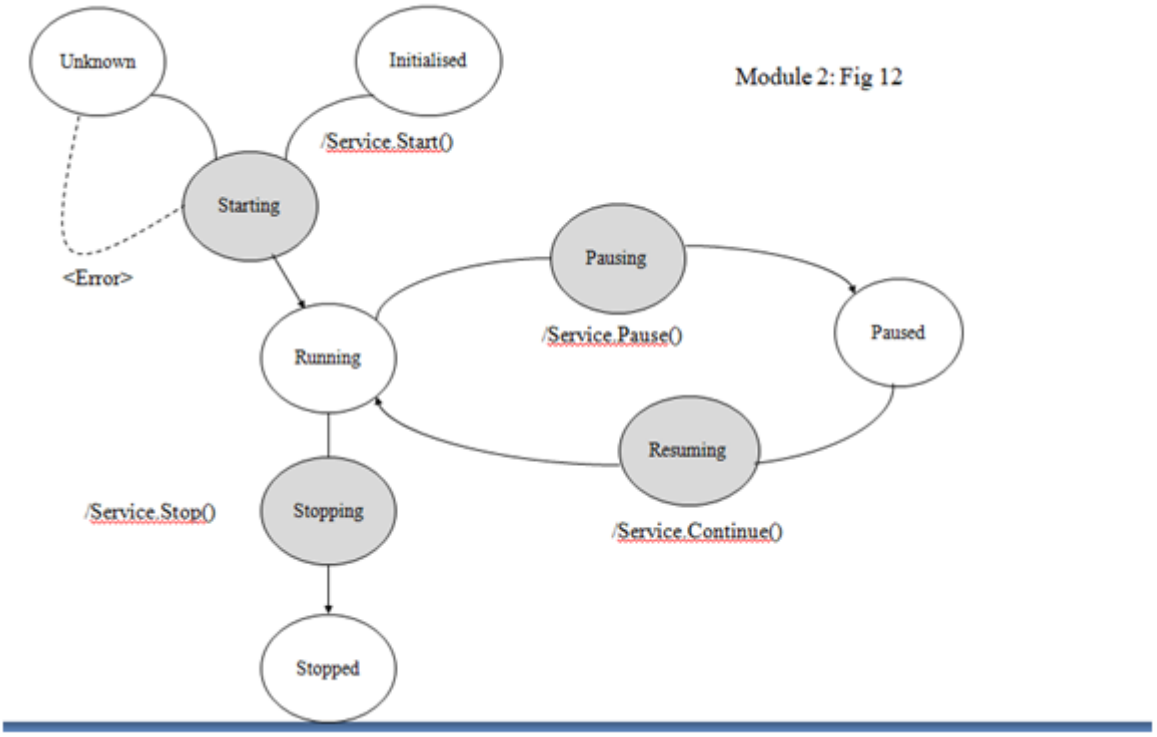
Service Model: The Aneka Service Model defines the basic requirements to implement a service that can be hosted in an Aneka Cloud.

The container defines the runtime environment in which services are hosted. Each service that is hosted in the container must be compliant with the IService interface, which exposes the following methods and properties:

• Name and status
• Control operations such as Start, Stop, Pause, and Continue methods
• Message handling by means of the HandleMessage method

A service instance can initially be in the Unknown or Initialized state, a condition that refers to the creation of the service instance by invoking its constructor during the configuration of the container. Once the container is started, it will iteratively call the Start method on each service method. As a result, the service instance is expected to be in a Starting state until the startup process is completed, after which it will exhibit the Running state.

This is the condition in which the service will last as long as the container is active and running. This is the only state in which the service is able to process messages. If an exception occurs while starting the service, it is expected that the service will fall back to the Unknown state, thus signaling an error.

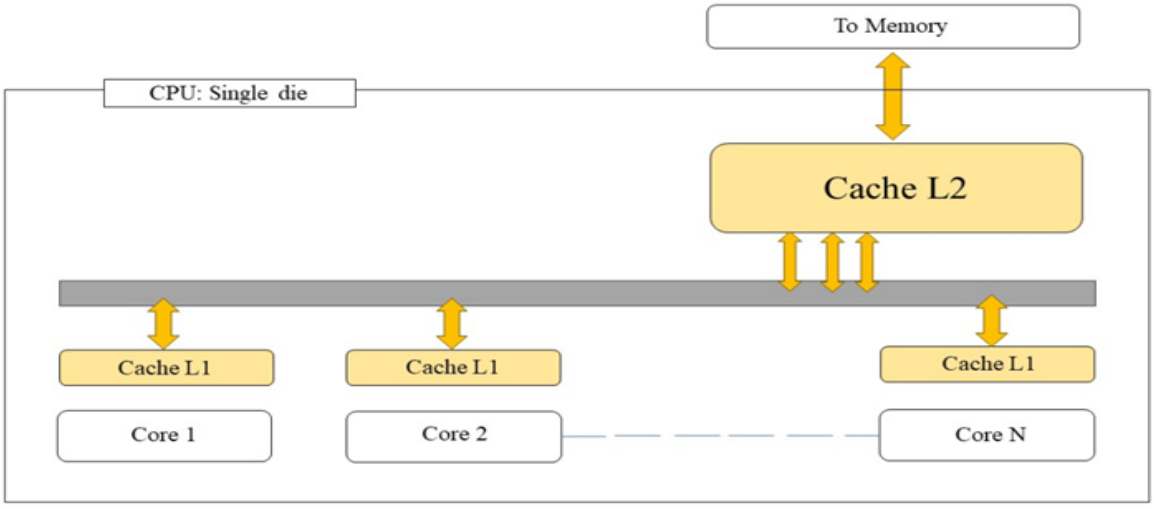


MODULE 3

1. Give a brief overview of parallel computing. (6M , FEB 2023)

Solution:

- Parallelism has been a technique for improving the performance of computers since the early 1960s. In particular, multiprocessing, which is the use of multiple processing units within a single machine, has gained a good deal of interest and gave birth to several parallel architectures.
- Asymmetric multiprocessing involves the concurrent use of different processing units that are specialized to perform different functions. Symmetric multiprocessing features the use of similar or identical processing units to share the computation load.



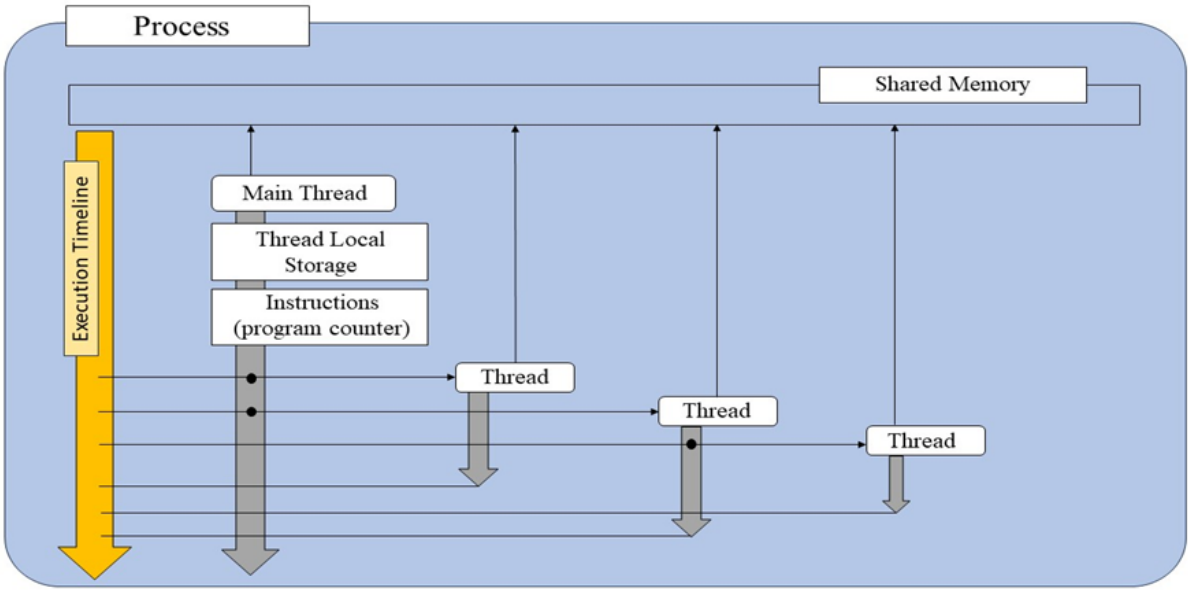
- Multicore systems are composed of a single processor that features multiple processing cores that share the memory. Each core generally has its own L1 cache, and the L2 cache is common to all the cores, which connect to it by means of a shared bus, as depicted in Figure.
- Dual- and quad-core configurations are quite popular nowadays and constitute the standard hardware configuration for commodity computers. Architectures with multiple cores are also available but are not designed for the commodity market. Multicore technology has been used not only as a support for processor design but also in other devices, such as GPUs and network devices, thus becoming a standard practice for improving performance.
- Multiprocessing is just one technique that can be used to achieve parallelism, and it does that by leveraging parallel hardware architectures. In particular, an important role is played by the operating system, which defines the runtime structure of applications by means of the abstraction of process and thread. A process is the runtime image of an application, or better, a program that is running, while a thread identifies a single flow of the execution within a process. A system that allows the execution of multiple processes at the same time supports multitasking. It supports multithreading when it provides structures for explicitly defining multiple threads within a process.

2. Define thread. Explain the relationship between process and thread with a neat diagram. (8M , JULY 2022)

Solution:

Modern applications perform multiple operations at the same time. The use of threads might be implicit or explicit.
Implicit threading happens when the underlying APIs use internal threads to perform specific tasks supporting the execution of applications such as graphical user interface (GUI) rendering, or garbage collection in the case of virtual machine-based languages.
Explicit threading is characterized by the use of threads within a program by application developers, who use this abstraction to introduce parallelism. Common cases in which threads are explicitly used are I/O from devices and network connections, long computations, or the execution of background operations.

- **What is a thread?**
- A thread identifies a single control flow, which is a logical sequence of instructions, within a process. By logical sequence of instructions, we mean a sequence of instructions that have been designed to be executed one after the other one. Operating systems that support multithreading identify threads as the minimal building blocks for expressing running code. Each process contains at least one thread but, in several cases, is composed of many threads having variable lifetimes. Threads within the same process share the memory space and the execution context.



- In a multitasking environment the operating system assigns different time slices to each process and interleaves their execution. The process of temporarily stopping the execution of one process, saving all the information in the registers, and replacing it with the information related to another process is known as a context switch
- Figure provides an overview of the relation between threads and processes and a simplified representation of the runtime execution of a multithreaded application.

3. Explain parallel computing techniques with threads. (8M, FEB 2023)

Solution:
Techniques for parallel computation with threads
Developing parallel applications requires an understanding of the problem and its logical structure. Decomposition is a useful technique that aids in understanding whether a problem is divided into

components (or tasks) that can be executed concurrently. it allows the breaking down into independent units of work that can be executed concurrently with the support provided by threads.

1 Domain decomposition
2 Functional decomposition
3 Computation vs. Communication

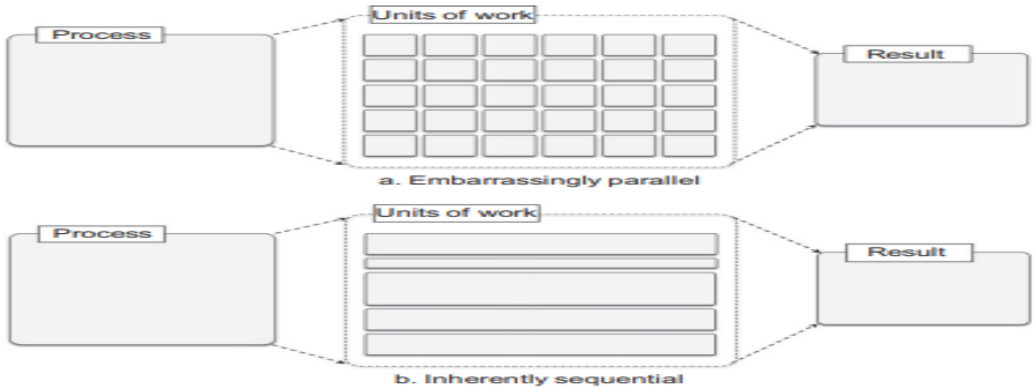
1 Domain decomposition: Domain decomposition is the process of identifying patterns of functionally repetitive, but independent, computation on data. This is the most common type of decomposition in the case of throughput computing, and it relates to the identification of repetitive calculations required for solving a problem. The master-slave model is a quite common organization for these scenarios:

- Embarrassingly parallel problems are quite common, they are based on the strong assumption that at each of the iterations of the decomposition method, it is possible to isolate an independent unit of work. This is what makes it possible to obtain a high computing throughput. If the values of all the iterations are dependent on some of the values obtained in the previous iterations, the problem is said to be inherently sequential.
- Figure provides a schematic representation of the decomposition of embarrassingly parallel and inherently sequential problems. The matrix product computes each element of the resulting matrix as a linear combination of the corresponding row and column of the first and second input matrices, respectively.

$$C_{ij} = \sum_{k=0}^{n-1} A_{ik} B_{kj}$$

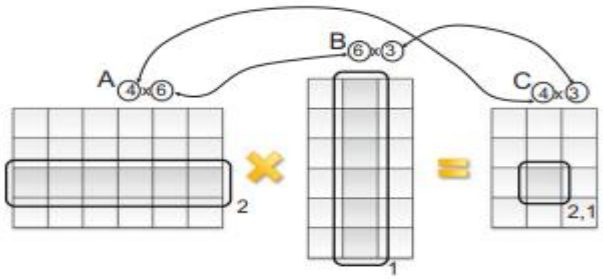
- The formula that applies for each of the resulting matrix elements is the following: Two conditions hold in order to perform a matrix product:

• Input matrices must contain values of a comparable nature for which the scalar product is defined.
• The number of columns in the first matrix must match the number of rows of the second matrix



The problem is embarrassingly parallel, and we can logically organize the multithreaded program in the following steps:

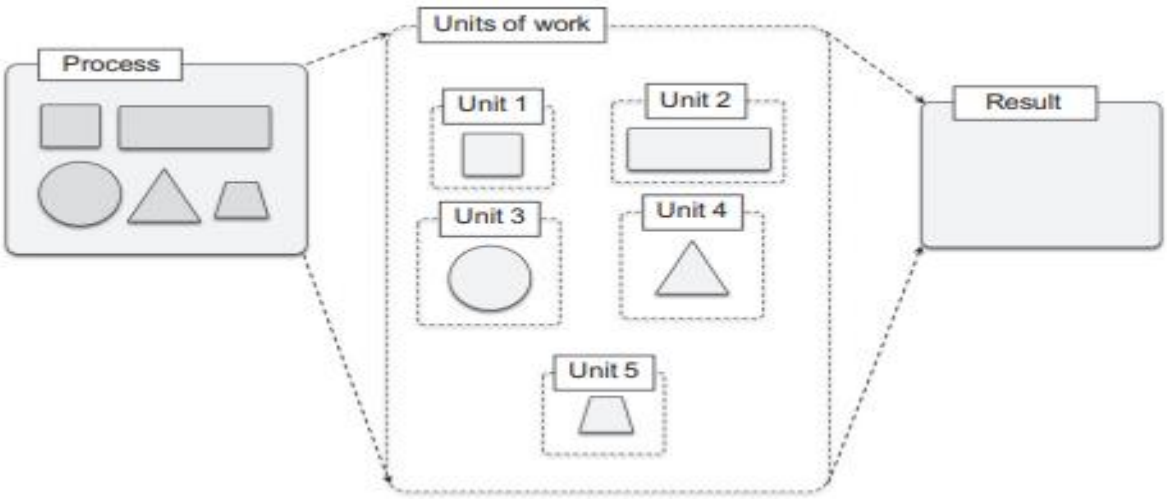
• Define a function that performs the computation of the single element of the resulting matrix by implementing the previous equation.
• Create a double for loop (the first index iterates over the rows of the first matrix and the second over the columns of the second matrix) that spawns a thread to compute the elements of the resulting matrix.
• Join all the threads for completion, and compose the resulting matrix.



The .NET framework provides the System.Threading. Thread class that can be configured with a function pointer, also known as a delegate, to execute asynchronously. This class will also define the method for performing the actual computation.

Functional decomposition

- Functional decomposition is the process of identifying functionally distinct but independent computations. The focus here is on the type of computation rather than on the data manipulated by the computation. This kind of decomposition is less common and does not lead to the creation of a large number of threads, since the different computations that are performed by a single program are limited. Functional decomposition leads to a natural decomposition of the problem in separate units of work. Figure provides a pictorial view of how decomposition operates and allows parallelization.
- The problems that are subject to functional decomposition can also require a composition phase in which the outcomes of each of the independent units of work are composed together. In the following, we show a very simple example of how a mathematical problem can be parallelized using functional decomposition. Suppose, for example, that we need to calculate the value of the following function for a given value of x: f(x) = sin(x) + cos(x) + tan(x)



- Once the value of x has been set, the three different operations can be performed independently of each other. This is an example of functional decomposition because the entire problem can be separated into three distinct operations.

Computation vs. Communication

- It is very important to carefully evaluate the communication patterns among the components that have been identified during problem decomposition. The two decomposition methods

presented in this section and the corresponding sample applications are based on the assumption that the computations are independent. This means that:

- The input values required by one computation do not depend on the output values generated by another computation.
- The different units of work generated as a result of the decomposition do not need to interact (i.e., exchange data) with each other. These two assumptions strongly simplify the implementation and allow achieving a high degree of parallelism and a high throughput.

4. Explain multithreading with Aneka 8M (FEB 2022)
Solution:
Multithreading with Aneka

As applications become increasingly complex, there is greater demand for computational power that can be delivered by a single multicore machine.
Often this demand cannot be addressed with the computing capacity of a single machine. It is then necessary to leverage distributed infrastructures such as clouds.
Decomposition techniques can be applied to partition a given application into several units of work, submitted for execution by leveraging clouds.
Aneka, as middleware for managing clusters, grids, and clouds, provides developers with advanced capabilities for implementing distributed applications.
Aneka threads, as they are called, let you easily port existing multithreaded compute-intensive applications to distributed versions that can run faster by utilizing multiple machines simultaneously, with minimum conversion effort.

Introducing the thread programming model Aneka offers the capability of implementing multithreaded applications over the cloud by means of the

Thread Programming Model. This model introduces the abstraction of distributed thread, also called Aneka thread, which mimics the behavior of local threads but executes over a distributed infrastructure. This model provides the best advantage in the case of embarrassingly parallel applications.
The Thread Programming Model exhibits APIs that mimic the ones exposed by .NET base class libraries for threading. In this way developers do not have to completely rewrite applications in order to leverage Aneka by replacing the System.Threading.Thread class and introducing the AnekaApplication class. There are three major elements that constitute the object model of applications based on the Thread Programming Model.

Application. This class represents the interface to the Aneka middleware and constitutes a local view of a distributed application. In the Thread Programming Model the single units of work are created by the programmer. Therefore, the specific class used will be Aneka.Entity. AnekaApplication, with T and M properly selected. Threads.

Threads represent the main abstractions of the model and constitute the building blocks of the distributed application. Aneka provides the Aneka.Threading.AnekaThread class, which represents a distributed thread.

Thread Manager. This is an internal component that is used to keep track of the execution of distributed threads and provide feedback to the application. Aneka provides a specific version of the manager for this model, which is implemented in the Aneka.Threading.ThreadManager class

Aneka thread vs. common threads

To efficiently run on a distributed infrastructure, Aneka threads have certain limitations compared to local threads. These limitations relate to the communication and synchronization strategies.

1 Interface compatibility
2 Thread life cycle
3 Thread synchronization
4 Thread priorities
5 Type serialization

1 Interface compatibility The AnekaThread class exposes almost the same interface as the System.Threading.Thread class with the exception of a few operations that are not supported. Table compares the operations that are exposed by the two classes.

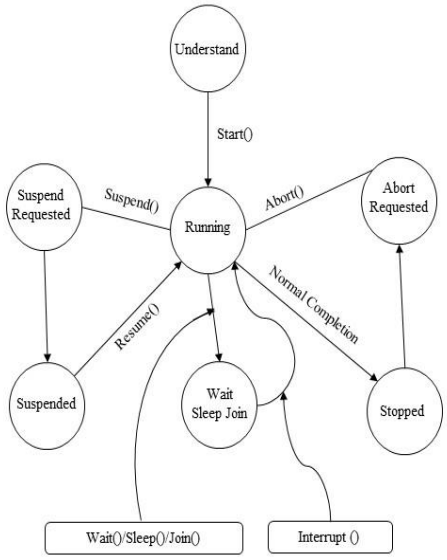
The basic control operations for local threads such as Start and Abort have a direct mapping, whereas operations that involve the temporary interruption of the thread execution have not been supported. The reasons for such a design decision are twofold. First, the use of the Suspend/Resume operations is generally a deprecated practice, even for local threads, since Suspend abruptly interrupts the execution state of the thread. Second, thread suspension in a distributed environment leads to an ineffective use of the infrastructure, where resources are shared among different tenants and applications. Sleep operation is not supported. Therefore, there is no need to support the Interrupt operation, which forcibly resumes the thread from a waiting or a sleeping state.

2 Thread life cycle Aneka Thread life cycle is different from the life cycle of local threads. It is not possible to directly map the state values of a local thread to Aneka threads. Figure provides a comparative view of the two life cycles.

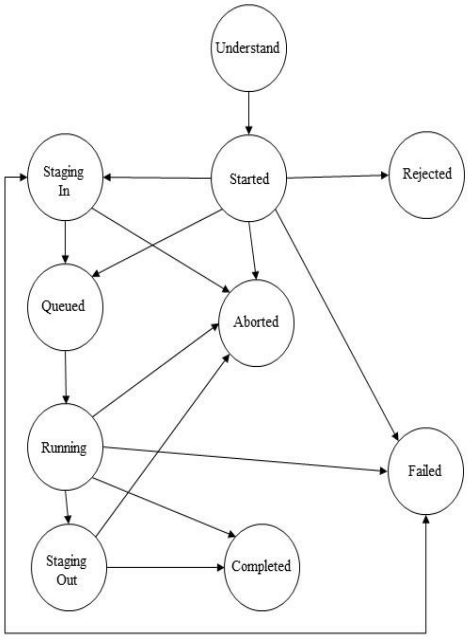
The white balloons in the figure indicate states that do not have a corresponding mapping on the other life cycle; the shaded balloons indicate the common states.

.

- An Aneka thread is initially found in the Unstarted state. Once the Start() method is called, the thread transits to the Started state, from which it is possible to move to the StagingIn state if there are files to upload for its execution or directly to the Queued state. If there is any error while uploading files, the thread fails and it ends its execution with the Failed state, which can also be reached for any exception that occurred while invoking Start().
- This is a final state and implies execution failure due to lack of rights. Once the thread is in the queue, if there is a free node where to execute it, the middleware moves all the object data and dependent files to the remote node and starts its execution, thus changing the state into Running. If the thread generates an exception or does not produce the expected output files, the execution is considered failed and the final state of the thread is set to Failed. If the execution is successful, the final state is set to Completed



a. System.Threading.Threadlife cycle.



b. Aneka.Threading.AnekaThreadlife cycle.

3 Thread synchronization: the .NET base class libraries provide advanced facilities to support thread synchronization by the means of monitors, semaphores, reader-writer locks, and basic synchronization constructs at the language level. Aneka provides minimal support for thread synchronization that is limited to the implementation of the joint operation for thread abstraction. This requirement is less stringent in a distributed environment, where there is no shared memory among the thread instances and therefore it is not necessary. Providing coordination facilities that introduce a locking strategy in such an environment might lead to distributed deadlocks that are hard to detect. Therefore, by design Aneka threads do not feature any synchronization facility except join operation.

4 Thread priorities: The System.Threading. Thread class supports thread priorities, where the scheduling priority can be one selected from one of the values of the ThreadPriority enumeration: Highest, AboveNormal, Normal, BelowNormal, or Lowest. Aneka does not support thread priorities, the Aneka.Threading.Thread class exhibits a Priority property whose type is ThreadPriority, but its value is always set to Normal, and changes to it do not produce any effect on thread scheduling by the Aneka middleware.

5 Type serialization: Serialization is the process of converting an object into a stream of bytes to store the object or transmit it to memory, a database, or a file. Main purpose is to save the state of an object to recreate it when needed.

- The reverse process is called deserialization. Local threads execute all within the same address space and share memory; therefore, they do not need objects to be copied or transferred into a different address space.
- Aneka threads are distributed and execute on remote computing nodes, and this implies that the object code related to the method to be executed within a thread needs to be transferred over the network.

5. Differences between Aneka thread vs. common threads (8M , DEC 2018)

Solution:

Local thread.

- In computer science, a thread of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system.
- The implementation of threads and processes differs between operating systems, but in most cases, a thread is a component of a process.
- Multiple threads can exist within one process, executing concurrently and sharing resources such as memory, while different processes do not share these resources.
- In particular, the threads of a process share its executable code and the values of its dynamically allocated variables and non-thread-local global variables at any given time

Aneka thread

- Aneka offers the capability of implementing multi-threaded applications over the Cloud by means of the Thread Programming Model.
- This model introduces the abstraction of distributed thread, also called Aneka thread, which mimics the behavior of local threads but executes over a distributed infrastructure.

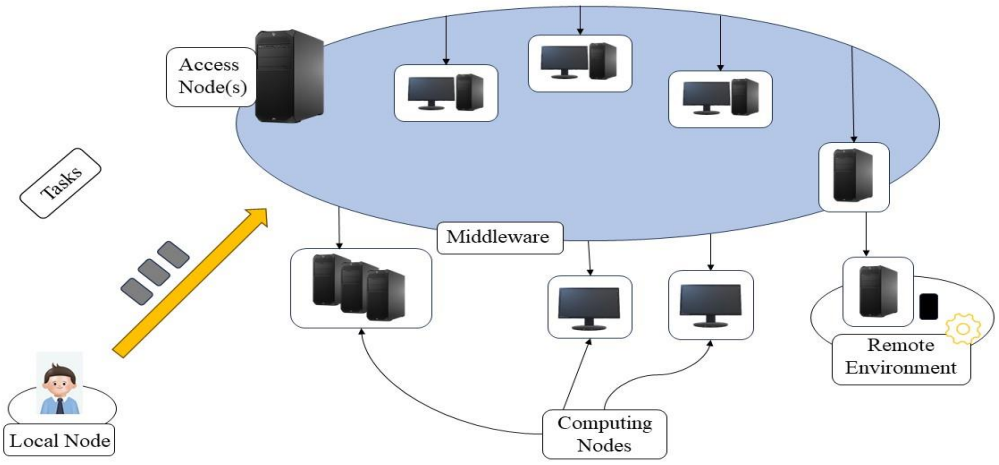
6. What is Task computing? Explain the task computing scenario with a neat diagram?

(6M, Dec2018, Feb 2023)

A task identifies one or more operations that produce a distinct output and that can be isolated as a single logical unit.

In practice, a task is represented as a distinct unit of code, or a program, that can be separated and executed in a remote run time environment.

A reference scenario for task computing is depicted in **Figure 7.1**.



The middleware is a software layer that enables the coordinated use of multiple resources, which are drawn from a data center or geographically distributed networked computers.

A user submits the collection of tasks to the access point(s) of the middleware, which will take care of scheduling and monitoring the execution of tasks.

Each computing resource provides an appropriate runtime environment.

Task submission is done using the APIs provided by the middleware, whether a Web or programming language interface.

Appropriate APIs are also provided to monitor task status and collect their results upon completion.

It is possible to identify a set of common operations that the middleware needs to support the creation and execution of task-based applications. These operations are:

- Coordinating and scheduling tasks for execution on a set of remote nodes
- Moving programs to remote nodes and managing their dependencies
- Creating an environment for execution of tasks on the remote nodes
- Monitoring each task's execution and informing the user about its status
- Access to the output produced by the task.

7. What are the Characterizing a task? (4M, July 2022)

A task represents a component of an application that can be logically isolated and executed separately. A task can be represented by different elements:

- A shell script composing together the execution of several applications
- A single program

A unit of code (a Java/C11/.NET class) that executes within the context of a specific runtime environment. A task is characterized by input files, executable code (programs, shell scripts, etc.), and output files.

The runtime environment in which tasks execute is the operating system or an equivalent sandboxed environment.

A task may also need specific software appliances on the remote execution nodes.

8. Write a note on Computing categories/ categories of task computing?

(6M, JUNE 2022)

These categories provide an overall view of the characteristics of the problems. They implicitly impose requirements on the infrastructure and the middleware.

Applications falling into this category are:

1 High-performance computing

2 High-throughput computing

3 Many-task computing

1 High-performance computing

High-performance computing (HPC) is the use of distributed computing facilities for solving problems that need large computing power.

The general profile of HPC applications is constituted by a large collection of compute-intensive tasks that need to be processed in a short period of time.

The metrics to evaluate HPC systems are floating-point operations per second (FLOPS), now tera-FLOPS or even peta-FLOPS, which identify the number of floating- point operations per second.

Ex: supercomputers and clusters are specifically designed to support HPC applications that are developed to solve "Grand Challenge" problems in science and engineering.

2 High-throughput computing

High-throughput computing (HTC) is the use of distributed computing facilities for applications requiring large computing power over a long period of time.

HTC systems need to be robust and to reliably operate over a long time scale.

The general profile of HTC applications is that they are made up of a large number of tasks of which the execution can last for a considerable amount of time.

Ex: scientific simulations or statistical analyses.

It is quite common to have independent tasks that can be scheduled in distributed resources because they do not need to communicate.

HTC systems measure their performance in terms of jobs completed per month.

3 Many-task computing

MTC denotes high-performance computations comprising multiple distinct activities coupled via file system operations.

MTC is the heterogeneity of tasks that might be of different nature: Tasks may be small or large, single- processor or multiprocessor, compute-intensive or data-intensive, static or dynamic, homogeneous or heterogeneous.

MTC applications includes loosely coupled applications that are communication-intensive but not naturally expressed using the message-passing interface.

It aims to bridge the gap between HPC and HTC. MTC is similar to HTC, but it concentrates on the use of many computing resources over a short period of time to accomplish many computational tasks.

9. Explain the Frameworks for task computing (10M, Feb 2023)

Some popular software systems that support the task-computing framework are:

1. **Condor**
2. **Globus Toolkit**
3. **Sun Grid Engine (SGE)**
4. **BOINC**
5. **Nimrod/G**

They consist of two main components: a scheduling node (one or more) and worker nodes. The organization of the system components may vary.

1. Condor

Condor is the most widely used and long-lived middleware for managing clusters, idle workstations, and a collection of clusters.

Condor supports features of batch-queuing systems along with the capability to checkpoint jobs and manage overload nodes.

It provides a powerful job resource-matching mechanism, which schedules jobs only on resources that have the appropriate runtime environment.

Condor can handle both serial and parallel jobs on a wide variety of resources.

It is used by hundreds of organizations in industry, government, and academia to manage infrastructures. Condor-G is a version of Condor that supports integration with grid computing resources, such as those managed by Globus.

2. Globus Toolkit

The Globus Toolkit is a collection of technologies that enable grid computing.

It provides a comprehensive set of tools for sharing computing power, databases, and other services across corporate, institutional, and geographic boundaries.

The toolkit features software services, libraries, and tools for resource monitoring, discovery, and management as well as security and file management.

The toolkit defines a collection of interfaces and protocol for interoperation that enable different systems to integrate with each other and expose resources outside their boundaries.

3. Sun Grid Engine (SGE)

Sun Grid Engine (SGE), now Oracle Grid Engine, is middleware for workload and distributed resource management.

Initially developed to support the execution of jobs on clusters, SGE integrated additional capabilities and now is able to manage heterogeneous resources and constitutes middleware for grid computing.

It supports the execution of parallel, serial, interactive, and parametric jobs and features advanced scheduling capabilities such as budget-based and group- based scheduling, scheduling applications that have deadlines, custom policies, and advance reservation.

4. BOINC

Berkeley Open Infrastructure for Network Computing (BOINC) is framework for volunteer and grid computing. It allows us to turn desktop machines into volunteer computing nodes that are leveraged to run jobs when such machines become inactive.

BOINC supports job check pointing and duplication.

BOINC is composed of two main components: the BOINC server and the BOINC client.

The BOINC server is the central node that keeps track of all the available resources and scheduling jobs. The BOINC client is the software component that is deployed on desktop machines and that creates the BOINC execution environment for job submission.
BOINC systems can be easily set up to provide more stable support for job execution by creating computing grids with dedicated machines.
When installing BOINC clients, users can decide the application project to which they want to donate the CPU cycles of their computer.
Currently several projects, ranging from medicine to astronomy and cryptography, are running on the BOINC infrastructure.

5. Nimrod/G

Tool for automated modeling and execution of parameter sweep applications over global computational grids.
It provides a simple declarative parametric modeling language for expressing parametric experiments. It uses novel resource management and scheduling algorithms based on economic principles.
It supports deadline- and budget-constrained scheduling of applications on distributed grid resources to minimize the execution cost and at the same deliver results in a timely manner.
It has been used for a very wide range of applications over the years, ranging from quantum chemistry to policy and environmental impact.

10.Write a note on Task-based application models

(6M,Model QP,July2022)

There are several models based on the concept of the task as the fundamental unit for composing distributed applications.
What makes these models different from one another is the way in which tasks are generated, the relationships they have with each other, and the presence of dependencies or other conditions.
In this section, we quickly review the most common and popular models based on the concept of the task.

7.1.2 Embarrassingly parallel applications

Embarrassingly parallel applications constitute the most simple and intuitive category of distributed applications.
The tasks might be of the same type or of different types, and they do not need to communicate among themselves.
This category of applications is supported by the majority of the frameworks for distributed computing. Since tasks do not need to communicate, there is a lot of freedom regarding the way they are scheduled.
Tasks can be executed in any order, and there is no specific requirement for tasks to be executed at the same time.

7.1.3 Parameter sweep applications

Parameter sweep applications are a specific class of embarrassingly parallel applications for which the tasks are identical in their nature and differ only by the specific parameters used to execute.

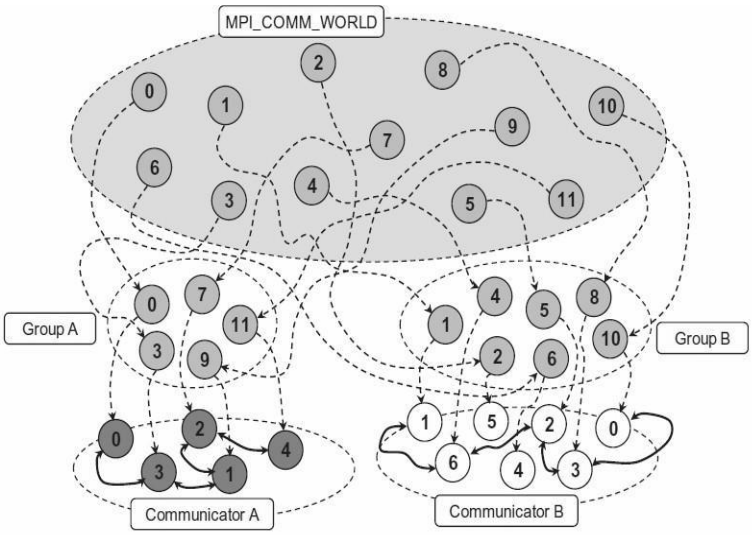
Parameter sweep applications are identified by a template task and a set of parameters. The template task defines the operations that will be performed on the remote node for the execution of tasks.
The parameter set identifies the combination of variables whose assignments specialize the template task into a specific instance.
Any distributed computing framework that provides support for embarrassingly parallel applications can also support the execution of parameter sweep applications.
The only difference is that the tasks that will be executed are generated by iterating over all the possible and admissible combinations of parameters.

11. With a neat diagram explain MPI applications (10M, July 2022)

Message Passing Interface (MPI) is a specification for developing parallel programs that communicate by exchanging messages.
MPI has originated as an attempt to create common ground from the several distributed shared memory and message-passing infrastructures available for distributed computing. Now a days, MPI has become a de facto standard for developing portable and efficient message-passing HPC applications.

MPI provides developers with a set of routines that:
• Manage the distributed environment where MPI programs are executed
• Provide facilities for point-to-point communication
• Provide facilities for group communication
• Provide support for data structure definition and memory allocation
• Provide basic support for synchronization with blocking calls

The general reference architecture is depicted in Figure 7.4. A distributed application in MPI is composed of a collection of MPI processes that are executed in parallel in a distributed



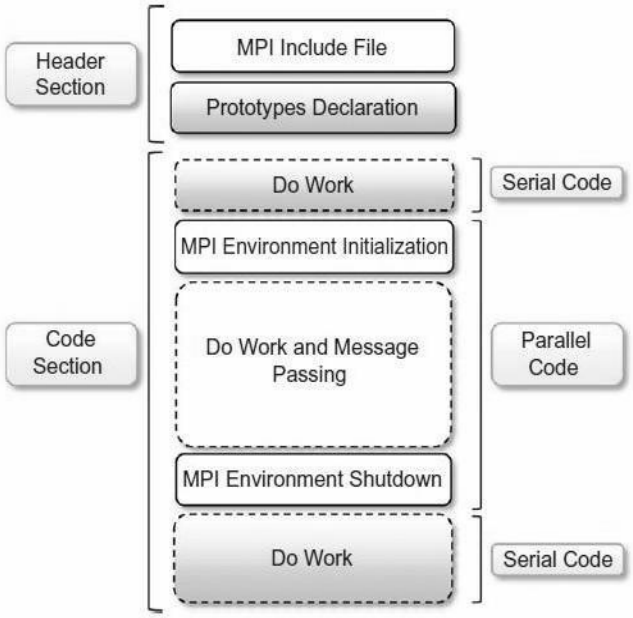
infrastructure that supports MPI.

MPI applications that share the same MPI runtime are by default as part of a global group called MPI_COMM_WORLD. Within this group, all the distributed processes have a unique identifier that allows the MPI runtime to localize and address them.

Each MPI process is assigned a rank within the group.
The rank is a unique identifier that allows processes to communicate with each other within a group.

To create an MPI application it is necessary to define the code for the MPI process that will be executed in parallel. This program has, in general, the structure described in **Figure 7.5**.

The section of code that is executed in parallel is clearly identified by two operations that set up the MPI environment and shut it down, respectively.
In the code section, it is possible to use all the MPI functions to send or receive messages in either asynchronous or synchronous mode.
The diagram in **Figure 7.5** might suggest that the MPI might allow the definition of completely symmetrical applications, since the portion of code executed in each node is the same.
A common model used in MPI is the master-worker model, where by one MPI process coordinates the execution of others that perform the same task.
Once the program has been defined in one of the available MPI implementations, it is compiled with a modified version of the compiler for the language.
The output of the compilation process can be run as a distributed application by using a specific tool provided with the MPI implementation.



12.What is a workflow? Explain the different workflow technologies?

(10M, July 2022, Dec 2018)

A workflow is the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant (a resource; human or machine) to another for action, according to a set of procedural rules.
The concept of workflow as a structured execution of tasks that have dependencies on each other has demonstrated itself to be useful for expressing many scientific experiments and

gave birth to the idea of scientific workflow.
A scientific workflow is generally expressed by a directed acyclic graph (DAG), which defines the dependencies among tasks or operations.
The nodes on the DAG represent the tasks to be executed in a workflow application; the arcs connecting the nodes identify the dependencies among tasks and the data paths that connect the tasks.
The most common dependency that is realized through a DAG is data dependency, which means that the output files of a task constitute the input files of another task.

1 Workflow technologies

Business-oriented computing workflows are defined as compositions of services.
There are specific languages and standards for the definition of workflows, such as Business Process Execution Language (BPEL).
An abstract reference model for a workflow management system, as depicted in **Figure 7.7**. Design tools allow users to visually compose a workflow application.
This specification is stored in the form of an XML document based on a specific workflow language and constitutes the input of the workflow engine, which controls the execution of the workflow by leveraging a distributed infrastructure.
The workflow engine is a client- side component that might interact directly with resources or with one or several middleware components for executing the workflow.

Some of the most relevant technologies for designing and executing workflow-based applications are:

1. **Kepler,**
2. **DAGMan,**
3. **Cloudbus Workflow Management System,** and
4. **Offspring.**

1. Kepler

Kepler is an open-source scientific workflow engine.
The system is based on the Ptolemy II system, which provides a solid platform for developing dataflow- oriented workflows.
Kepler provides a design environment based on the concept of actors, which are reusable and independent blocks of computation such as Web services, data- base calls.
The connection between actors is made with ports.
An actor consumes data from the input ports and writes data/results to the output ports. Kepler supports different models, such as synchronous and asynchronous models.
The workflow specification is expressed using a proprietary XML language.

2. DAGMan

DAGMan (Directed Acyclic Graph Manager) constitutes an extension to the Condor scheduler to handle job interdependencies.
DAGMan acts as a meta scheduler for Condor by submitting the jobs to the scheduler in the appropriate order.

The input of DAGMan is a simple text file that contains the information about the jobs, pointers to their job submission files, and the dependencies among jobs.

3. Cloudbus Workflow Management System

Cloudbus Workflow Management System (WfMS) is a middleware platform built for managing large application workflows on distributed computing platforms such as grids and clouds.

It comprises software tools that help end users compose, schedule, execute, and monitor workflow applications through a Web-based portal.

The portal provides the capability of uploading workflows or defining new ones with a graphical editor.

To execute workflows, WfMS relies on the Gridbus Broker, a grid/cloud resource broker that supports the execution of applications with quality-of- service (QoS) attributes.

4. Offspring

It offers a programming-based approach to developing workflows.

Users can develop strategies and plug them into the environment, which will execute them by leveraging a specific distribution engine.

The advantage provided by Offspring is the ability to define dynamic workflows.

This strategy represents a semi structured workflow that can change its behavior at runtime according to the execution of specific tasks.

This allows developers to dynamically control the dependencies of tasks at runtime.

Offspring supports integration with any distributed computing middleware that can manage a simple bag-of- tasks application.

Offspring allows the definition of workflows in the form of plug-ins.

13. Explain the Task programming model with a neat diagram? (8M, Dec 2018)

The Task Programming Model provides a very intuitive abstraction for quickly developing distributed applications on top of Aneka.

It provides a minimum set of APIs that are mostly centered on the Aneka.Tasks.ITask interface.

Figure 7.8 provides an overall view of the components of the Task Programming Model and their roles during application execution.

developers create distributed applications in terms of ITask instances, the collective execution of which describes a running application.

These tasks, together with all the required dependencies (data files and libraries), are grouped and managed through the Aneka Application class, which is specialized to support the execution of tasks.

Two other components, AnekaTask and TaskManager, constitute the client-side view of a task-based application. The former constitutes the runtime wrapper Aneka uses to represent a task within the middleware; the latter is the underlying component that interacts with Aneka, submits the tasks, monitors their execution, and collects the results.

In the middleware, four services coordinate their activities inorder to execute task-based applications. These are MembershipCatalogue, TaskScheduler, ExecutionService, and StorageService.

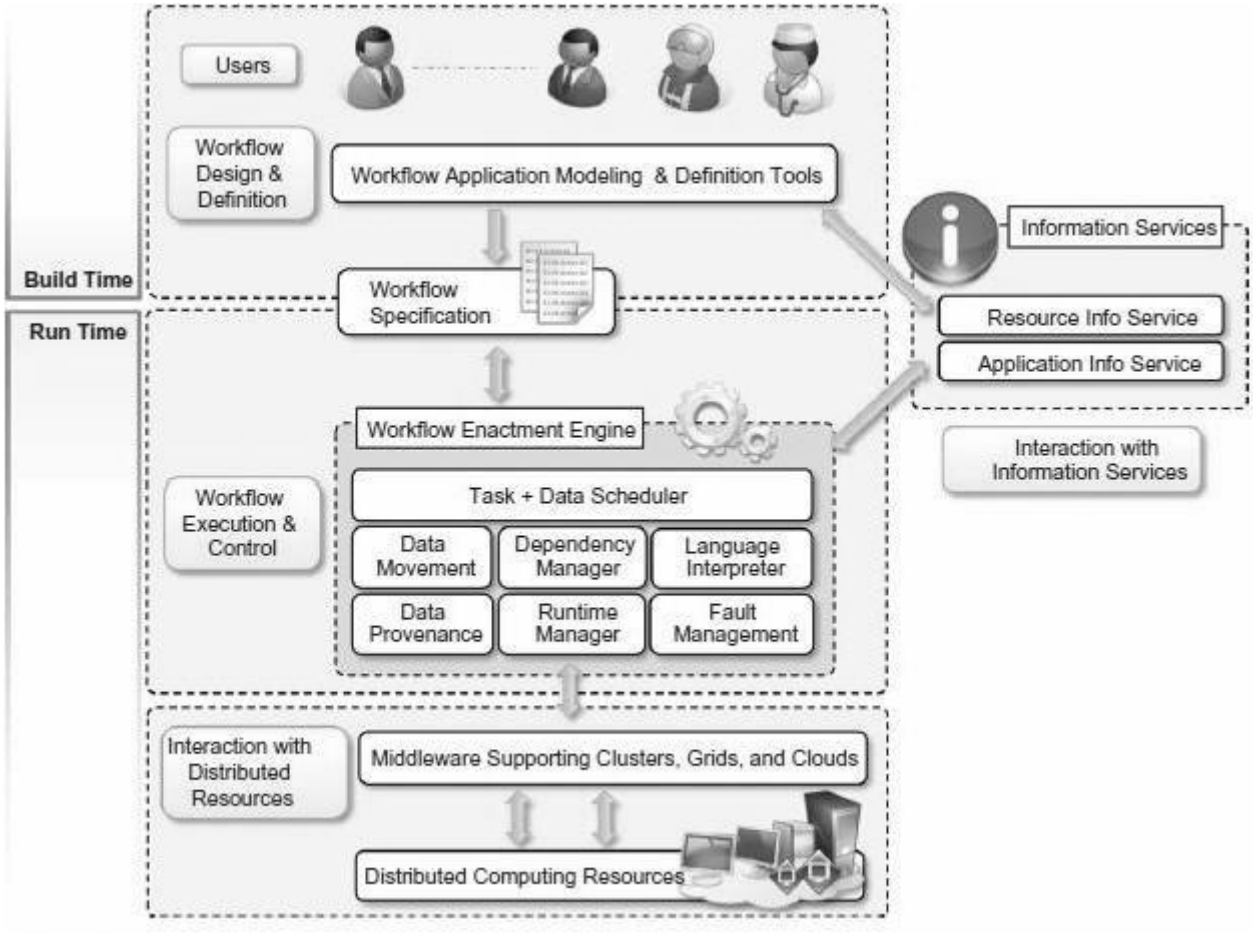
MembershipCatalogue constitutes the main access point of the cloud and acts as a service directory to locate the TaskScheduler service that is incharge of managing the execution of

task-based applications.
Its main responsibility is to allocate task instances to resources featuring the Execution Service for task execution and for monitoring task state.

14.Explain the workflow working methodology with a neat diagram?

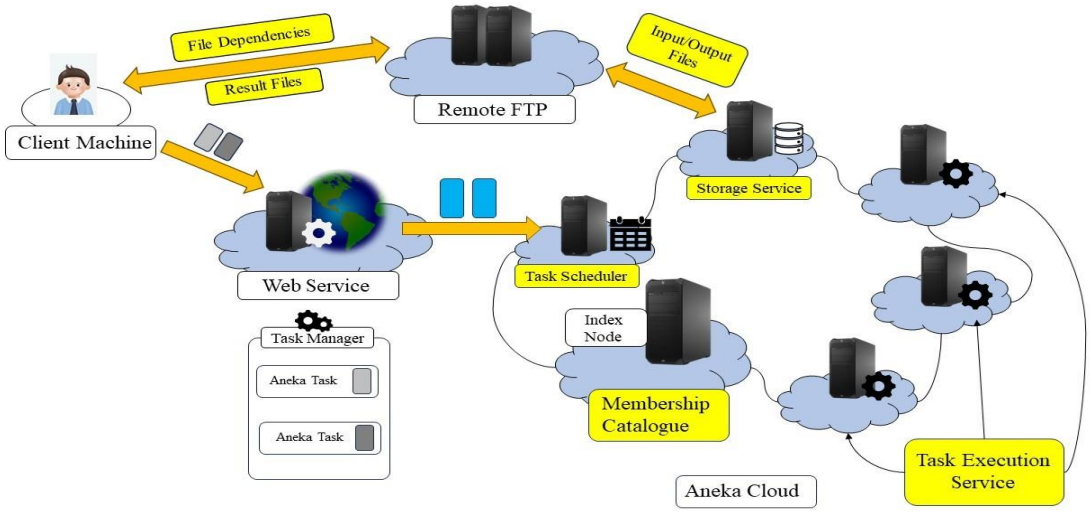
(6M, Dec 2018, Feb 2023)

A workflow is the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant (a resource; human or machine) to another for action, according to a set of procedural rules.
The concept of workflow as a structured execution of tasks that have dependencies on each other has demonstrated itself to be useful for expressing many scientific experiments and gave birth to the idea of scientific workflow.
A scientific workflow is generally expressed by a directed acyclic graph (DAG), which defines the dependencies among tasks or operations.
The nodes on the DAG represent the tasks to be executed in a workflow application; the arcs connecting the nodes identify the dependencies among tasks and the data paths that connect the tasks.
The most common dependency that is realized through a DAG is data dependency, which means that the output files of a task constitute the input files of another task.



15.Write a note on Web services integration (6M, Model Question Paper)

The task submission Web service is an additional component that can be deployed in any ASP.NET Web server and that exposes a simple interface for job submission, which is compliant with the Aneka Application Model.
The task Web service provides an interface that is more compliant with the traditional way fostered by grid computing.
The reference scenario for Web-based submission is depicted in **Figure 7.9.**
Users create a distributed application instance on the cloud, they can submit jobs querying the status of the application or a single job.
It is up to the users to then terminate the application when all the jobs are completed or abort it if there is no need to complete job execution.



Operations supported through the Web service interface are the following:
• Local file copy on the remote node
• File deletion
• Legacy application execution through the common shell services
• Parameter substitution.

16. Write a note on five different types of parameters used in parameter sweep applications? (6M,Dec 2019,Aug 2020)

- Constant parameter (PSM Single Parameter Info). This parameter identifies a specific value that is set at design time and will not change during the execution of the application.
- Range parameter (PSM Range Parameter Info). This parameter allows defining a range of allowed values, which might be integer or real. The parameter identifies a domain composed of discrete values and requires the specification of a lower bound, an upper bound, and a step for the generation of all the admissible values.
- Random parameter (PSM Random Parameter Info). This parameter allows the generation of a random value in between a given range defined by a lower and an upper bound.
- Enumeration parameter (PSM Enum Parameter Info). This parameter allows for specifying a discrete set of values of any type. It is useful to specify discrete sets that

are not based on numeric values.

- System parameter (PSM System Parameter Info). This parameter allows for mapping specific value that will be substituted at runtime while the task instance is executed on the remote node.

17. With a neat diagram explain how parameter sweep applications are executed? (8M,Aug 2021)

A parameter sweep application is executed by means of a job manager (IJob Manager), which interfaces the developer with the underlying APIs of the task model.

Figure 7.12 shows the relationships among the PSM APIs, with a specific reference to the job manager, and the task model APIs.

The implementation of IJob Manager will then create a corresponding Aneka application instance and leverage the task model API to submit all the task instances generated from the template task. The interface also exposes facilities for controlling and monitoring the execution of the parameter sweep application as well as support for registering the statistics about the application.

18. Explain Development and monitoring tools for Task progamming in aneka ? (6M,Aug 2022)

The core libraries allow developers to directly program parameter sweep applications and embed them into other applications.

Additional tools simplify design and development of parameter sweep applications. These tools are the:

Aneka Design Explorer and The **Aneka PSM Console**.

Aneka Design Explorer

The Aneka Design Explorer is an integrated visual environment for quickly prototyping parameter sweep applications, executing them, and monitoring their status.

It provides a simple wizard that helps the user visually define any aspect of parameter sweep applications, such as file dependencies and result files, parameters, and template tasks.

The environment also provides a collection of components that help users monitor application execution, aggregate statistics about application execution, gain detailed task transition monitoring, and gain extensive access to application logs.

The Aneka PSM Console

The Aneka PSM Console is a command-line utility designed to run parameter sweep applications in non- interactive mode.

The console offers a simplified interface for running applications with essential features for monitoring their execution.

19. How are workflows Managed in aneka ? (10M,Sep 2021)

Two different workflow managers can leverage Aneka for task execution:

1. **The Workflow Engine** and
2. **Offspring**.

1. The Workflow Engine

The **former leverages the task submission Web service** exposed by Aneka; the latter directly interacts with the **Aneka programming APIs.**

The Workflow Engine plug-in for Aneka which allows client applications developed with any technology and language to leverage Aneka for task execution.

2. Offspring

Figure 7.13 describes the Offspring architecture. The system is composed of two types of components:

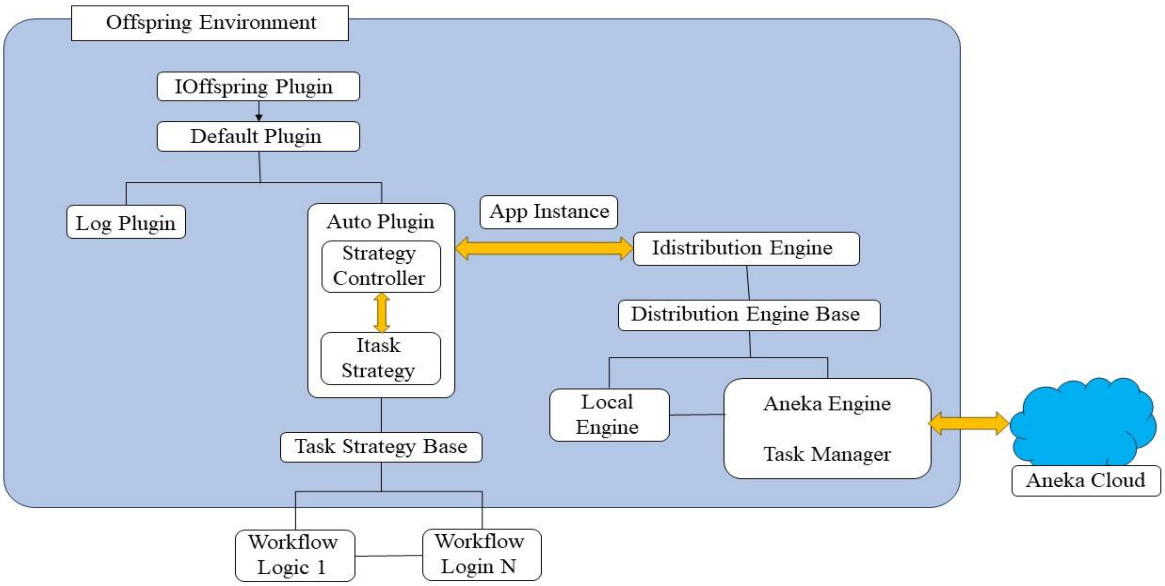
plug-ins and a **distribution engine**.

Plug-ins **are used to enrich the environment of features**; the distribution engine represents access to the distributed computing infrastructure leveraged for task execution.

Auto Plugin provides facilities for the definition of workflows in terms of strategies.

A **strategy generates the tasks that are submitted for execution and defines** the logic, in terms of sequencing, coordination, and dependencies, used to submit the task through the engine.

The Strategy Controller, decouples the strategies from the distribution engine.



The connection with Aneka is realized through the Aneka Engine, which implements the operations of IDistribution Engine for the Aneka middleware and relies on the services exposed by the task model programming APIs.wo different types of tasks can be defined: **native tasks** and **legacy tasks**. Native tasks are completely implemented in managed code.

Legacy tasks manage file dependencies and wrap all the data necessary for the execution of legacy programs

on a remote node.

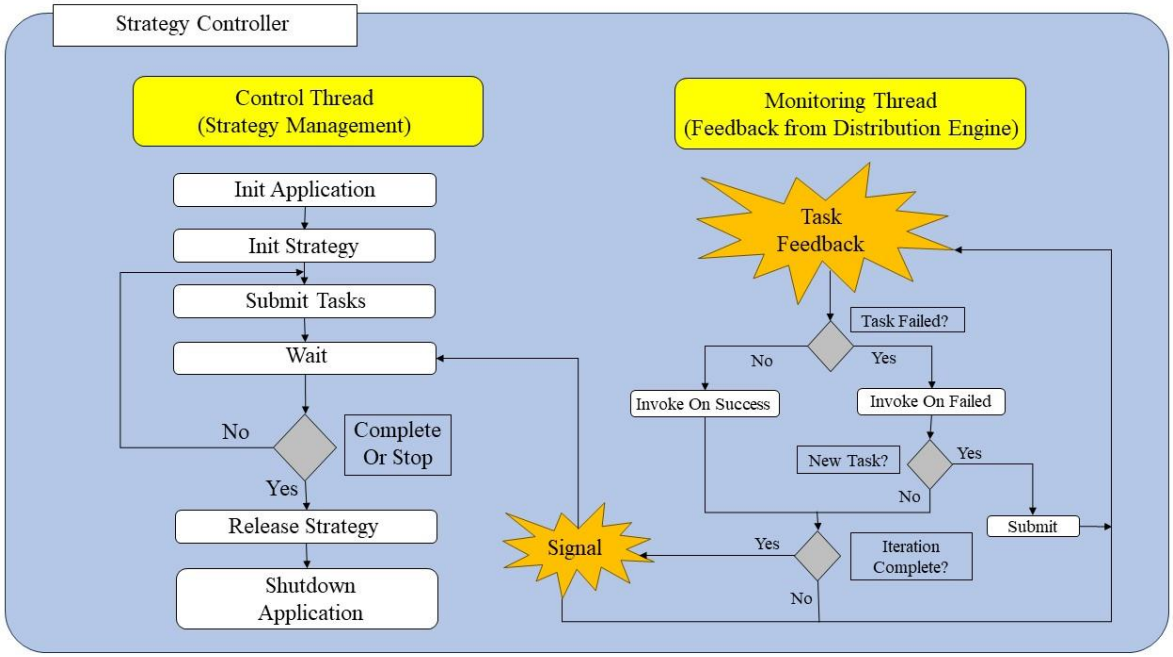
Figure7.14 describes the interactions among these components. Two main execution threads control the execution of a strategy.

A control thread manages the execution of the strategy, where as a monitoring thread collects the feedback from the distribution engine and allows for the dynamic reaction of the strategy to the execution of previously submitted tasks.

The execution of a strategy is composed of three macro steps: setup, execution, and finalization. The first step involves the setup of the strategy and the application mapping it. Correspondingly, the finalization step is in charge of releasing all the internal resources allocated by the strategy and shutting down the application.

The core of the workflow execution resides in the execution step, which is broken down into a set of iterations. During each of the iterations a collection of tasks is submitted; these tasks do not have dependencies from each other and can be executed in parallel. As soon as a task completes or fails, the strategy is queried to see whether a new set of tasks needs to be executed.

At the end of each iteration, the controller checks to see whether the strategy has completed the execution, and in this case, the finalization step is performed.



Data-Intensive Computing: MapReduce Programming
(Chapter - 8)

1.What is data-intensive computing? What are the DIC challenges? (10M,AUG 2022)

Data-intensive computing is concerned with **production, manipulation, and analysis** of **large-scale data** in the range of hundreds of **megabytes (MB) to petabytes (PB)** and **beyond**.

Characterizing data-intensive computations

Data-intensive applications dealS with huge volumes of data, also exhibit compute-intensive properties. **Figure 8.1** identifies the domain of data-intensive computing in the two upper quadrants of the graph. Data-intensive applications handle datasets on the scale of multiple terabytes and petabytes.

Challenges ahead

The huge amount of data produced, analyzed, or stored imposes requirements on the supporting infrastructures and middleware that are hardly found in the traditional solutions.
Moving terabytes of data becomes an obstacle for high-performing computations.

Data partitioning, **content replication** and **scalable algorithms** help in improving the performance.

Open challenges in data-intensive computing given by Ian Gorton et al. are:

1. **Scalable algorithms** that can search and process massive datasets.
2. New **metadata management technologies** that can handle complex, heterogeneous, and distributed data sources.
3. Advances in **high-performance computing platforms** aimed at providing a better support for accessing in-memory multiterabyte data structures.
4. High-performance, highly reliable, petascale **distributed file systems.**
5. **Data signature-generation** techniques for data reduction and rapid processing.
6. **Software mobility** that are able to move the computation to where the data are located.
7. **Interconnection architectures** that provide better support for filtering multi gigabyte datastreams coming from high-speed networks and scientific instruments.
8. **Software integration** techniques that facilitate the combination of software modules running on different platforms to quickly form analytical pipelines.

2.Historical perspective of Data-intensive computing(10M)

Data-intensive computing involves the production, management, and analysis of large volumes of data. Support for data-intensive computations is provided by harnessing storage, networking, technologies, algorithms, and infrastructure software all together.

The early age: high-speed wide-area networking

In 1989, **the first experiments in high-speed networking** as a support for remote visualization of scientific data led the way.

Two years later, the potential of using high-speed wide area networks for enabling high-speed, TCP/IP-based distributed applications was demonstrated at **Supercomputing 1991 (SC91).**

Kaiser project, leveraged the **Wide Area Large Data Object (WALDO)** system, used to provide following capabilities:

1. automatic generation of metadata;
2. automatic cataloguing of data and metadata processing data in real time;
3. facilitation of cooperative research by providing local and remote users access to data; and
4. mechanisms to incorporate data into databases and other documents.

The **Distributed Parallel Storage System (DPSS)** was developed, later used to support TerraVision, a terrain visualization application that lets users explore and navigate a tridimensional real landscape.

Clipper project, the goal of designing and implementing a collection of independent, architecturally consistent service components to support data-intensive computing. The challenges addressed by Clipper project include management of computing resources, generation or consumption of high-rate and high-volume data flows, human interaction management, and aggregation of resources.

3. What are Data grids? Explain the characteristics of data grid

(5M,AUG 2022)

Huge computational power and storage facilities could be obtained by harnessing heterogeneous resources across different administrative domains.

Data grids emerge as infrastructures that support data-intensive computing.

A data grid provides services that help users discover, transfer, and manipulate large datasets stored in distributed repositories as well as create and manage copies of them.

Data grids offer two main functionalities:

- high-performance and reliable file transfer for moving large amounts of data, and
- scalable replica discovery and management mechanisms.

Data grids mostly provide storage and dataset management facilities as support for scientific experiments that produce huge volumes of data.

Datasets are replicated by infrastructure to provide better availability.

Data grids have their own characteristics and introduce new challenges:

a. **Massive datasets**. The size of datasets can easily be on the scale of gigabytes, terabytes, and beyond. It is therefore necessary to minimize latencies during bulk transfers, replicate content with appropriate strategies, and manage storage resources.
b. **Shared data collections.** Resource sharing includes distributed collections of data. For example, repositories can be used to both store and read data.
c. **Unified namespace.** Data grids impose a unified logical namespace where to locate data collections and resources. Every data element has a single logical name, which is eventually mapped to different physical filenames for the purpose of replication and accessibility.
d. **Access restrictions.** Even though one of the purposes of data grids is to facilitate sharing of results and data for experiments, some users might want to ensure confidentiality for their data and restrict access to them to their collaborators. Authentication and

authorization in data grids involve both coarse-grained and fine-grained access control over shared data collections.

As a result, several scientific research fields, including high-energy physics, biology, and astronomy, leverage data grids.

4.What are data clouds? How Cloud technologies support data-intensive computing ?
(4M,Model question paper)

- By providing a large amount of compute instances on demand, which can be used to process and analyze large datasets in parallel.
- By providing a storage system optimized for keeping large blobs of data and other distributed data store architectures.
- By providing frameworks and programming APIs optimized for the processing and management of large amounts of data.

A data cloud is a combination of these components.

Ex 1: MapReduce framework, which provides the best performance for leveraging the Google File System on top of Google's large computing infrastructure.

Ex 2: Hadoop system, the most mature, large, and open-source data cloud. It consists of the Hadoop Distributed File System (HDFS) and Hadoop's implementation of MapReduce.

Ex 3: Sector, consists of the Sector Distributed File System (SDFS) and a compute service called Sphere that allows users to execute arbitrary user-defined functions (UDFs) over the data managed by SDFS.

Ex 4: Greenplum uses a shared-nothing massively parallel processing (MPP) architecture based on commodity hardware.

5.Write a note on Technologies for data-intensive computing

/ Write a note on High-performance distributed file systems and storage clouds/

Write a note on the any 3 file systems used in distributed systems? (10MJuly 2021)

Distributed file systems constitute the primary support for **data management**. They provide an interface whereby to store information in the form of files and later access them for read and write operations.

a.Lustre. The Lustre file system is a **massively parallel distributed file system** that covers the needs of a small workgroup of clusters to a large-scale computing cluster. The file system is used by several of the Top 500 **supercomputing systems.**

Lustre is designed to provide access to petabytes (PBs) of storage to serve thousands of clients with an I/O throughput of hundreds of gigabytes per second (GB/s). The **system is composed of a metadata server that contains the metadata about the file system and a collection of object storage servers that are in charge of providing storage.**

b.IBM General Parallel File System (GPFS). GPFS is the **high-performance** distributed file system developed by IBM that provides support **for the RS/6000 supercomputer and Linux computing clusters**. GPFS is a multiplatform distributed file system built over several years of academic research and provides **advanced recovery mechanisms**. GPFS is built on the concept of shared disks, in which a collection of disks is attached to the file system nodes by means of some switching fabric. The file system makes this infrastructure

transparent to users and stripes large files over the disk array by replicating portions of the file to ensure high availability.

c.Google File System (GFS). GFS is the storage infrastructure that supports the execution of distributed applications in Google's computing cloud.

GFS is designed with the following assumptions:

1. The system is built on top of commodity hardware that often fails.
2. The system stores a modest number of large files; multi-GB files are common and should be treated efficiently, and small files must be supported, but there is no need to optimize for that.
3. The workloads primarily consist of two kinds of reads: large streaming reads and small random reads.
4. The workloads also have many large, sequential writes that append data to files.
5. High-sustained bandwidth is more important than low latency.

The architecture of the file system is organized into a single master, which contains the metadata of the entire file system, and a collection of chunk servers, which provide storage space. From a logical point of view the system is composed of a collection of software daemons, which implement either the master server or the
chunk server.

d.**Sector**. Sector is the storage cloud that supports the execution of data-intensive applications defined according to the Sphere framework. It is a user space file system that can be deployed on commodity hardware across a wide-area network. The system's architecture is composed of four nodes: a security server, one or more master nodes, slave nodes, and client machines. The security server maintains all the information about access control policies for user and files, whereas master servers coordinate and serve the I/O requests of clients, which ultimately interact with slave nodes to access files. The protocol used to exchange data with slave nodes is UDT, which is a lightweight connection-oriented protocol.

e. Amazon Simple Storage Service (S3). Amazon S3 is the online storage service provided by Amazon.

The system offers a flat storage space organized into buckets, which are attached to an Amazon Web Services (AWS) account. Each bucket can store multiple objects, each identified by a unique key. Objects are identified by unique URLs and exposed through HTTP, thus allowing very simple get-put semantics.

6. Write a note on NoSQL database systems? (10M,July 2021)

The term **Not Only SQL (NoSQL)** was coined in 1998 to identify a set of UNIX shell scripts and commands to operate on text files containing the actual data.

NoSQL cannot be considered a relational database, it is a collection of scripts that allow users to manage most of the simplest and more common database tasks by using text files as information stores.

Two main factors have determined the growth of the NoSQL:

- simple data models are enough to represent the information used by applications, and
- the quantity of information contained in unstructured formats has grown.

Let us now examine some prominent implementations that support data-intensive

applications.

Apache CouchDB and MongoDB.

Apache CouchDB and MongoDB are two examples of document stores. Both provide a schema-less store whereby the primary objects are documents organized into a collection of key-value fields. The value of each field can be of type string, integer, float, date, or an array of values.

The databases expose a RESTful interface and represent data in JSON format. Both allow querying and indexing data by using the MapReduce programming model, expose JavaScript as a base language for data querying and manipulation rather than SQL, and support large files as documents.

a. Amazon Dynamo.

The main goal of Dynamo is to provide an incrementally scalable and highly available storage system. This goal helps in achieving reliability at a massive scale, where thousands of servers and network components build an infrastructure serving 10 million requests per day. Dynamo provides a simplified interface based on get/put semantics, where objects are stored and retrieved with a unique identifier (key).

The architecture of the Dynamo system, shown in **Figure 8.3**, is composed of a collection of storage peers organized in a ring that shares the key space for a given application. The key space is partitioned among the storage peers, and the keys are replicated across the ring, avoiding adjacent peers. Each peer is configured with access to a local storage facility where original objects and replicas are stored.

Each node provides facilities for distributing the updates among the rings and to detect failures and unreachable nodes.

b. Google Bigtable.

Bigtable provides storage support for several Google applications that expose different types of workload: from throughput-oriented batch-processing jobs to latency-sensitive serving of data to end users.

Bigtable's key design goals are wide applicability, scalability, high performance, and high availability. To achieve these goals, Bigtable organizes the data storage in tables of which the rows are distributed over the distributed file system supporting the middleware, which is the Google File System.

7. Explain the Amazon Dynamo architecture with a neat diagram? (10M, Dec 2018)

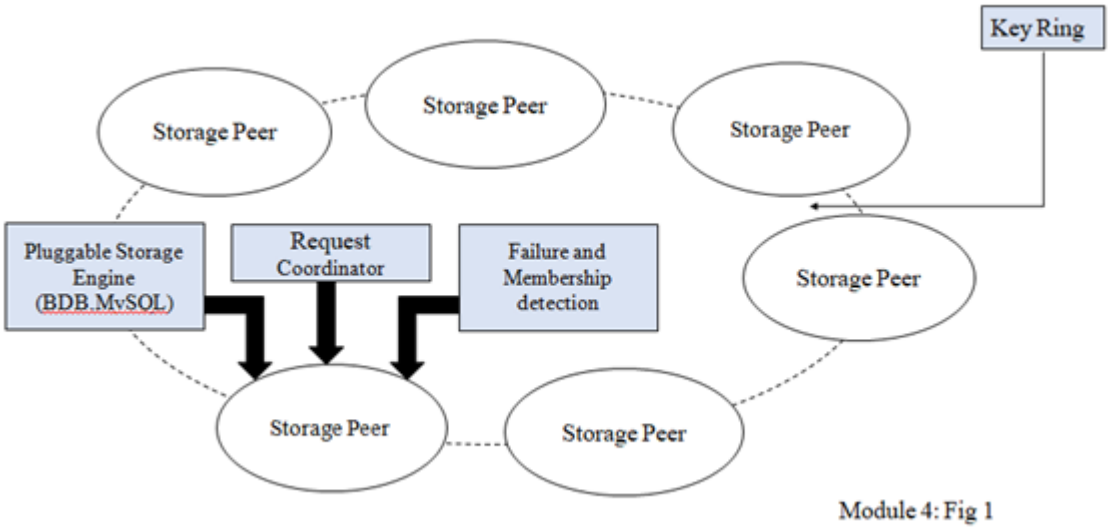
Distributed key-value store supporting the management of information of several of the business services offered by Amazon Inc.

The main goal of Dynamo is to provide an incrementally scalable and highly available storage system.

This goal helps in achieving reliability at a massive scale, where thousands of servers and network components build an infrastructure serving 10 million requests per day

The architecture of the Dynamo system, shown in **Figure 8.3**, is composed of a collection of storage peers organized in a ring that shares the key space for a given application. The key space is partitioned among the storage peers, and the keys are replicated across the ring, avoiding adjacent peers. Each peer is configured with access to a local storage facility where

original objects and replicas are stored.
Each node provides facilities for distributing the updates among the rings and to detect failures and unreachable nodes.



Module 4: Fig 1

8. Write a note on

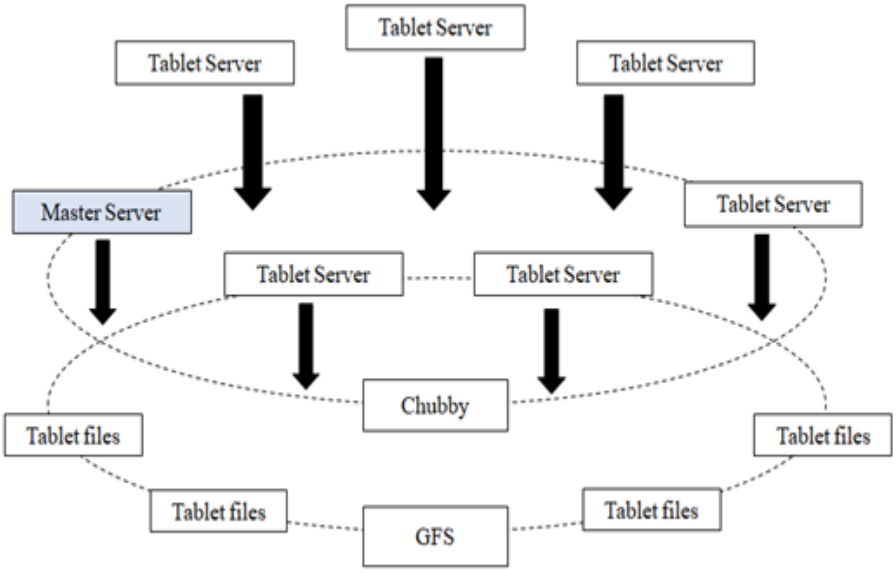
a.Apache Cassandra b. Hbase (5M,Aug 2022)

The system is designed to avoid a single point of failure and offer a highly reliable service. Cassandra was initially developed by Facebook; now it is part of the Apache incubator initiative. Currently, it provides storage support for several very large Web applications such as Facebook itself, Digg, and Twitter.

The data model exposed by Cassandra is based on the concept of a table that is implemented as a distributed multidimensional map indexed by a key. The value corresponding to a key is a highly structured object and constitutes the row of a table. Cassandra organizes the row of a table into columns, and sets of columns can be grouped into column families. The APIs provided by the system to access and manipulate the data are very simple: insertion, retrieval, and deletion. The insertion is performed at the row level; retrieval and deletion can operate at the column level.

a. Hadoop HBase.

HBase is designed by taking inspiration from Google Bigtable; its main goal is to offer real-time read/write operations for tables with billions of rows and millions of columns by leveraging clusters of commodity hardware. The internal architecture and logic model of HBase is very similar to Google Bigtable, and the entire system is backed by the Hadoop Distributed File System (HDFS).



Module 4: Fig 2

9. With a neat block diagram explain MapReduce programming model and its workflow? (10M, June 2020,July 2021,AUG 2022)

MapReduce expresses the computational logic of an application in two simple functions: map and reduce.

Data transfer and management are completely handled by the distributed storage infrastructure (i.e., the Google File System), which is in charge of providing access to data, replicating files, and eventually moving them where needed.

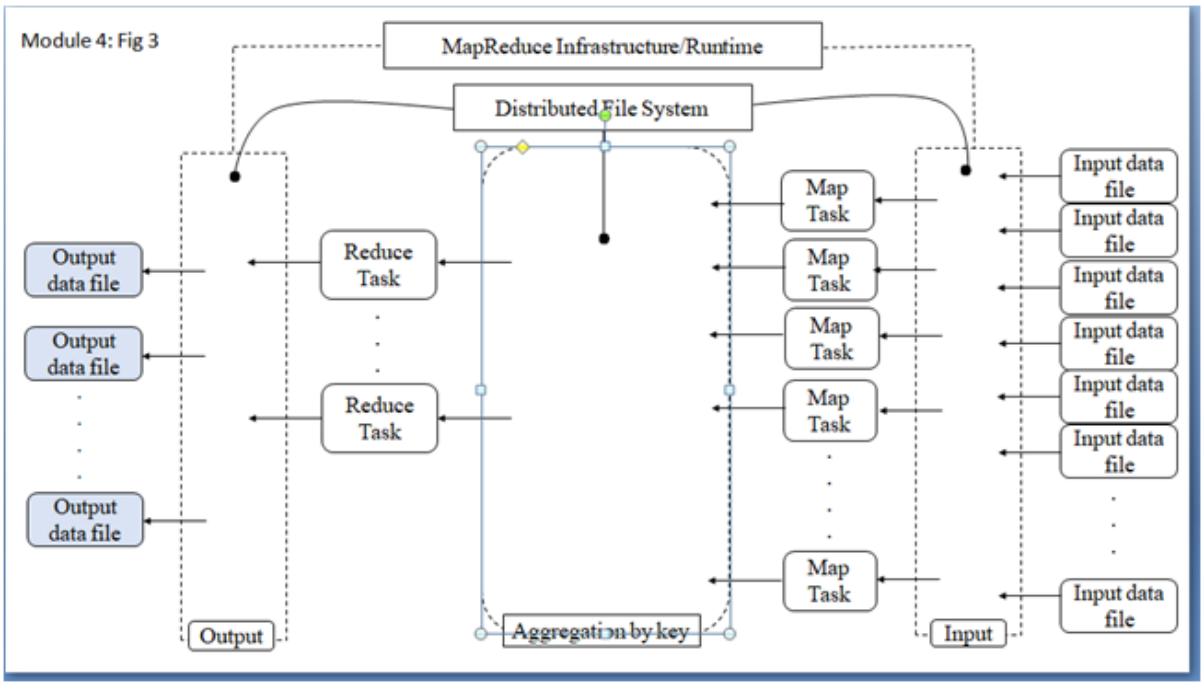
the MapReduce model is expressed in the form of the two functions, which are defined as follows:

$$map\ (k1, v1) \rightarrow list(k2, v2)$$
$$reduce(k2, list(v2)) \rightarrow list(v2)$$

The map function reads a key-value pair and produces a list of key-value pairs of different types. The reduce function reads a pair composed of a key and a list of values and produces a list of values of the same type. The types (k1,v1,k2,kv2) used in the expression of the two functions provide hints as to how these two functions are connected and are executed to carry out the computation of a MapReduce job:

Figure 8.5 depicts a reference workflow characterizing MapReduce computations. As shown, the user submits a collection of files that are expressed in the form of a list of < k1,v1 > pairs and specifies the map and reduce functions. These files are entered into the distributed file system that supports MapReduce and, if necessary, partitioned in order to be the input of map tasks. Map tasks generate intermediate files that store collections of

< k2, list(v2) > pairs, and these files are saved into the distributed file system. These files constitute the input of reduce tasks, which finally produce output files in the form of list(v2).



The computation model expressed by MapReduce is very straightforward and allows greater productivity for people who have to code the algorithms for processing huge quantities of data.

In general, any computation that can be expressed in the form of two major stages can be represented in terms of MapReduce computation.

These stages are:

1. **Analysis.** This phase operates directly on the data input file and corresponds to the operation performed by the map task. Moreover, the computation at this stage is expected to be embarrassingly parallel, since map tasks are executed without any sequencing or ordering.
2. **Aggregation.** This phase operates on the intermediate results and is characterized by operations that are aimed at aggregating, summing, and/or elaborating the data obtained at the previous stage to present the data in their final form. This is the task performed by the reduce function.

Figure 8.6 gives a more complete overview of a MapReduce infrastructure, according to the implementation proposed by Google.

As depicted, the user submits the execution of MapReduce jobs by using the client libraries that are in charge of submitting the input data files, registering the map and reduce functions, and returning control to the user once the job is completed. A generic distributed infrastructure (i.e., a cluster) equipped with job-scheduling capabilities and distributed storage can be used to run MapReduce applications.

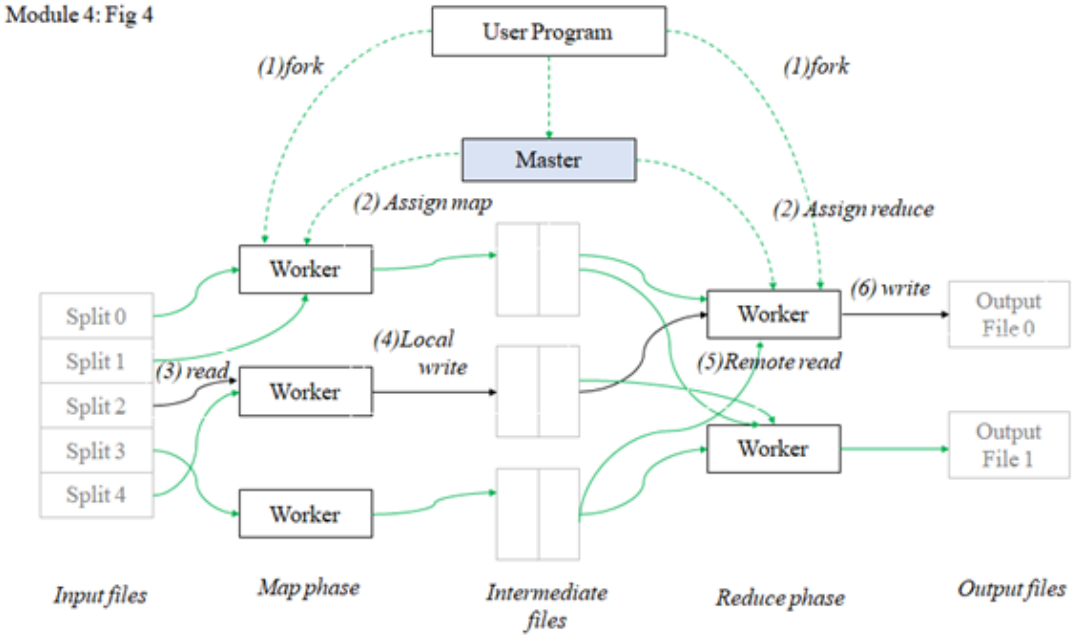
Two different kinds of processes are run on the distributed infrastructure: a master process and a worker process.

The master process is in charge of controlling the execution of map and reduce tasks, partitioning, and reorganizing the intermediate output produced by the map task in order to feed the reduce tasks.

The master process generates the map tasks and assigns input splits to each of them by

balancing the load.

The worker processes are used to host the execution of map and reduce tasks and provide basic I/O facilities that are used to interface the map and reduce tasks with input and output files. Worker processes have input and output buffers that are used to optimize the performance of map and reduce tasks. In particular, output buffers for map tasks are periodically dumped to disk to create intermediate files. Intermediate files are partitioned using a user-defined function to evenly split the output of map tasks.



10. Write a note on the following **(10M,AUG 2022)**

A. **Pig.**

B. **Hive.**

C. **Map-Reduce-Merge.**

D. **Twister.**

E. **Hadoop**

A. Pig.

Pig is a platform that allows the analysis of large datasets. Developed as an Apache project, Pig consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The Pig infrastructure's layer consists of a compiler for a high-level language that produces a sequence of MapReduce jobs that can be run on top of distributed infrastructures.

B. Hive.

Hive is another Apache initiative that provides a data warehouse infrastructure on top of Hadoop MapReduce. It provides tools for easy data summarization, ad hoc queries, and analysis of large datasets stored in Hadoop MapReduce files.

Hive's major advantages reside in the ability to scale out, since it is based on the Hadoop framework, and in the ability to provide a data warehouse infrastructure in environments where there is already a Hadoop system running.

C. Map-Reduce-Merge.

Map-Reduce-Merge is an extension of the MapReduce model, introducing a third phase to the standard MapReduce pipeline—the Merge phase—that allows efficiently merging data already partitioned and sorted (or hashed) by map and reduce modules. The Map-Reduce-Merge framework simplifies the management of heterogeneous related datasets and provides an abstraction able to express the common relational algebra operators as well as several join algorithms.

D. Twister.

Twister is an extension of the MapReduce model that allows the creation of iterative executions of MapReduce jobs. With respect to the normal MapReduce pipeline, the model proposed by Twister proposes the following extensions:

1. Configure Map
2. Configure Reduce
3. While Condition Holds True Do
    a. Run MapReduce
    b. Apply Combine Operation to Result
    c. Update Condition
4. Close

Twister provides additional features such as the ability for map and reduce tasks to refer to static and in- memory data; the introduction of an additional phase called combine, run at the end of the MapReduce job, that aggregates the output together.

E.Hadoop.

Apache Hadoop is a collection of software projects for reliable and scalable distributed computing. The initiative consists of mostly two projects: Hadoop Distributed File System (HDFS) and Hadoop **MapReduce**. The former is an implementation of the Google File System; the latter provides the same features and abstractions as Google MapReduce.

11.Explain the Aneka's MapReduce infrastructure /programming model?

(10M,Dec 2020,Jan 2021)

The MapReduce Programming Model defines the abstractions and runtime support for developing MapReduce applications on top of Aneka.

Figure 8.7 provides an overview of the infrastructure supporting MapReduce in Aneka.

The application instance is specialized, with components that identify the map and reduce functions to use.

These functions are expressed in terms of **Mapper and Reducer classes** that are extended from the Aneka MapReduce APIs.

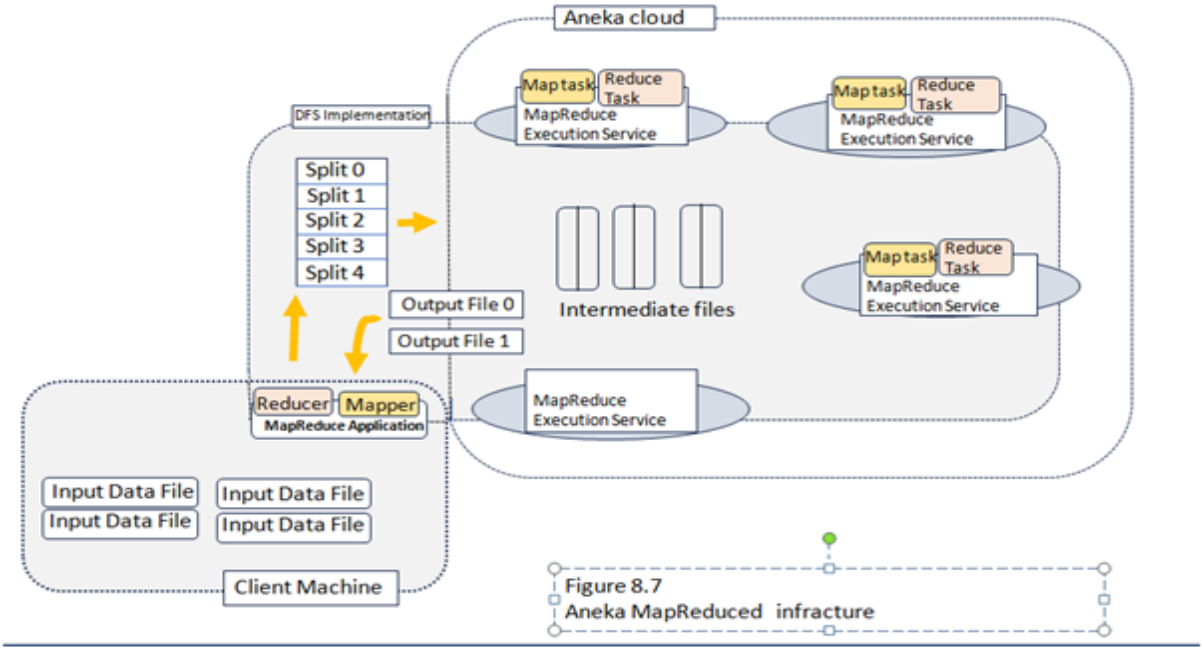


Figure 8.7
Aneka MapReduced infracture

The runtime support is composed of **three** main elements:

1. **MapReduce Scheduling Service**, which plays the role of the master process in the Google and Hadoop implementation.
2. **MapReduce Execution Service**, which plays the role of the worker process in the Google and Hadoop implementation.
3. A specialized **distributed file system** that is used to move data files.

Client components, namely the MapReduce Application, are used to submit the execution of a MapReduce job, upload data files, and monitor it.

The management of data files is transparent: local data files are automatically uploaded to Aneka, and output files are automatically downloaded to the client machine if requested.

In the following sections, we introduce these major components and describe how they collaborate to execute MapReduce jobs.

12. Explain the working of Google Bigtable infrastructure with a neat diagram? Explain the advantages (10M,AUG 2020,July 2021)

Bigtable provides storage support for several Google applications that expose different types of workload: from throughput-oriented batch-processing jobs to latency-sensitive serving of data to end users.

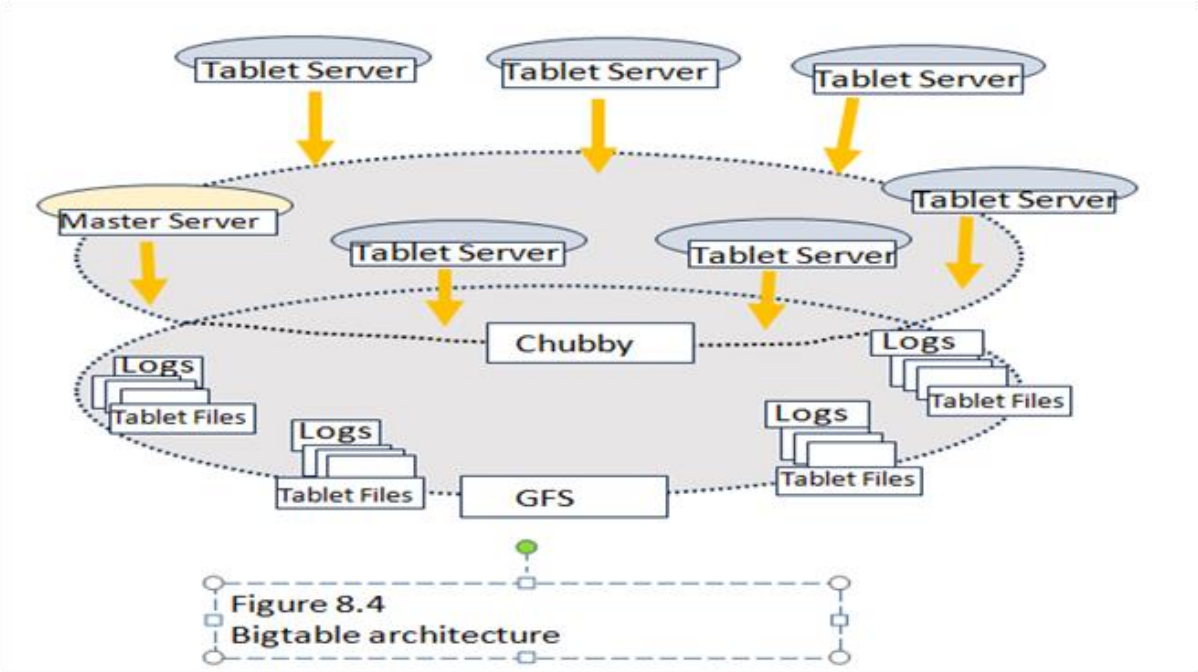
Bigtable's key design goals are wide applicability, scalability, high performance, and high availability. To achieve these goals, Bigtable organizes the data storage in tables of which the

rows are distributed over the distributed file system supporting the middleware, which is the Google File System.

From a logical point of view, a table is a multidimensional sorted map indexed by a key that is represented by a string of arbitrary length. A table is organized into rows and columns; columns can be grouped in column family, which allow for specific optimization for better access control, the storage and the indexing of data.

=Bigtable APIs also allow more complex operations such as single row transactions and advanced data manipulation.

Figure 8.4 gives an overview of the infrastructure that enables Bigtable.



- The service is the result of a collection of processes that coexist with other processes in a cluster-based environment.
- Bigtable identifies two kinds of processes: master processes and tablet server processes.
- A tablet server is responsible for serving the requests for a given tablet that is a contiguous partition of rows of a table.
- Each server can manage multiple tablets (commonly from 10 to 1,000). The master server is responsible for keeping track of the status of the tablet servers and of the allocation of tablets to tablet servers.
- The server constantly monitors the tablet servers to check whether they are alive, and in case they are not reachable, the allocated tablets are reassigned and eventually partitioned to other servers.
- Chubby
- a **distributed, highly available, and persistent lock service—supports** the activity of the master and tablet servers.
- **System monitoring and data access** are **filtered through Chubby**, which is also responsible for managing replicas and providing consistency among them.
- At **the very bottom layer, the data are stored in the Google File System in** the

form of files, and all the update operations are logged into the file for the easy recovery of data in case of failures or when tablets need to be reassigned to other servers.

- It serves as a storage back-end for 60 applications (such as Google Personalized Search, Google Anlytics, Google Finance, and Google Earth) and manages petabytes of data.

13. Explain the Job and Task Scheduling and Execution Service?

(10M,Model question Paper)

The scheduling of jobs and tasks is the responsibility of the MapReduce Scheduling Service, which covers the same role as the master process in the Google MapReduce implementation. The architecture of the Scheduling Service is organized into two major components: the MapReduceSchedulerService and the MapReduceScheduler.

Main role of the service wrapper is to translate messages coming from the Aneka runtime or the client applications into calls or events directed to the scheduler component, and vice versa. The relationship of the two components is depicted in **Figure 8.9**.

The core functionalities for job and task scheduling are implemented in the MapReduceScheduler class. The scheduler manages multiple queues for several operations, such as uploading input files into the distributed file system; initializing jobs before scheduling; scheduling map and reduce tasks; keeping track of unreachable nodes; resubmitting failed tasks; and reporting execution statistics.

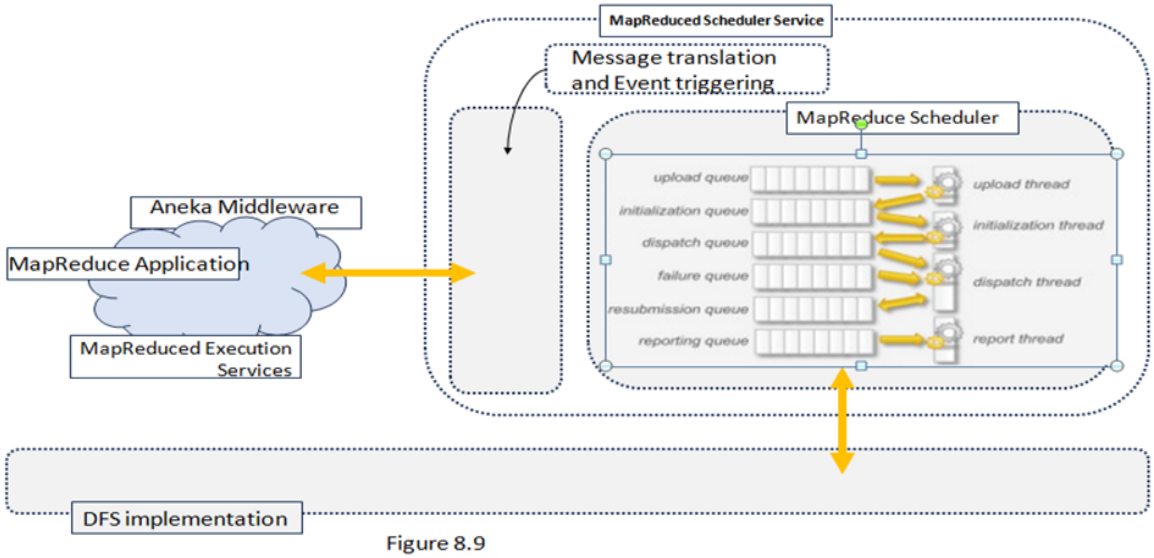


Figure 8.9
MapReduced Scheduling Service architecture

Task Execution. The execution of tasks is controlled by the MapReduce Execution Service. This component plays the role of the worker process in the Google MapReduce implementation. The service manages the execution of map and reduce tasks and performs other operations, such as sorting and merging intermediate files. The service is internally organized, as described in **Figure 8.10**.

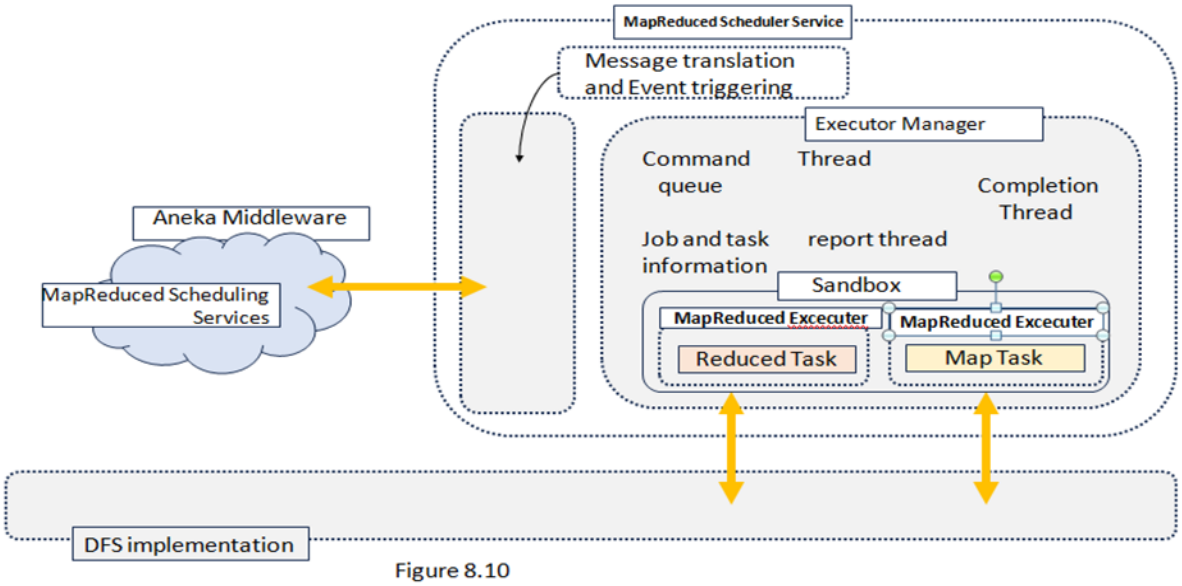


Figure 8.10
MapReduced Execution Service architecture

There are three major components that coordinate together for executing tasks

1. MapReduce- SchedulerService,
2. ExecutorManager, and
3. MapReduceExecutor.

The MapReduceSchedulerService interfaces the ExecutorManager with the Aneka middleware; the ExecutorManager is in charge of keeping track of the tasks being executed by demanding the specific execution of a task to the MapReduceExecutor and of sending the statistics about the execution back to the Scheduler Service

14. Write a note on Distributed file system support on Aneka cloud? (10M, Aug 2021)

Aneka supports the MapReduce model that uses a distributed file system implementation.

Distributed file system implementations guarantee high availability and better efficiency by means of replication and distribution.

the original MapReduce implementation assumes the existence of a distributed and reliable storage; hence, the use of a distributed file system for implementing the storage layer is natural.

Aneka provides the capability of interfacing with different storage implementations and it maintains the same flexibility for the integration of a distributed file system.

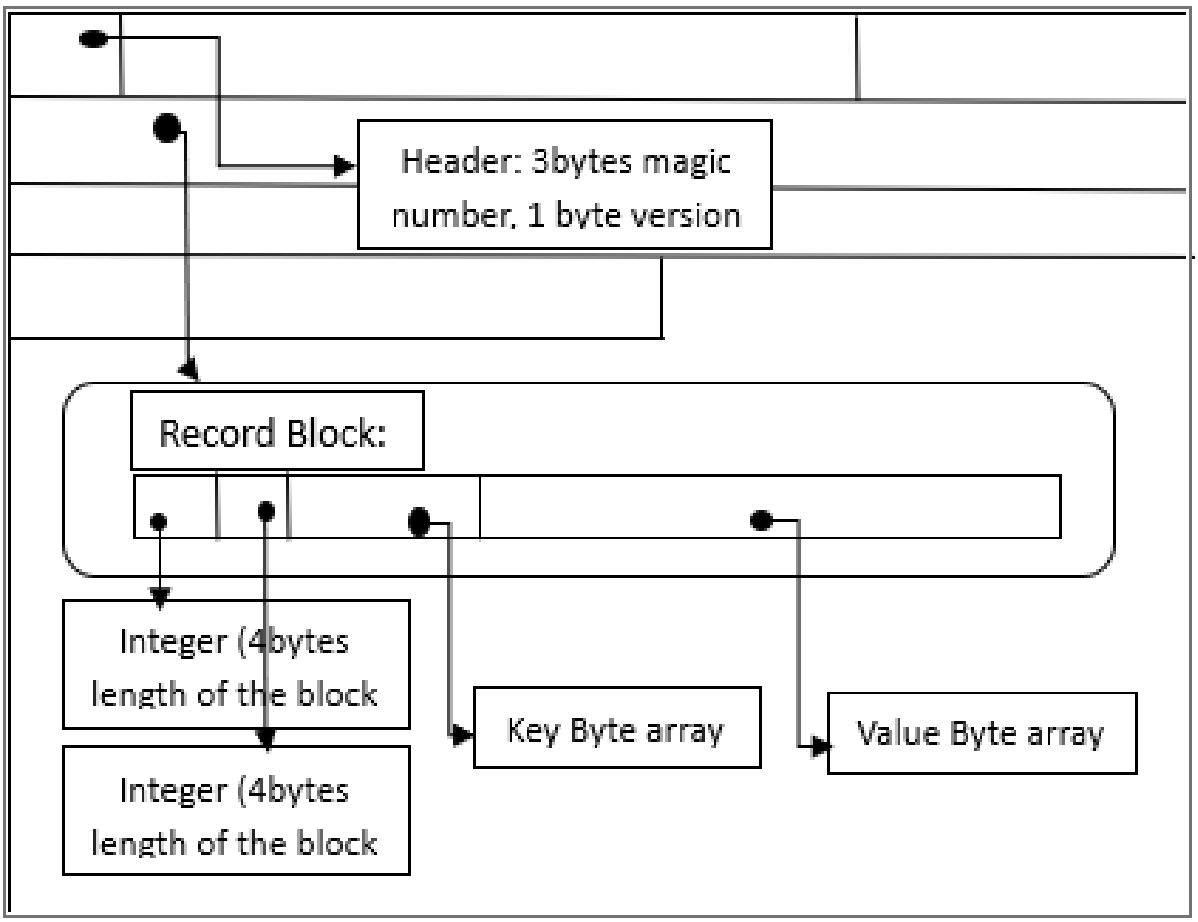
The level of integration required by MapReduce requires the ability to perform the following tasks:

- Retrieving the location of files and file chunks
- Accessing a file by means of a stream

The first operation is useful to the scheduler for optimizing the scheduling of map and reduce tasks according to the location of data; the second operation is required for the usual I/O operations to and from data files.

On top of these low-level interfaces, MapReduce programming model offers classes to read from and write to files in a sequential manner. These are classes **SeqReader** and **SeqWriter**. They provide sequential access for reading and writing key-value pairs, and they expect specific file format, which is described in **Figure 8.11**.

An Aneka MapReduce file is composed of a header, used to identify the file, and a sequence of record blocks, each storing a key-value pair. The header is composed of 4 bytes: the first 3 bytes represent the character sequence SEQ and the fourth byte identifies the version of the file. The record block is composed as follows: the first 8 bytes are used to store two integers representing the length of the rest of the block and the length of the key section, which is immediately following. The remaining part of the block stores the data of the value component of the pair. The SeqReader and SeqWriter classes are designed to read and write files in this format by transparently handling the file format information and translating key and value instances to and from their binary representation.


Header: 3bytes magic number, 1 byte version
Record Block:
Integer (4bytes length of the block
Integer (4bytes length of the block
Key Byte array
Value Byte array

Module - 5
(Chapter -9 AND 10)

1. What are Amazon web services? Explain any 4 compute services available in the AWS ecosystem?

(10M,Aug 2022)

Amazon Web Services (AWS) is a platform that allows the development of flexible applications by providing solutions for elastic infrastructure scalability, messaging, and data storage.

The platform is accessible through SOAP or RESTful Web service interfaces and provides a Web-based console where users can handle administration and monitoring of the resources required, as well as their expenses computed on a pay-as-you-go basis.

Figure 9.1 shows all the services available in the AWS ecosystem. At the base of the solution stack are services that provide raw compute and raw storage: Amazon Elastic Compute (EC2) and Amazon Simple Storage Service (S3).

Amazon EC2

EC2 instances are executed within a virtual environment. The EC2 environment is in charge of allocating addresses, attaching storage volumes, and configuring security in terms of access control and network connectivity.

It is possible to associate an Elastic IP to each instance. Elastic IPs allow instances running in EC2 to act as servers reachable from the Internet.

EC2 instances are given domain name in the form ec2-xxx-xxx-xxx.compute-x.amazonaws.com,

where xxx-xxx-xxx normally represents the four parts of the external IP address separated by a dash, and compute-x gives information about the availability zone where instances are deployed.

Currently, there are five availability zones: two in the United States (Virginia and Northern California), one in Europe (Ireland), and two in Asia Pacific (Singapore and Tokyo).

1. Advanced compute services

AWS CloudFormation constitutes an extension of the simple deployment model that characterizes EC2 instances. CloudFormation introduces the concepts of templates, which are JSON formatted text files that describe the resources needed to run an application or a service in EC2 together.

Templates provide a simple and declarative way to build complex systems and integrate EC2 instances with other AWS services such as S3, SimpleDB, SQS, SNS, Route 53, Elastic Beanstalk, and others.

AWS Elastic Beanstalk constitutes a simple and easy way to package applications and deploy them on the AWS Cloud. This service simplifies the process of provisioning instances and deploying application code and provides appropriate access to them.

Currently, this service is available for Web applications developed with the Java/Tomcat

technology stack. Beanstalk simplifies tedious tasks without removing the user's capability of accessing—and taking over control of—the underlying EC2 instances.

Amazon Elastic MapReduce provides AWS users with a cloud computing platform for MapReduce applications. It utilizes Hadoop as the MapReduce engine, deployed on a virtual infrastructure composed of EC2 instances, and uses Amazon S3 for storage needs.

Elastic MapReduce introduces elasticity and allows users to dynamically size the Hadoop cluster according to their needs, as well as select the appropriate configuration of EC2 instances to compose the cluster.

2.Explain the EC2 environment and its features

(10M, Dec 2021)

- The fundamental service in this space is Amazon EC2, which delivers an IaaS solution that has served as a reference model for several offerings from other vendors in the same market segment.
- Amazon EC2 allows deploying servers in the form of virtual machines created as instances of a specific image.
- Images come with a preinstalled operating system and a software stack, and instances can be configured for memory, number of processors, and storage

.

- Users are provided with credentials to remotely access the instance and further configure or install software if needed.

  EC2 consists of following services
- Amazon machine image EC2 instances EC2 environment Advanced compute services
- Amazon Machine Images (AMIs) are templates from which it is possible to create a virtual machine.
- They are stored in **Amazon S3 and identified by a unique identifier in the form of ami-xxxxxx and a manifest XML file**.
- AMI contains physical file system layout with a predefined operating system installed as Amazon RamdiskImage (ARI, id: ari-yyyyyy) and the Amazon Kernel Image (AKI, id: aki-zzzzzz)
- A common practice is to prepare new AMIs to create an instance from a preexisting AMI, log into it once it is booted and running, and install all the software needed.
- Using the tools provided by Amazon, **we can convert the instance into a new image. Once an AMI is created, it is stored in an in an S3 bucket** and the user can decide whether to make it available to other users or keep it for personal use.

  EC2 instances
- **EC2 instances represent virtual machines.**
- **They are created using AMI as templates, which are specialized by selecting the number of cores, their computing power, and the installed memory.**
- **The processing power is expressed in terms of virtual cores and EC2 Compute**

Units (ECUs).

- **Use of ECUs helps give users a consistent view of the performance offered by EC2 instances.**
- **One ECU is defined as giving the same performance as a 1.0 - 1.2 GHz 2007 Opteron or 2007 Xeon processor.**

EC2 Environment

- EC2 instances represent virtual machines.
- They are created using AMI as templates, which are specialized by **selecting the number of cores, their computing power, and the installed memory**.
- The processing power is expressed in terms of virtual cores and **EC2 Compute Units (ECUs).**
- Use of ECUs helps give users a consistent view of the performance offered by EC2 instances.
- One ECU is defined as giving the same performance as a 1.0 - 1.2 GHz 2007 Opteron or 2007 Xeon processor.

3.Explain six major configurations for EC2 instances.? (6M,Dec 2018)

● Standard instances. offers a set of configurations that are suitable for most applications. EC2 provides three different categories of increasing computing power, storage, and memory.

● Micro instances.

suitable for those applications that consume limited amount of computing power and memory and need bursts in CPU cycles to process surges in workload.

● High-memory instances.

targets applications that need to process huge workloads and require large amounts of memory. Three-tier Web applications characterized by high traffic are the target profile.

● High-CPU instances.

targets compute-intensive applications.

- **Cluster Compute instances.**

used to provide virtual cluster services. Instances in this category are characterized by high CPU compute power and large memory and an extremely high I/O and network performance, which makes it suitable for HPC applications.

- **Cluster GPU instances. provides instances featuring graphic processing units (GPUs) and high compute** power, large memory, and extremely high I/O and network performance. This class is particularly suited for cluster applications that perform heavy graphic computations.

4. Write a note on the following amazon storage services?

i)S3 key concepts ii)Amazon elastic block store 3.Amazon ElastiCache v)Amazon CloudFront or Discuss the storage services provided by AWS ?

(10M, July -Aug 2021,)(12M)

The core service is represented by Amazon Simple Storage Service (S3). This is a distributed object store that allows users to store information in different formats. The core components of S3 are two: buckets and objects. Buckets represent virtual containers in which to store

objects; objects represent the content that is actually stored. Objects can also be enriched with metadata that can be used to tag the stored content with additional information.

1 S3 key concepts

2.Amazon elastic block store 3 Amazon ElastiCache

4 Structured storage solutions 5 Amazon CloudFront

1 S3 key concepts

S3 has been designed to provide a simple storage service that's accessible through a Representational State Transfer (REST) interface.

1.stored objects cannot be manipulated like standard files..

S3 has been designed to essentially provide storage for objects that will not change over time. Therefore, it does not allow renaming, modifying, or relocating an object. Once an object has been added to a bucket, its content and position is immutable, and the only way to change it is to remove the object from the store and add it again.

2Requests will occasionally fail.

Due to the large distributed infrastructure being managed, requests for object may occasionally fail. Under certain conditions, S3 can decide to drop a request by returning an internal server error. Therefore, it is expected to have a small failure rate during day-to-day operations, which is generally not identified as a persistent failure.

3.The storage is organized in a two-level hierarchy.

S3 organizes its storage space into buckets that cannot be further partitioned. This means that it is not possible to create directories or other kinds of physical groupings for objects stored in a bucket. Despite this fact, there are few limitations in naming objects, and this allows users to simulate directories and create logical groupings.

4.Content is not immediately available to users.

The main design goal of S3 is to provide an eventually consistent data store. As a result, because it is a large distributed storage facility, changes are not immediately reflected. For instance, S3 uses replication to provide redundancy and efficiently serve objects across the globe; this practice introduces latencies when adding objects to the store—especially large ones—which are not available instantly across the entire globe

Access to S3 is provided with RESTful Web services. These express all the operations that can be performed on the storage in the form of HTTP requests (GET, PUT, DELETE, HEAD, and POST).

Resource naming

Buckets, objects, and attached metadata are made accessible through a REST interface. Therefore, they are represented by uniform resource identifiers (URIs) under the s3.amazonaws.com domain.

Amazon offers three different ways of addressing a bucket:

Canonical form: http://s3.amazonaws.com/bukect_name/

Subdomain form: http://bucketname.s3.amazon.com/

Virtual hosting form: http://bucket-name.com/ Buckets

A bucket is a container of objects. It can be thought of as a virtual drive hosted on the S3 distributed storage, which provides users with a flat store to which they can add objects. Buckets are top-level elements of the S3 storage architecture and do not support nesting. That is, it is not possible to create "sub buckets" or other kinds of physical divisions.

Objects and metadata

Objects constitute the content elements stored in S3. Users either store files or push to the S3 text stream representing the object's content. An object is identified by a name that needs to be unique within the bucket in which the content is stored. The name cannot be longer than 1,024 bytes when encoded in UTF-8, and it allows almost any character. Buckets do not support nesting.

Access control and security

Amazon S3 allows controlling the access to buckets and objects by means of Access Control Policies (ACPs). An ACP is a set of grant permissions that are attached to a resource expressed by means of an XML configuration file.

A policy allows defining up to 100 access rules, each of them granting one of the available permissions to a grantee.

Currently, five different permissions can be used:

A. READ allows the grantee to retrieve an object and its metadata and to list the content of a bucket as well as getting its metadata.
B. WRITE allows the grantee to add an object to a bucket as well as modify and remove it.
C. READ_ACP allows the grantee to read the ACP of a resource.
D. WRITE_ACP allows the grantee to modify the ACP of a resource.
E. FULL_CONTROL grants all of the preceding permissions.

2 Amazon elastic block store

The Amazon Elastic Block Store (EBS) allows AWS users to provide EC2 instances with persistent storage in the form of volumes that can be mounted at instance startup. They accommodate up to 1 TB of space and are accessed through a block device interface, thus allowing users to format them according to the needs of the instance they are connected to.

EBS volumes normally reside within the same availability zone of the EC2 instances that will use them to maximize the I/O performance. It is also possible to connect volumes located in different availability zones. Once mounted as volumes, their content is lazily loaded in the background and according to the request made by the operating system. This reduces the number of I/O requests that go to the network.

3 Amazon ElastiCache

ElastiCache is an implementation of an elastic in-memory cache based on a cluster of EC2 instances.

It provides fast data access through a Memcached-compatible protocol so that applications can transparently migrate to ElastiCache.

ElastiCache is based on a cluster of EC2 instances running the caching software, which is made available through Web services.

An ElastiCache cluster can be dynamically resized according to the demand of the client

applications.

4 Structured storage solutions

Amazon provides applications with structured storage services in three different forms:

- Preconfigured EC2 AMIs,
- Amazon Relational Data Storage (RDS), and
- Amazon SimpleDB.

Preconfigured EC2 AMIs are predefined templates featuring an installation of a given database management system. EC2 instances created from these AMIs can be completed with an EBS volume for storage persistence. Available AMIs include installations of IBM DB2, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, Sybase, and Vertica.

RDS is a relational database service that relies on the EC2 infrastructure and is managed by Amazon. Developers do not have to worry about configuring the storage for high availability, designing failover strategies, or keeping the servers up-to-date with patches. Moreover, the service provides users with automatic backups, snapshots, point-in-time recoveries, and facilities for implementing replications.

Amazon SimpleDB is a lightweight, highly scalable, and flexible data storage solution for applications that do not require a fully relational model for their data. SimpleDB provides support for semistructured data, the model for which is based on the concept of domains, items, and attributes.

SimpleDB uses domains as top-level elements to organize a data store. These domains are roughly comparable to tables in the relational model. Unlike tables, they allow items not to have all the same column structure; each item is therefore represented as a collection of attributes expressed in the form of a key-value pair.

5 Amazon CloudFront

CloudFront is an implementation of a content delivery network on top of the Amazon distributed storage infrastructure. It leverages a collection of edge servers strategically located around the globe to better serve requests for static and streaming Web content so that the transfer time is reduced.

AWS provides users with simple Web service APIs to manage CloudFront. To make available content through CloudFront, it is necessary to create a distribution. This identifies an origin server, which contains the original version of the content being distributed, and it is referenced by a DNS domain under the Cloudfront.net domain name.

The content that can be delivered through CloudFront is static (HTTP and HTTPS) or streaming (Real Time Messaging Protocol, or RMTP).

5.Explain the Amazon Communication services (8M, Model question paper)

Amazon provides facilities to structure and facilitate the communication among existing applications and services residing within the AWS infrastructure. These facilities can be organized into two major categories:

1. Virtual networking and

2. Messaging.

1. Virtual networking

Virtual networking comprises a collection of services that allow AWS users to control the connectivity to and between compute and storage services.

Amazon Virtual Private Cloud (VPC) and Amazon Direct Connect provide connectivity solutions in terms of infrastructure.

Amazon VPC provides flexibility in creating virtual private networks within the Amazon infrastructure and beyond. The service providers prepare templates for network service for advanced configurations. Templates include public subnets, isolated networks, private networks accessing Internet through network address translation (NAT), and hybrid networks including AWS resources and private resources.

Amazon Direct Connect allows AWS users to create dedicated networks between the user private network and Amazon Direct Connect locations, called ports. This connection can be further partitioned in multiple logical connections and give access to the public resources hosted on the Amazon infrastructure. The advantage is the consistent performance of the connection between the users premises and the Direct Connect locations.

Amazon Route 53 implements dynamic DNS services that allow AWS resources to be reached through domain names different from the amazon.com domain. By leveraging the large and globally distributed network of Amazon DNS servers.

2. Messaging.

The three different types of messaging services offered are

- Amazon Simple Queue Service (SQS),
- Amazon Simple Notification Service (SNS), and
- Amazon Simple Email Service (SES).

Amazon SQS constitutes disconnected model for exchanging messages between applications by means of message queues, hosted within the AWS infrastructure. Using the AWS console or directly the underlying Web service AWS, users can create an unlimited number of message queues and configure them to control their access. Applications can send messages to any queue they have access to. These messages are securely and redundantly stored within the AWS infrastructure for a limited period of time, and they can be accessed by other (authorized) applications.

Amazon SNS provides a publish-subscribe method for connecting heterogeneous applications. Amazon SNS allows applications to be notified when new content of interest is available. This feature is accessible through a Web service whereby AWS users can create a topic, which other applications can subscribe.

Amazon SES provides AWS users with a scalable email service that leverages the AWS infrastructure. Once users are signed up for the service, they have to provide an email that SES will use to send emails on their behalf. To activate the service, SES will send an email to verify the given address and provide the users with the necessary information for the activation.

6.Write a note on S3 buckets? What rules make efficient S3 storage structure?

(6M,Dec 2020)

1.stored objects cannot be manipulated like standard files..

S3 has been designed to essentially provide storage for objects that will not change over time.

Therefore, it does not allow renaming, modifying, or relocating an object. Once an object has been added to a bucket, its content and position is immutable, and the only way to change it is to remove the object from the store and add it again.

2Requests will occasionally fail.

Due to the large distributed infrastructure being managed, requests for object may occasionally fail. Under certain conditions, S3 can decide to drop a request by returning an internal server error. Therefore, it is expected to have a small failure rate during day-to-day operations, which is generally not identified as a persistent failure.

3.The storage is organized in a two-level hierarchy.

S3 organizes its storage space into buckets that cannot be further partitioned. This means that it is not possible to create directories or other kinds of physical groupings for objects stored in a bucket. Despite this fact, there are few limitations in naming objects, and this allows users to simulate directories and create logical groupings**.**

4.Content is not immediately available to users.

The main design goal of S3 is to provide an eventually consistent data store. As a result, because it is a large distributed storage facility, changes are not immediately reflected. For instance, S3 uses replication to provide redundancy and efficiently serve objects across the globe; this practice introduces latencies when adding objects to the store—especially large ones—which are not available instantly across the entire globe

7.Explain the Google AppEngine platform architecture? or core components of google engine

(08 Marks,Dec,2021)

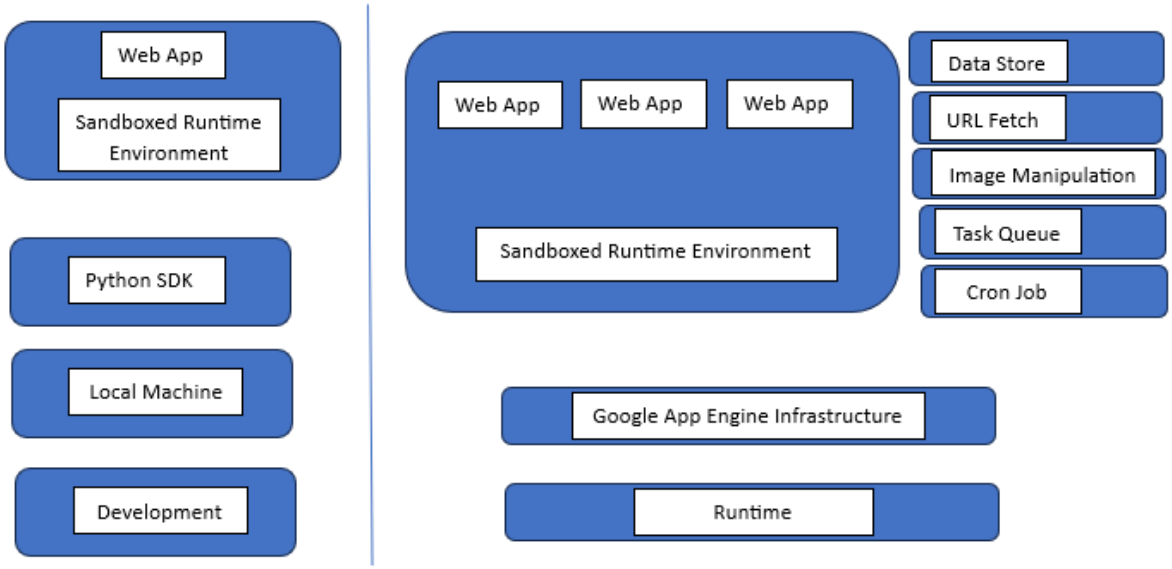
Google AppEngine is a PaaS implementation.

Distributed and scalable runtime environment that leverages Google's distributed infrastructure to scale out applications.

AppEngine is a platform for developing scalable applications accessible through the Web.

Figure 9.2.

The platform is logically divided into four major components: infrastructure, the runtime environment, the underlying storage, and the set of scalable services.



1 Infrastructure

AppEngine hosts Web applications, and its primary function is to serve users requests efficiently.

AppEngine's infrastructure takes advantage of many servers available within Google datacenters. For each HTTP request, AppEngine locates the servers hosting the application that processes the request, evaluates their load, and, if necessary, allocates additional resources or redirects the request to an existing server.

The infrastructure is also responsible for monitoring application performance and collecting statistics on which the billing is calculated.

2 Runtime environment

The runtime environment represents the execution context of applications hosted on AppEngine.

Sandboxing- One of the major responsibilities of the runtime environment is to provide the application environment with an isolated and protected context in which it can execute without causing a threat to the server and without being influenced by other applications. In other words, it provides applications with a sandbox.

If an application tries to perform any operation that is considered potentially harmful, an exception is thrown and the execution is interrupted.

Supported runtimes- Currently, it is possible to develop AppEngine applications using three different languages and related technologies: Java, Python, and Go.

AppEngine currently supports Java 6, and developers can use the common tools for Web application development in Java, such as the Java Server Pages (JSP), and the applications interact with the environment by using the Java Servlet standard.

Support for Python is provided by an optimized Python 2.5.2 interpreter. As with Java, the runtime environment supports the Python standard library.

Developers can use a specific Python Web application framework, called webapp,

simplifying the development of Web applications.

The Go runtime environment allows applications developed with the Go programming language to be hosted and executed in AppEngine. Currently the release of Go that is supported by AppEngine is r58.1. The SDK includes the compiler and the standard libraries for developing applications in Go and interfacing it with AppEngine services.

3 Storage

AppEngine provides various types of storage, which operate differently depending on the volatility of the data. **Static file servers**- Web applications are composed of dynamic and static data. Dynamic data are a result of the logic of the application and the interaction with the user. Static data often are mostly constituted of the components that define the graphical layout of the application or data files.

DataStore- DataStore is a service that allows developers to store semi structured data. The service is designed

to scale and optimized to quickly access data. DataStore can be considered as a large object database in which to store objects that can be retrieved by a specified key.

DataStore imposes less constraint on the regularity of the data but, at the same time, does not implement some of the features of the relational model.

The underlying infrastructure of DataStore is based on Bigtable, a redundant, distributed, and semistructured data store that organizes data in the form of tables.

DataStore provides high-level abstractions that simplify interaction with Bigtable. Developers define their data in terms of entity and properties, and these are persisted and maintained by the service into tables in Bigtable.

DataStore also provides facilities for creating indexes on data and to update data within the context of a transaction. Indexes are used to support and speed up queries. A query can return zero or more objects of the same kind or simply the corresponding keys.

4 Application services

Applications hosted on AppEngine take the most from the services made available through the runtime environment. These services simplify most of the common operations that are performed in Web applications **UrlFetch -** The sandbox environment does not allow applications to open arbitrary connections through sockets, but it does provide developers with the capability of retrieving a remote resource through HTTP/HTTPS by means of the UrlFetch service. Applications can make synchronous and asynchronous Web requests and integrate the resources obtained in this way into the normal request- handling cycle of the application.

UrlFetch is not only used to integrate meshes into a Web page but also to leverage remote Web services in accordance with the SOA reference model for distributed applications.

MemCache- This is a distributed in-memory cache that is optimized for fast access and provides developers with a volatile store for the objects that are frequently accessed. The caching algorithm implemented by MemCache will automatically remove the objects that are rarely accessed. The use of MemCache can significantly reduce the access time to data; developers can structure their applications so that each object is first looked up into MemCache and if there is a miss, it will be retrieved from DataStore and put into the cache for future lookups.

Mail and instant messaging- AppEngine provides developers with the ability to send and receive mails through Mail. The service allows sending email on behalf of the application to specific user accounts. It is also possible to include several types of attachments and to target multiple recipients.

AppEngine provides also another way to communicate with the external world: the Extensible Messaging and Presence Protocol (XMPP). Any chat service that supports XMPP, such as Google Talk, can send and receive chat messages to and from the Web application, which is identified by its own address.

Account management- AppEngine simplifies account management by allowing developers to leverage Google account management by means of Google Accounts.

Using Google Accounts, Web applications can conveniently store profile settings in the form of key-value pairs, attach them to a given Google account, and quickly retrieve them once the user authenticates.

5 Compute services

AppEngine offers additional services such as Task Queues and Cron Jobs that simplify the execution of computations.

Task queues- A task is defined by a Web request to a given URL, and the queue invokes the request handler by passing the payload as part of the Web request to the handler. It is the responsibility of the request handler to perform the "task execution," which is seen from the queue as a simple Web request.

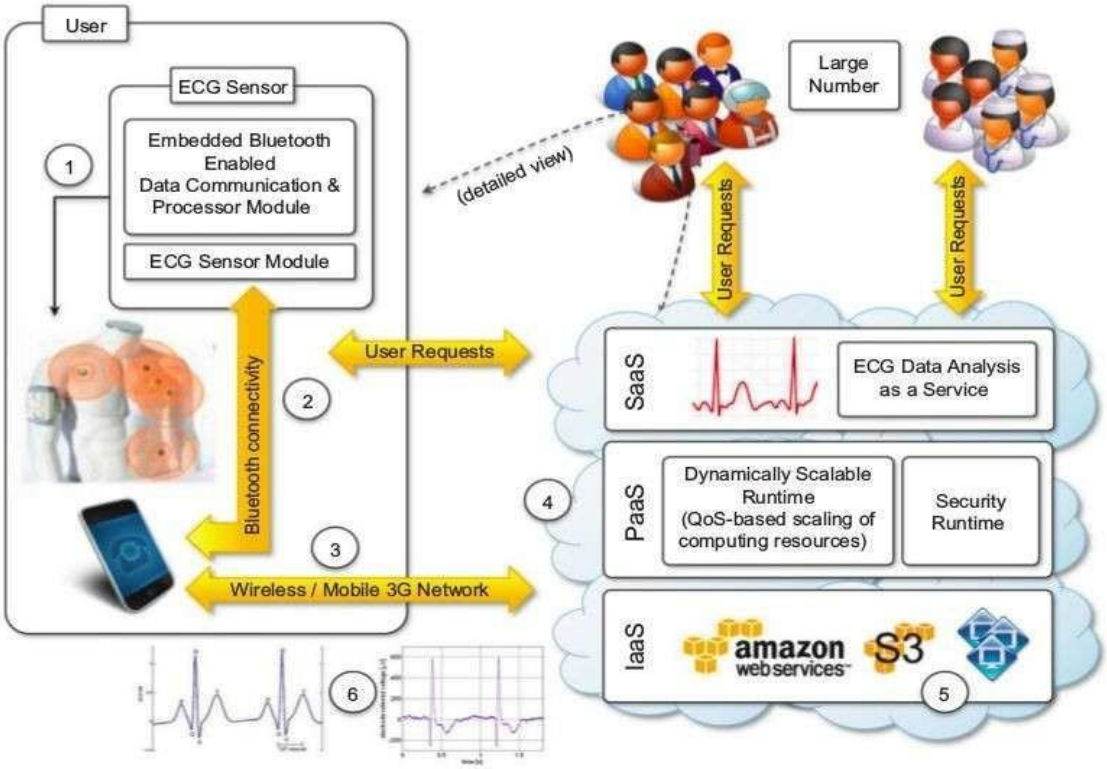
Cron jobs- the required operation needs to be performed at a specific time of the day, which does not coincide with the time of the Web request. In this case, it is possible to schedule the required operation at the desired time by using the Cron Jobs service.

8.Describe how cloud computing can be applied to remote ECG monitoring with a required diagram?

(10 Marks,Dec 2021, Aug 2022)

Healthcare is a domain in which computer technology has found several and diverse applications: from supporting the business functions to assisting scientists in developing solutions to cure diseases.

An illustration of the infrastructure and model for supporting remote ECG monitoring is shown in **Figure 10.1**. Wearable computing devices equipped with ECG sensors constantly monitor the patient's heartbeat. Such information is transmitted to the patient's mobile device, which will eventually forward it to the cloud-hosted Web service for analysis.



The Web service forms the front-end of a platform that is hosted in cloud and leverages three layers of cloud computing stack: SaaS, PaaS, and IaaS.

The Web service constitute SaaS application that will store ECG data in the Amazon S3 service and issue a processing request to the scalable cloud platform.

The runtime platform is composed of a dynamically sizable number of instances running the workflow engine and Aneka.

The number of workflow engine instances is controlled according to the number of requests in the queue of each instance, while Aneka controls the number of EC2 instances used to execute the single tasks defined by the workflow engine for a single ECG processing job.

Advantages

1. The elasticity of cloud infrastructure that can grow and shrink according to the requests served. As a result, doctors and hospitals do not have to invest in large computing infrastructures designed after capacity planning, thus making more effective use of budgets.

2. Ubiquity. Cloud computing technologies are easily accessible and promise to deliver systems with minimum or no downtime. Computing systems hosted in cloud are accessible from any Internet device through simple interfaces (such as SOAP and REST-based Web services). This makes systems easily integrated with other systems maintained on hospital's premises.

3. Cost savings. Cloud services are priced on a pay-per-use basis and with volume prices for large numbers of service requests.

- Wearable computing devices equipped with ECG sensors constantly monitor the patient's heartbeat.
- This information is transmitted to the patient's mobile device, which will eventually forward it to the cloud-hosted Web service for analysis.

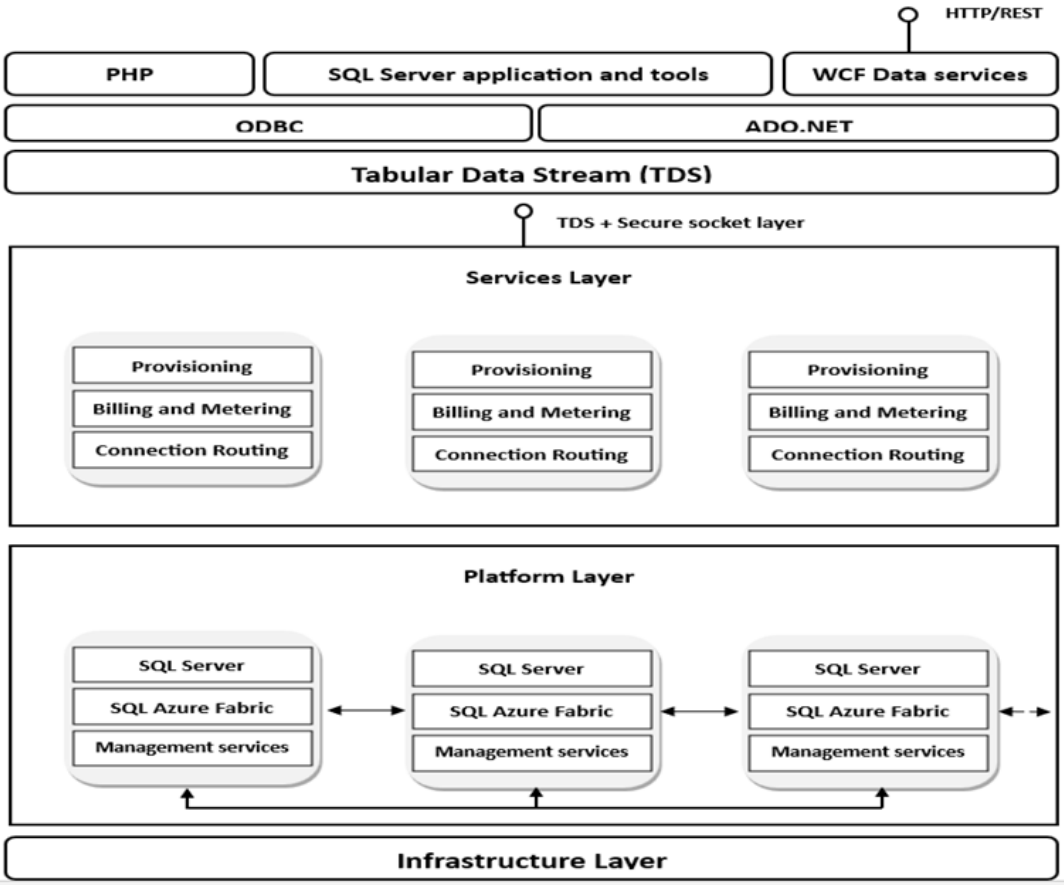
- The Web service forms the front-end of a platform that is entirely hosted in the cloud and that leverages the three layers of the cloud computing stack: SaaS, PaaS, and IaaS.
- The Web service constitute the SaaS application that will store ECG data in the Amazon S3 service ECG data in the Amazon S3 service and issue a processing request to the scalable cloud platform.
- Each of these jobs consists of a set of operations involving the extraction of the waveform from the heartbeat data and the comparison of the waveform with a reference waveform to detect anomalies.
- If anomalies are found, doctors and first-aid personnel can be notified to act on a f anomalies are found, doctors and first-aid personnel can be notified to act on a specific patient.

9. Write a note on Amazon Cloud front?

(4M,Model Questions)

- CloudFront is an implementation of a content delivery network on top of the Amazon distributed storage infrastructure
- CloudFront, it is necessary to create a distribution. This identifies an origin server, which contains the original version of the content being distributed, and it is referenced by a DNS domain under the **Cloudfront.net domain**
- **Name**
- leverages a collection of edge servers strategically located around the globe to better serve requests for static and streaming Web content so that the transfer time is reduced.
- AWS provides users with simple Web service APIs to manage CloudFront.

10..Explain SQL Azure architecture with a neat diagram (8M,Aug 2020)



SQL Azure is a relational database service hosted on Windows Azure and built on the SQL Server technologies. The service extends the capabilities of SQL Server to the cloud and provides developers with a scalable, highly available, and fault-tolerant relational database. SQL Azure is accessible from either the Windows Azure Cloud or any other location that has access to the Azure Cloud. It is fully compatible with the interface exposed by SQL Server, so applications built for SQL Server can transparently migrate to SQL Azure. **Figure 9.4** shows the architecture of SQL Azure. Access to SQL Azure is based on the Tabular Data Stream (TDS) protocol, which is the communication protocol underlying all the different interfaces used by applications to connect to a SQL Server-based installation such as ODBC and ADO.NET.

Developers have to sign up for a Windows Azure account in order to use SQL Azure. Once the account is activated, they can either use the Windows Azure Management Portal or the REST APIs to create servers and logins and to configure access to servers.

SQL Azure servers are abstractions that closely resemble physical SQL Servers: They have a fully qualified domain name under the database.windows.net (i.e., server-name.database.windows.net) domain name. This simplifies the management tasks and the interaction with SQL Azure from client applications.

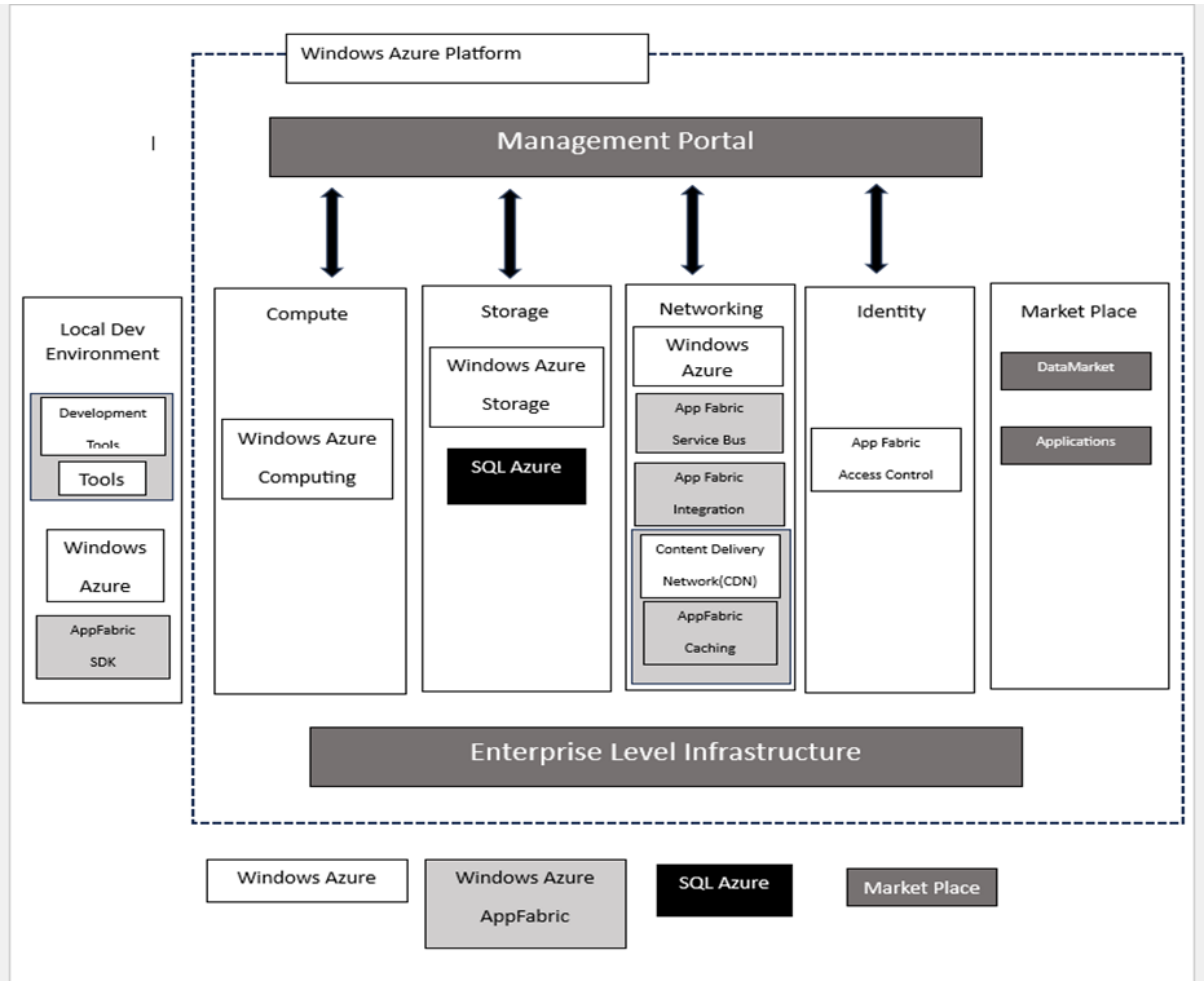
Currently, two different editions are available: Web Edition and Business Edition. The former is suited for small Web applications and supports databases with a maximum size of 1 GB or 5 GB. The latter is suited for

independent software vendors, line-of-business applications, and enterprise applications and

supports databases with a maximum size from 10 GB to 50 GB, in increments of 10 GB.

11 .Explain the Microsoft Windows Azure and services with a net diagram?

(10M,Dec 2019)



Microsoft Windows Azure is a cloud operating system built on top of Microsoft data centers' infrastructure and provides developers with a collection of services for building applications with cloud technology.

Services range from compute, storage, and networking to application connectivity, access control, and business intelligence.

Figure 9.3 provides an overview of services provided by Azure. These services can be managed and controlled through the Windows Azure Management Portal, which acts as an administrative console for all the services offered by azure platform

offered by the Azure platform.

The Windows Azure platform is made up of a foundation layer and a set of developer services that can be used to build scalable applications. These services cover compute, storage, networking, and identity management, which are tied together by middleware called AppFabric.

1 Compute services

Compute services are the core components of Microsoft Windows Azure, and they are delivered by means of the abstraction of roles.

Currently, there are three different roles: Web role, Worker role, and Virtual Machine (VM) role.

Web role- The Web role is designed to implement scalable Web applications. Web roles represent the units of deployment of Web applications within the Azure infrastructure. They are hosted on the IIS 7 Web Server.

Since version 3.5, the .NET technology natively supports Web roles; developers can directly develop their applications in Visual Studio, test them locally, and upload to Azure.

Web roles can be used to run and scale PHP Web applications on Azure (CGI Web Role).

Worker role - Worker roles are designed to host general compute services on Azure. They can be used to quickly provide compute power or to host services that do not communicate with the external world through HTTP. A common practice for Worker roles is to use them to provide background processing for Web applications developed with Web roles.

A Worker role runs continuously from the creation of its instance until it is shut down. The Azure SDK provides developers with convenient APIs and libraries that allow connecting the role with the service provided by the runtime and easily controlling its startup as well as being notified of changes in the hosting environment. **Virtual machine role-** The Virtual Machine role allows developers to control computing stack of their compute service by defining a custom image of the Windows Server 2008 R2 operating system and all the service stack required by their applications. The Virtual Machine role is based on the Windows Hyper-V virtualization technology.

Developers can image a Windows server installation complete with all the required applications and

components, save it into a Virtual Hard Disk (VHD).

2 Storage services

Compute resources are equipped with local storage in the form of a directory on the local file system.

Windows Azure provides different types of storage solutions that complement compute services with a more durable and redundant option.

Blobs

Azure allows storing large amount of data in the form of binary large objects (BLOBs) by means of the blobs service.

Block blobs- Block blobs are composed of blocks and are optimized for sequential access; therefore they are appropriate for media streaming. Currently, blocks are of 4 MB, and a single block blob can reach 200 GB in dimension.

Page blobs- Page blobs are made of pages that are identified by offset from beginning of blob. A page blob can be split into multiple pages or constituted of single page. This type of blob is optimized for random access and can be used to host data different from streaming. Maximum dimension of page blob can be 1TB.

Azure drive

Page blobs can be used to store an entire file system in the form of a single Virtual Hard Drive (VHD) file. This can then be mounted as a part of the NTFS file system by Azure

compute resources, thus providing persistent and durable storage.

Tables

Tables constitute a semi structured storage solution, allowing users to store information in the form of entities with a collection of properties. Entities are stored as rows in the table and are identified by a key, which also constitutes the unique index built for the table. Users can insert, update, delete, and select a subset of the rows stored in the table.

Currently, a table can contain up to 100 TB of data, and rows can have up to 255 properties, with a maximum of 1 MB for each row. The maximum dimension of a row key and partition keys is 1 KB.

Queues

Queue storage allows applications to communicate by exchanging messages through durable queues, thus avoiding lost or unprocessed messages. Applications enter messages into a queue, and other applications can read them in a first-in, first-out (FIFO) style.

3 Core infrastructure: AppFabric

AppFabric is a comprehensive middleware for developing, deploying, and managing applications on the cloud or for integrating existing applications with cloud services.

AppFabric implements an optimized infrastructure supporting scaling out and high availability; sandboxing and multi tenancy; state management; and dynamic address resolution and routing.

Access control- AppFabric provides the capability of encoding access control to resources in Web applications and services into a set of rules that are expressed outside the application code base. These rules give a great degree of flexibility in terms of the ability to secure components of the application and define access control policies for users and groups.

Service bus - Service Bus constitutes the messaging and connectivity infrastructure provided with AppFabric for building distributed and disconnected applications The service is designed to allow transparent network traversal and to simplify the development of loosely coupled applications, without renouncing security and reliability and letting developers focus on the logic of the interaction rather than the details of its implementation. Service Bus allows services to be available by simple URLs, which are untied from their deployment location.

Azure cache- Windows Azure provides a set of durable storage solutions that allow applications to persist their data. Azure Cache is a service that allows developers to quickly access data persisted on Windows Azure storage or in SQL Azure. The service implements a distributed in-memory cache of which the size can be dynamically adjusted by applications according to their needs.

12..Write a short note on (10M,Aug 2021,Dec 2022,Aug2022)

1 Animoto 2.Maya rendering with Aneka 3. encoding on the cloud

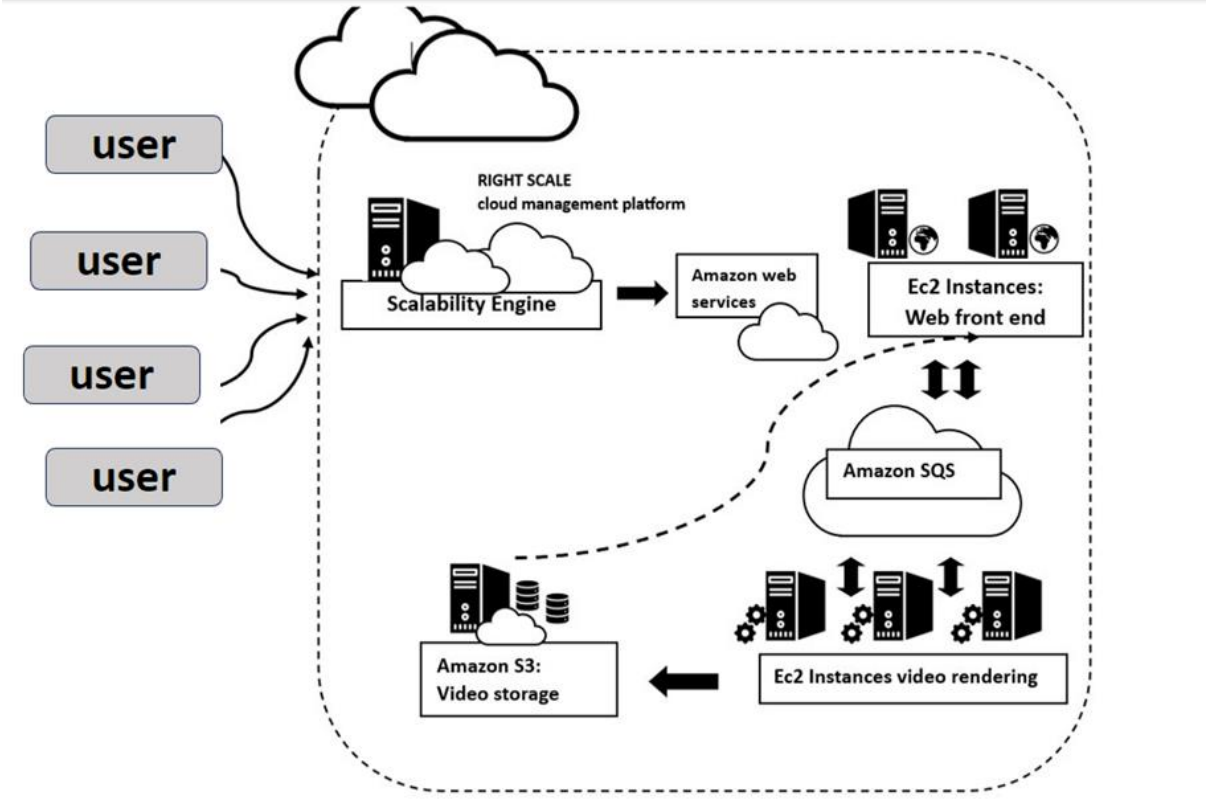
Animoto

Animoto is the most popular example of media applications on the cloud. The Website provides users with a very straightforward interface for quickly creating videos out of

images, music, and video fragments submitted by users. Users select a specific theme for a video, upload the photos and videos and order them in the sequence they want to appear, select the song for the music, and render the video. The process is executed in the background and the user is notified via email once the video is rendered.

A proprietary artificial intelligence (AI) engine, which selects the animation and transition effects according to pictures and music, drives the rendering operation. Users only have to define the storyboard by organizing pictures and videos into the desired sequence.

The infrastructure of Animoto is complex and is composed of different systems that all need to scale (**Figure 10.8**). The core function is implemented on top of the Amazon Web Services infrastructure. It uses Amazon EC2 for the Web front-end and worker nodes; Amazon S3 for the storage of pictures, music, and videos; and Amazon SQS for connecting all the components.The system's auto-scaling capabilities are managed by Rightscale, which monitors the load and controls the creation of new worker instances.



2 Maya rendering with Aneka

A private cloud solution for rendering train designs has been implemented by the engineering department of GoFront group, a division of China Southern Railway (**Figure 10.9**). The department is responsible for designing models of high-speed electric locomotives, metro cars, urban transportation vehicles, and motor trains. The design process for prototypes requires high-quality, three-dimensional (3D) images. The analysis of these images can help engineers identify problems and correct their design.

Three-dimensional rendering tasks take considerable amounts of time, especially in the case of huge numbers of frames, but it is critical for the department to reduce the time spent in these iterations. This goal has been achieved by leveraging cloud computing technologies,

which turned the network of desktops in the department into a desktop cloud managed by Aneka.

3 Video encoding on the cloud: Encoding.com

Video encoding and transcoding are operations that can greatly benefit from using cloud technologies: They are computationally intensive and potentially require considerable amounts of storage.

Encoding.com is a software solution that offers video-transcoding services on demand and leverages cloud technology to provide both the horsepower required for video conversion and the storage for staging videos. The service integrates with both Amazon Web Services technologies (EC2, S3, and CloudFront) and Rackspace (Cloud Servers, Cloud Files, and Limelight CDN access).

To use the service, users have to specify the location of the video to transcode, the destination format, and the target location of the video. Encoding.com also offers other video-editing operations such as the insertion of thumbnails, watermarks, or logos. Moreover, it extends its capabilities to audio and image conversion.

13.Briefly explain how protein structure prediction is done in cloud computing Architecture? (10 Marks, , Dec 2020)

Applications in biology require high computing capabilities and operate on large data-sets that cause extensive I/O operations.

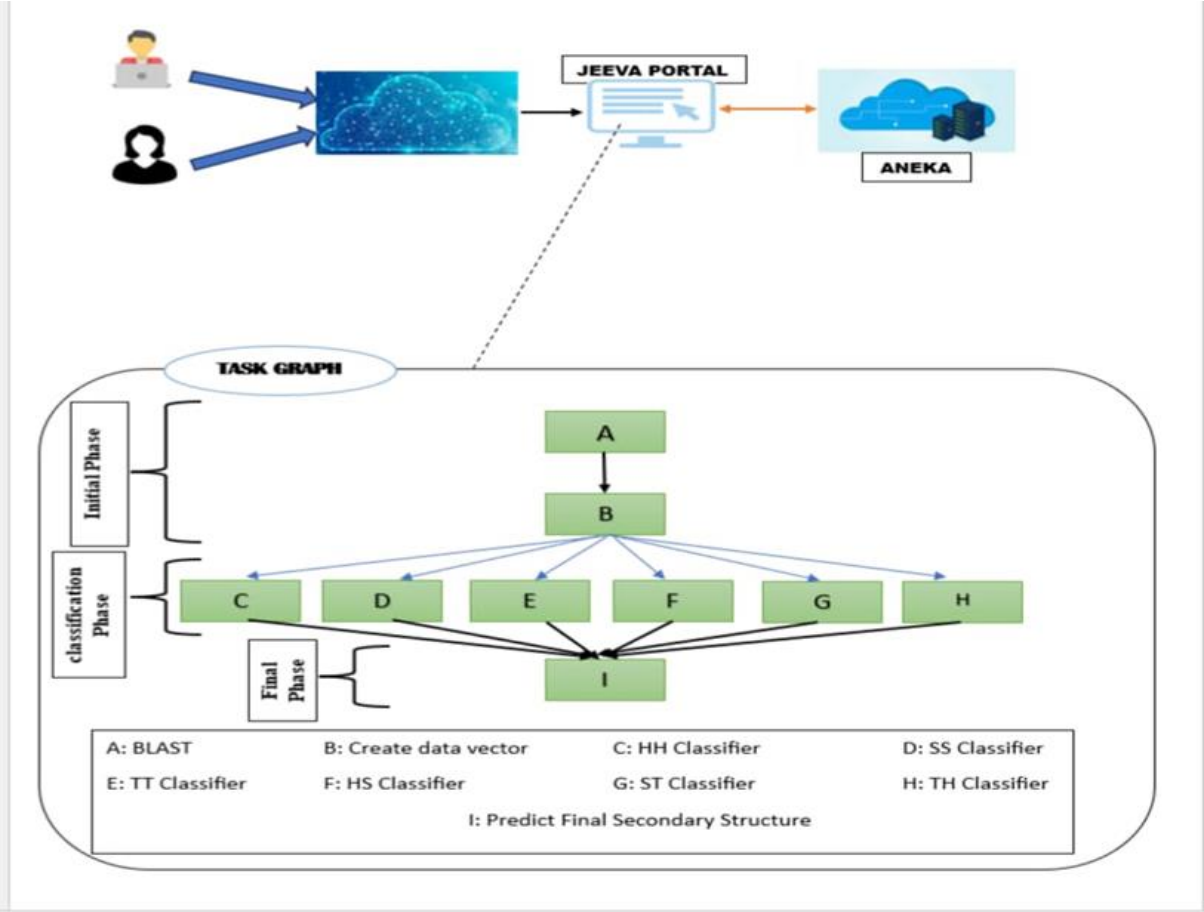
Therefore biology applications have made use of supercomputing and cluster computing infrastructures. Similar capabilities can be leveraged using cloud computing technologies in a more dynamic fashion, thus opening new opportunities for bioinformatics applications.

Protein structure prediction is a computationally intensive task that is fundamental to different types of research in the life sciences.

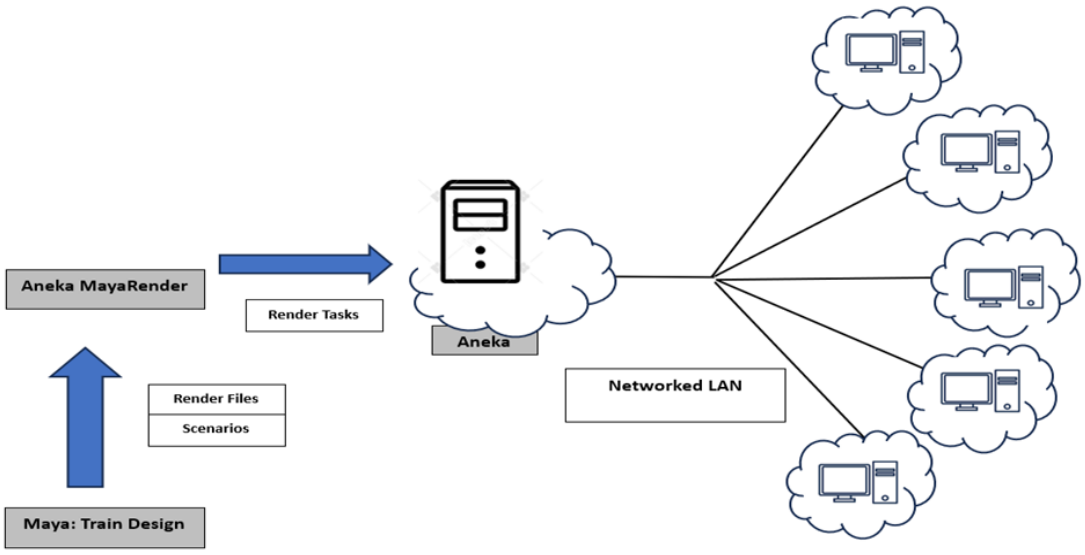
The geometric structure of a protein cannot be directly inferred from the sequence of genes that compose its structure, but it is the result of complex computations aimed at identifying the structure that minimizes the required energy.

This task requires the investigation of a space with a massive number of states, consequently creating a large number of computations for each of these states.

One project that investigates the use of cloud technologies for protein structure prediction is **Jeeva** - an integrated Web portal that enables scientists to offload the prediction task to a computing cloud based on Aneka (**Figure 10.2**).



The prediction task uses machine learning techniques (support vector machines) for



determining the secondary structure of proteins.

These techniques translate problem into one of pattern recognition, where a sequence has to

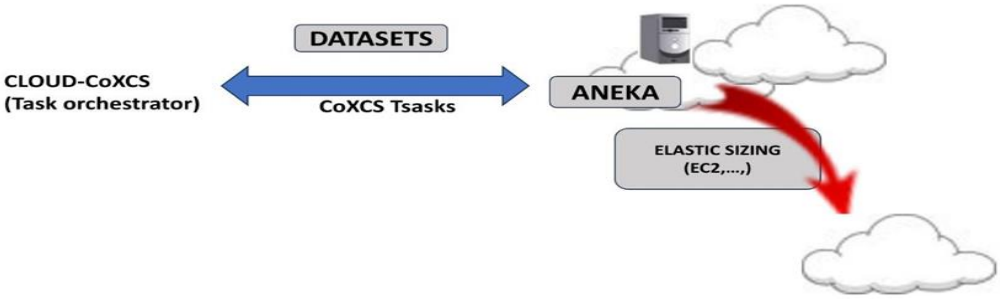
be classified into one of three possible classes (E, H, and C).
A popular implementation based on support vector machines divides the pattern recognition problem into three phases: initialization, classification, and a final phase.
These three phases have to be executed in sequence, we can perform parallel execution in the classification phase, where multiple classifiers are executed concurrently.
This reduces computational time of the prediction.
The prediction algorithm is then translated into a task graph that is submitted to Aneka.
Once the task is completed, the middleware makes the results available for visualization through the portal. **The advantage** of using cloud technologies is the capability to leverage

a scalable computing infrastructure that can be grown and shrunk on demand.

14.How gene expression data analysis for cancer diagnosis is performed on cloud ? Explain with a neat diagram (10M,Model Questions)

Gene expression profiling is the measurement of the expression levels of thousands of genes at once. It is used to understand the biological processes that are triggered by medical treatment at a cellular level.
an important application of gene expression profiling is cancer diagnosis and treatment.
Cancer is a disease characterized by uncontrolled cell growth and proliferation. This behavior occurs because genes regulating the cell growth mutate.
Gene expression profiling is utilized to provide a more accurate classification of tumors.

The dimensionality of typical gene expression datasets ranges from several thousands to over tens of thousands of genes.
This problem is approached with learning classifiers, which generate a population of condition-action rules that guide the classification process. The eXtended Classifier System (XCS) has been utilized for classifying large datasets in bioinformatics and computer science domains.
A variation of this algorithm, CoXCS [162], has proven to be effective in these conditions. CoXCS divides the entire search space into subdomains and employs the standard XCS algorithm in each of these subdomains. Such a process is computationally intensive but can be easily parallelized because the classifications problems on the subdomains can be solved concurrently.
Cloud-CoXCS (**Figure 10.3**) is a cloud-based implementation of CoXCS that leverages Aneka to solve the classification problems in parallel and compose their outcomes. The algorithm is controlled by strategies, which define the way the outcomes are composed together and whether the process needs to be iterated.



15. Explain how cloud computing is used in geo science /satellite image processing applications (10M,Aug 2022)

Geoscience applications collect, produce, and analyze massive amounts of geospatial and nonspatial data. As the technology progresses and our planet becomes more instrumented, volume of data that needs to be processed increases significantly.

Geographic information system (GIS) applications capture, store, manipulate, analyze, manage, and present all types of geographically referenced data.

Cloud computing is an attractive option for executing these demanding tasks and extracting meaningful information to support decision makers.

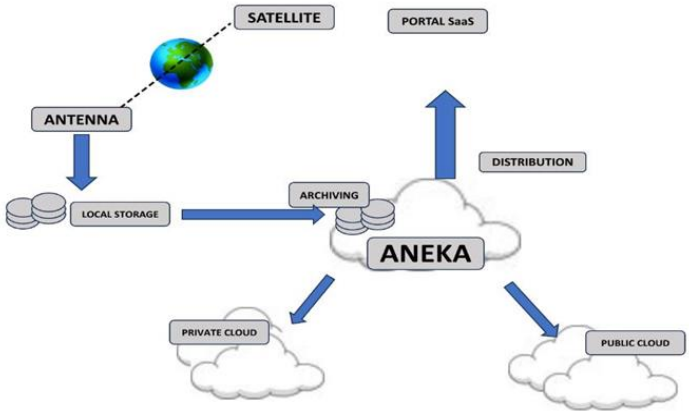
Satellite remote sensing generates hundreds of gigabytes of raw images that need to be further processed to become the basis of several different GIS products. This process requires both I/O and compute-intensive tasks. Large images need to be moved from a ground station's local storage to compute facilities, where several transformations and corrections are applied.

The system shown in **Figure 10.4** integrates several technologies across the entire computing stack.

A SaaS application provides a collection of services for such tasks as geocode generation and data visualization.

At the PaaS level, Aneka controls the importing of data into the virtualized infrastructure and the execution of image-processing tasks that produce the desired outcome from raw satellite images.

The platform leverages a Xen private cloud and the Aneka technology to dynamically provision the required resources.



16. Write a note on following

1.Sales force 2.Microsoft dynamics CRM 3. NetSuite

(10M,Aug 2021)

Salesforce.com is most popular and developed CRM solution available today.

As of today more than 100,000 customers have chosen Safesforce.com to implement their CRM solutions.

The application provides customizable CRM solutions that can be integrated with additional features developed by third parties.

Salesforce.com is based on the Force.com cloud development platform.

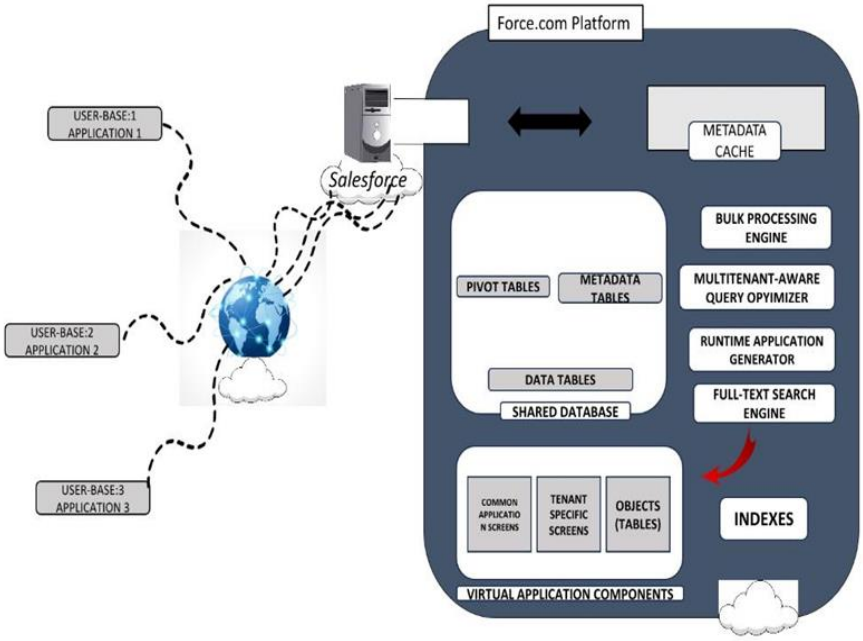
This represents scalable and high-performance middleware executing all operations of all Salesforce.com applications.

The architecture of the Force.com platform is shown in Figure 10.5. At the core of the platform resides its metadata architecture, which provides the system with flexibility and scalability.

Application core logic and business rules are saved as metadata into the Force.com store.

Both application structure and application data are stored in the store. A runtime engine executes application logic by retrieving its metadata and then performing the operations on the data.

A full-text search engine supports the runtime engine. This allows application users to have an effective user experience The search engine maintains its indexing data in a separate store.



1 Microsoft dynamics CRM

The system is completely hosted in Microsoft's data centers across the world and offers to customers a 99.9% SLA.

Each CRM instance is deployed on a separate database, and the application provides users with facilities for marketing, sales, and advanced customer relationship management.

Dynamics CRM Online features can be accessed either through a Web browser interface or

by means of SOAP and RESTful Web services.

This allows Dynamics CRM to be easily integrated with both other Microsoft products and line-of-business applications.

Dynamics CRM can be extended by developing plug-ins that allow implementing specific behaviors triggered on the occurrence of given events.

3.NetSuite

NetSuite provides a collection of applications that help customers manage every aspect of the business enterprise.

Its offering is divided into three major products: NetSuite Global ERP, NetSuite Global CRM1 , and NetSuite Global Ecommerce.

Moreover, an all-in-one solution: NetSuite One World, integrates all three products together.

The services NetSuite delivers are powered by two large datacenters on the East and West coasts of the United States, connected by redundant links.

This allows NetSuite to guarantee 99.5% uptime to its customers.

The NetSuite Business Operating System (NS-BOS) is a complete stack of technologies for building SaaS business applications that leverage the capabilities of NetSuite products.

On top of the SaaS infrastructure, the NetSuite Business Suite components offer accounting,

ERP, CRM, and ecommerce capabilities.

17.Write a note on i)Dropbox and iCloud ii)Google docs (10M,Dec 2021)

1.Dropbox and iCloud

Online storage solutions have turned into SaaS applications and become more usable as well as more advanced and accessible.

Dropbox provides users with a free storage that is accessible through the abstraction of a folder. Users can either access their Dropbox folder through a browser or by downloading and installing a Dropbox client, which provides access to the online storage by means of a special folder. Another interesting application in this area is **iCloud**, a cloud-based document-sharing application provided by Apple to synchronize iOS-based devices in a completely transparent manner.

Documents, photos, and videos are automatically sync as changes are made, without any explicit operation. This allows the system to efficiently automate common operations without any human intervention.

2.Google docs

Google Docs is a SaaS application that delivers the basic office automation capabilities with support for collaborative editing over the Web.

Google Docs allows users to create and edit text documents, spreadsheets, presentations, forms, and drawings. It aims to replace desktop products such as Microsoft Office and OpenOffice and provide similar interface and functionality as a cloud service.

By being stored in the Google infrastructure, these documents are always available from anywhere and from any device that is connected to the Internet.

ANNEXURE 2.2.1