
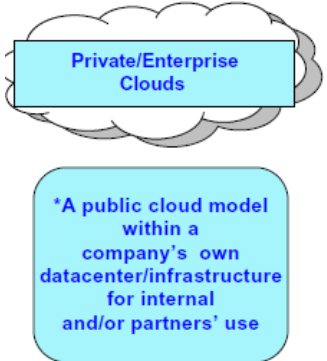
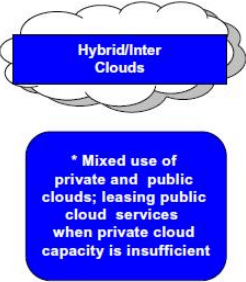
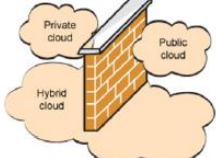
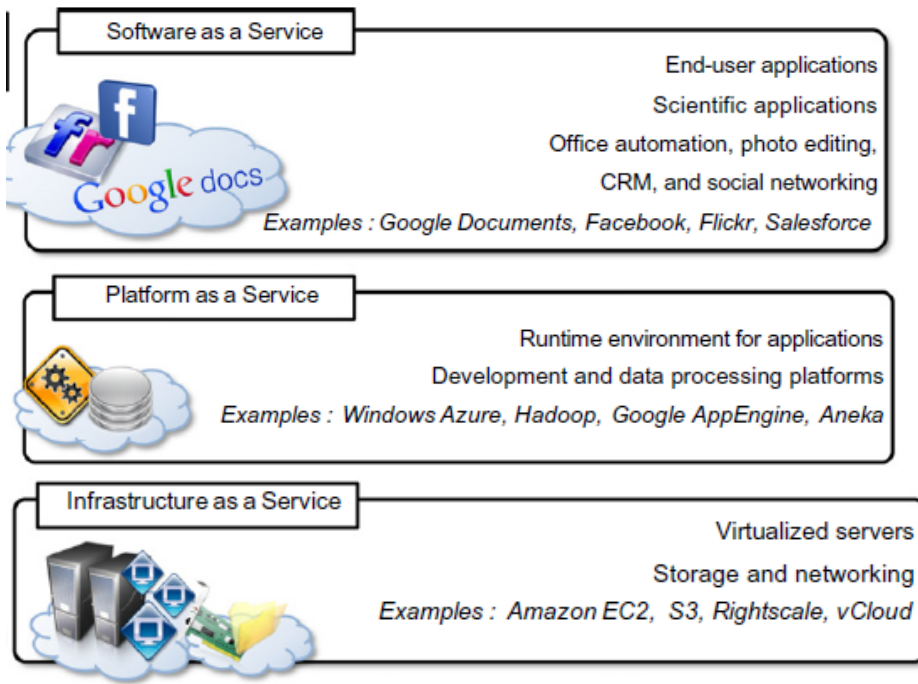


CMR Institute of Technology, Bangalore			
Department(s): Computer Science & Engineering			
Semester: 07	Section(s):	Lectures/week: 04	
Subject: Cloud Computing And Its applications		Code: 17SCS742	

VTU Question Paper Jan 2021 - Solutions

MODULE 1		
1 a.	Define Cloud Computing. With a neat diagram, explain major deployment models for cloud computing	8 Mar ks
Ans.	<p>According to NIST, 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.</p> <p>3 Major Deployment models :</p> <div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Public clouds :</p> <ul style="list-style-type: none"> established by a third-party service provider Deployed using virtualized data centers. available to any consumer on a subscription basis. users' data and applications are deployed on cloud datacenters on the vendor's premises </div> <div style="width: 45%; text-align: right;">  </div> </div> <p>Private Cloud: Large organizations that own massive computing infrastructures.</p> <ul style="list-style-type: none"> Provides more efficient use of the computing facilities. Keeps confidential information within an organization's premises. governments and banks that have high security, privacy, and regulatory concerns prefer enterprise/private clouds. <div style="display: flex; justify-content: space-between; margin-top: 20px;"> <div style="width: 45%;">  </div> <div style="width: 45%;"> <p>Hybrid Cloud</p> <ul style="list-style-type: none"> If private cloud resources are unable to meet users' quality-of-service requirements, hybrid computing systems : public cloud resources and privately owned infrastructures. </div> </div> 	
b.	Explain Cloud computing reference model with a neat diagram	8 mar ks

Ans



IaaS

- ▶ 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 VM instances.
 - ▶ Created on user's request on provider's infrastructure.
 - ▶ Users are given tools and interfaces to configure the software stack installed in the VM.
 - ▶ Pricing Model : dollars per hour depending on the characteristics of the virtual hardware.
- ▶ **Virtual Storage Space** : raw disk space – for persistent storage.
 - ▶ object space – high level abstraction for storing entities rather than files.
- ▶ **Virtual Networking** – virtual services that manage the networking among virtual instances.

PaaS

- ▶ Deliver **scalable and elastic runtime environments** on demand and host the execution of applications.
- ▶ These services are supported by a **core middleware platform** that helps create the **abstract platform** where applications are deployed and executed.
- ▶ Service provider focus on **scalability and fault tolerance**.
- ▶ User's focus on **logic of application** by using **provider's API and libraries**.
- ▶ Increases **level of abstraction** but user is also working in a **controlled environment**.

SaaS

- ▶ Software as a service : provides **application and services** on demand.
- ▶ **Common desktop applications** : office automation, photo editing, Customer Relationship Management(CRM) are replicated in provider's infrastructure.
- ▶ Made more **scalable and accessible** through a **browser** on demand.
- ▶ Interaction is **shared** among multiple users
- ▶ **SaaS** is what sustains the load based on popularity on social networking sites.

c. Discuss major milestones which has lead to cloud computing. 4 m

Ans

- 3 Major milestones that has lead to cloud computing.
- Mainframes**
- Used for bulk data processing
 - Highly reliable and capable of tolerating failures transparently.
- Clusters**
- Low cost alternative to mainframes

- Can be connected via high bandwidth network.
- Controlled by s/w tools that manage them and make them appear as a single system.
- Standard for parallel and high-performance computing

Grids

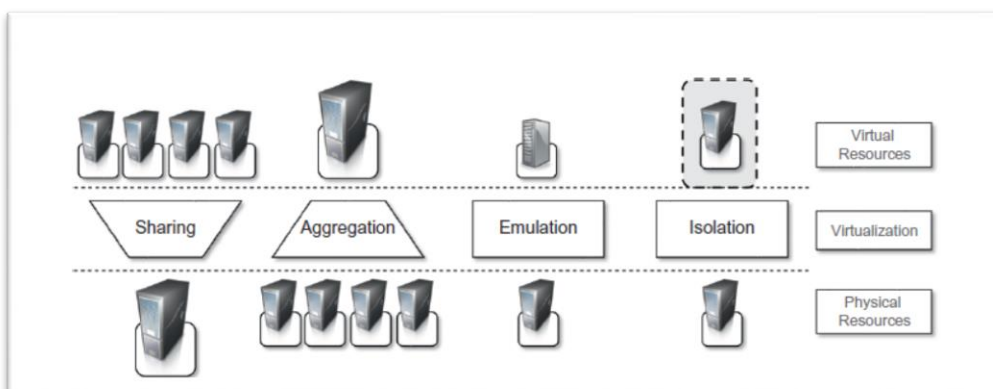
- Appeared in 1990's and is an evolution of cluster computing
- Analogy of power grid
- Access large computational power, huge storage facilities
- Started as aggregation of geographically dispersed clusters
- Grids were nation-wide or world-wide

2 a. Describe the characteristics of virtualized environments with the required diagrams

8
Mar
ks

Ans. Three major characteristics of virtualized environments are

1. Managed Execution



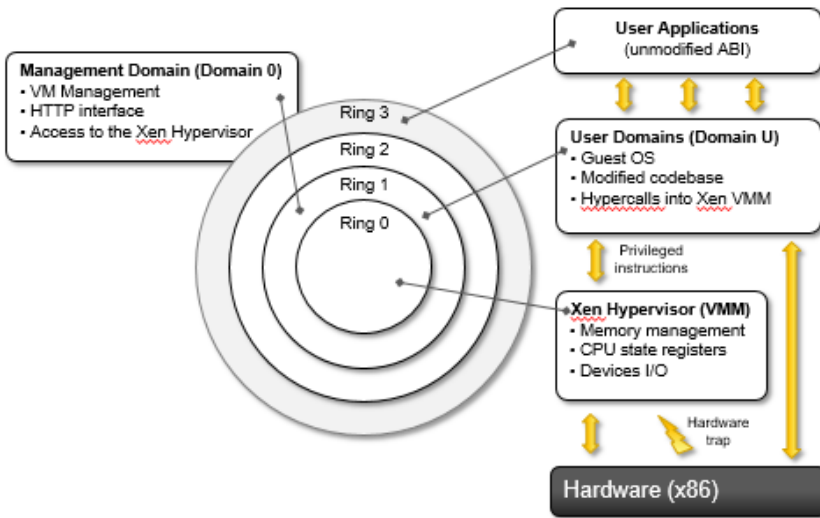
- ▶ **Sharing** – allows separate computing environments within the same host
 - ▶ **Aggregation** – when group of separate hosts are tied together to represent a single host.
 - ▶ **Emulation** - when a different environment is presented to the user.
 - ▶ Allows for controlling and tuning the environment exposed to guests
 - ▶ Useful in testing purposes where a particular platform should be validated
 - ▶ Hardware support like file I/O or SCSI can be emulated
 - ▶ Old and legacy software can be run on emulated hardware
 - ▶ Eg. Arcade game emulator on PC.
 - ▶ **Isolation** – allows guests to operate on a separate environment.
 - ▶ Guest interacts with the abstraction layer which provides access to the underlying layers.
 - ▶ It allows multiples hosts to run without interfering with one another
 - ▶ Provides a separation between a guest and a host.
 - ▶ VM can filter activity of guest.
 - ▶ Another important characteristic : **performance Tuning**
 - ▶ Migration of VM's
 - ▶ Provide QoS stated in the SLA
- 2. Increased Security**
- ▶ All operations by guest are performed by VM
 - ▶ This allows for **control and filter**
 - ▶ Resources exposed to host are hidden or protected from guest.
 - ▶ Increase security is a requirement when running untrusted code.
 - ▶ Sandboxed inside JVM
- 3. Portability**
- ▶ **Hardware virtualization** : guest is packaged as a VM image.
 - ▶ **Programming level virtualization** : JVM or .NET this represents the application components(jars or assemblies) that can run without any recompilation.

- Means having your own system and being able to run it as long as the VMM is available.

b. With neat diagram, explain Xen architecture and guest OS management.

6 m

Ans.



- Describes the architecture of Xen and its mapping onto a classic x86 privilege model.
- A Xen-based system is managed by the Xen hypervisor,
 - Runs in the highest privileged mode and controls the access of guest operating system to the underlying hardware.
- Guest OS run within domains.
- Special control software is at Domain 0

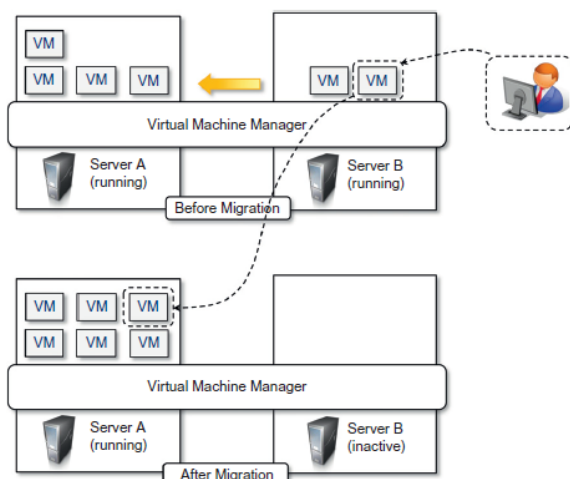
Guest OS Management

- Guest operating systems are executed within domains
- 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, configuration, and termination.
- Many of the x86 implementation 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.

c. Explain live migration and server consolidation with a neat diagram.

6m

Ans.



- **Server Consolidation:** When resources are underutilized, the number of

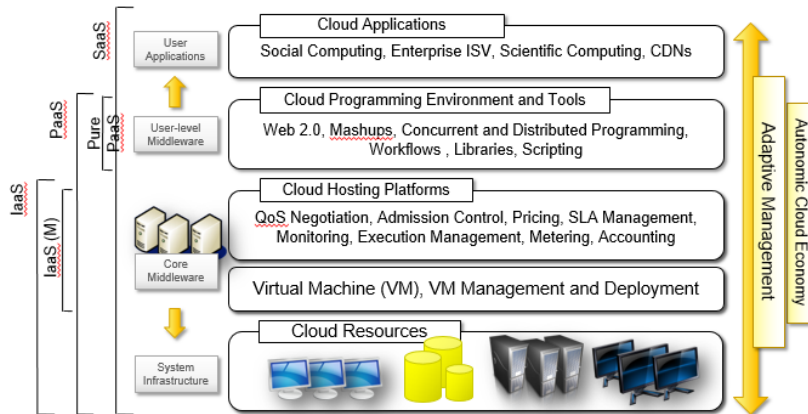
- active resources can be reduced by aggregating virtual machines over a smaller number of resources that become fully utilized.
- ▶ **Live Migration** : the movement of virtual machine instances with no disruption of activity

MODULE 2

3 a. Explain cloud computing architecture with a neat diagram

8 m

Ans.



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**.

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.

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. Hypervisors manage the pool of resources and expose the distributed infrastructure as a collection of virtual machines. By using virtual machine technology it is possible to finely partition the hardware resources such as CPU and memory and to virtualize specific devices, thus meeting the requirements of users and applications.

IaaS :

The combination of cloud hosting platforms and resources is generally classified as a Infrastructure-as-a-Service(IaaS) solution. We can organize the different examples of IaaS into two categories: Some of them provide both the management layer and the physical infrastructure; others provide only the management layer(IaaS (M)).

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

PaaS : Platform-as-a-Service(PaaS) service offered to the user is a development platform rather than an infrastructure. 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.

Category	Description	Product Type	Vendors and Products
PaaS-I	Runtime environment with Web-hosted application development platform. Rapid application prototyping.	Middleware + Infrastructure Middleware + Infrastructure	Force.com Longjump
PaaS-II	Runtime environment for scaling Web applications. The runtime could be enhanced by additional components that provide scaling capabilities.	Middleware + Infrastructure Middleware Middleware + Infrastructure Middleware + Infrastructure Middleware + Infrastructure Middleware	Google AppEngine AppScale Heroku Engine Yard Joyent Smart Platform GigaSpaces XAP
PaaS-III	Middleware and programming model for developing distributed applications in the cloud.	Middleware + Infrastructure Middleware Middleware Middleware Middleware Middleware	Microsoft Azure DataSynapse Cloud IQ Manjrasof Aneka Apprenda SaaSGrid GigaSpaces DataGrid

SaaS : Coined in 2001 by the Software Information & Industry Association(SIIA) *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.*

b. Discuss how SaaS provides access to application through the Internet as a web based service **4m**

Ans. The SaaS model is appealing for applications serving a wide range of users and that can be adapted to specific needs with little further customization. This requirement characterizes SaaS as a "one-to-many" software delivery model, whereby an application is shared across multiple users. This is the case of Customer Relationship Management(CRM) and Enterprise Resource Planning(ERP) applications that constitute common needs for almost all enterprises, from small to medium-sized and large business.

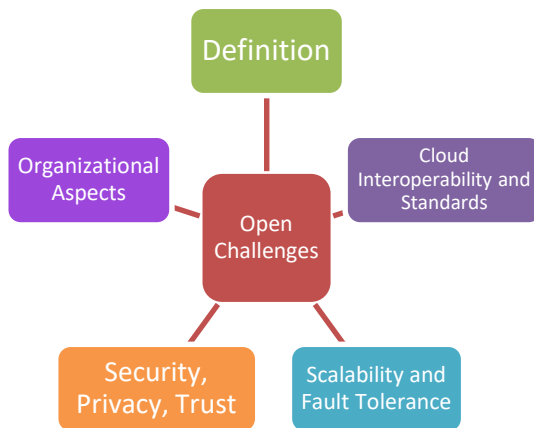
Every enterprise will have the same requirements for the basic features concerning CRM and ERP; different needs can be satisfied with further customization.

This scenario facilitates the development of software platforms that provide a general set of features and support specialization and ease of integration of new components. Moreover, it constitutes the perfect candidate for hosted solutions, since the applications delivered to the user are the same, and the applications themselves provide users with the means to shape the applications according to user needs.

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.

c. Discuss the various open challenges in cloud computing **8m**

Ans



Cloud Definition: Despite the general agreement on the NIST definition, there are alternative taxonomies for cloud services 10 different classes and better suits the vision of cloud computing within the enterprise.

Another proposition which departs from the XaaS concept and tries to define an ontology for cloud computing. In their work the concept of a cloud is dissected into five main layers: applications, software environments, software infrastructure, software kernel, and hardware.

Cloud Interoperability and Standards:

Vendor lock-in constitutes one of the major strategic barriers against the seamless 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.

The standardization efforts are mostly concerned with the lower level of the cloud computing architecture, which is the most popular and developed. In particular, in the IaaS market, the use of a proprietary virtual machine format constitutes the major reasons for the vendor lock-in, and efforts to provide virtual machine image compatibility between IaaS vendors can possibly improve the level of interoperability among them. Another direction in which standards try to move is devising a general reference architecture for cloud computing systems and providing a standard interface through which one can interact with them.

Scalability and Fault tolerance:

cloud middleware has to be designed with the principle of scalability along different dimensions in mind—for example, performance, size, and load. The challenge is designing highly scalable and fault-tolerant systems that are easy to manage and at the same time provide competitive performance.

Security, trust and privacy:

The massive use of virtualization technologies exposes the existing system to new threats, which previously were not considered applicable. On one side we need to decide whether to trust the provider itself; on the other side, specific regulations can simply prevail over the agreement the provider is willing to establish with us concerning the privacy of the information managed on our behalf. Moreover, cloud services delivered to the end user can be the result of a complex stack of services that are obtained by third parties via the primary cloud service provider.

Organizational Aspects:

- What is the new role of the IT department in an enterprise that completely or significantly relies on the cloud?
- How will the compliance department perform its activity when there is a considerable lack of control over application workflows?
- What are the implications (political, legal, etc.) for organizations that lose control over some aspects of their services?

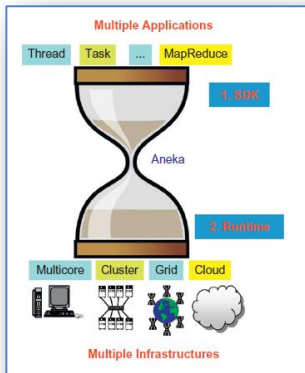
- What will be the perception of the end users of such services?

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

4.a. Discuss the anatomy of Aneka container in detail.

12 m

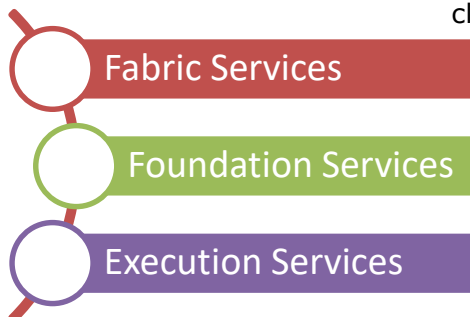
Ans



- Aneka consists of a **scalable cloud middleware** that can be deployed on top of heterogeneous computing resources
 - Extensible **API - Task, Thread, MapReduce**
 - For development and deployment of application in different type of cloud deployment models - **private, public, hybrid.**
 - Pure PaaS** solution for cloud computing
 - Physical and virtual resources representing the bare metal of the cloud are managed by the **Aneka container.**
 - The container is installed in each node
 - Collection of **interconnected containers:** Aneka Cloud

Aneka Container

- These take care of infrastructure management, supporting services for Aneka cloud, application management and execution.
 - Made available to developers and administrators by means of **application management and development layer.**
 - Includes **interfaces and APIs** for developing cloud applications and management tools and interfaces for controlling Aneka clouds.
 - Implements Service Oriented Architecture (SOA)



- is a **lightweight software layer** designed to host services and interact with the underlying operating system and hardware
- Role:** deploy services and some basic capabilities such as communication channels through which it interacts with other nodes in the Aneka Cloud.
- Almost all operations performed within Aneka are carried out by the services managed by the container.
- Services stack resides on top of **the Platform Abstraction Layer(PAL)**
- It provides a uniform view of the soft- ware and hardware environment in which the container is running

PAL :

- The PAL is responsible for **detecting the supported hosting environment** and providing the corresponding implementation to interact with it to support the activity of the container.
- The PAL is a small layer of software
 - a **detection engine**, which automatically configures the container at boot time, with the platform-specific component to access the above information
 - an implementation of the **abstraction layer** for the Windows, Linux, and Mac OS X operating systems.
- additional custom information can be retrieved by querying the **properties of the hardware**
- using **name-value pairs** that can host any kind of information about the hosting platform
- these properties can contain additional information about the processor, such

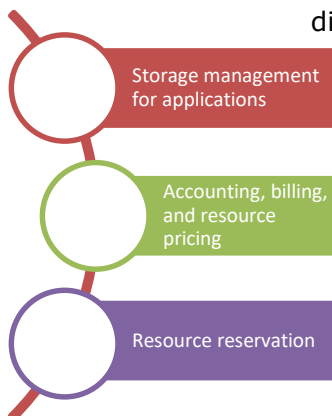
as the **model and family**, or additional data about the process running the container

Fabric Services :

- ▶ 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** :dynamically provide new nodes on demand by relying on virtualization technologies.
- ▶ **monitoring services** allow for **hardware profiling**
 - ▶ implement a basic monitoring infrastructure that can be used by **all the services installed in the container .**
- ▶ **Monitoring and Reporting Services** : implement a generic infrastructure for monitoring the activity of any service in the Aneka Cloud.
- ▶ **Heartbeat Service** : periodically collects the **dynamic performance information** about the node and publishes this information to the **membership service** in the Aneka Cloud
- ▶ **Reporting Service** manages the store for monitored data and makes them accessible to the services or external applications for analysis purposes.
- ▶ **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.
- ▶ The **Membership Catalogue** 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 makes available information about data transfer, such as upload and download times, filenames, and sizes.
- ▶ The **Resource Provisioning Service** tracks the provisioning and life time information of virtual nodes.

Foundation Services:

- ▶ 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.



- ▶ Foundation Services provide a **uniform approach to managing distributed applications**

- ▶ allow developers to concentrate only on the **logic** that distinguishes a specific programming model from the others.

- ▶ The Fabric Services and Foundation Services constitute the **core of the Aneka middleware**

Storage Management and Application : 2 facilities are provided by Aneka

- ▶ **a centralized file storage** : used for the execution of **compute-intensive applications**
 - ▶ require **powerful processors**
 - ▶ do not have **high demands in terms of storage**
- ▶ **a distributed file system** : more suitable for the execution of **data-intensive applications**
- ▶ **Accounting Services** keep track of the status of applications in the Aneka Cloud
 - ▶ provides a **detailed breakdown** of the distributed infrastructure usage
 - ▶ vital for the proper management of resources.
- ▶ **Billing Service** : provides detailed information about each user's usage of resources, with the associated costs
 - ▶ **Resource Reservation** :keeps track of all the reserved time slots in

the Aneka Cloud and provides a unified view 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.

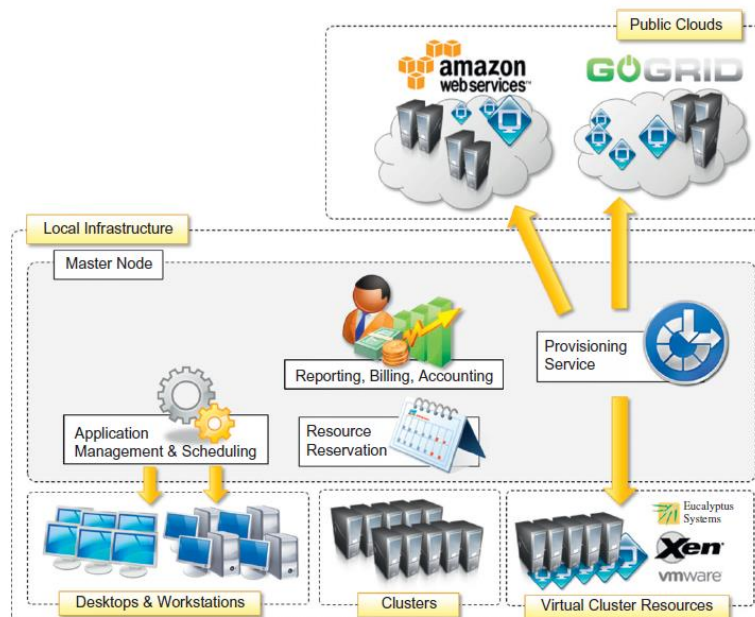
Execution Services

- ▶ manage the **execution of applications**
- ▶ A layer that differentiates according to the specific **programming model** used for developing distributed applications on top of Aneka.
- ▶ Two major types of activities that are common across all the supported models.
- ▶ Scheduling Services in charge of **planning the execution** of distributed applications on top of Aneka
- ▶ governing the **allocation of jobs** composing an application **to nodes**
- ▶ The integration point with several other Foundation and Fabric Services:
 - ▶ the Resource Provisioning Service
 - ▶ the Reservation Service
 - ▶ the Accounting Service
 - ▶ Reporting Service
- ▶ Execution Service : control **the execution of single jobs** that compose applications.
- ▶ They are in charge of **setting up the runtime environment** hosting the execution of jobs.
 - ▶ **Unpacking** the jobs received from the scheduler
 - ▶ Retrieval of **input files** required for job execution
 - ▶ **Sandboxed** execution of jobs
 - ▶ Submission of **output files** at the end of execution
 - ▶ Execution **failure management** (i.e., capturing sufficient contextual information useful to identify the nature of the failure)
 - ▶ **Performance monitoring**
 - ▶ **Packing jobs** and sending them back to the scheduler

b. Explain Aneka hybrid cloud deployment mode with a neat diagram.

8m

Ans



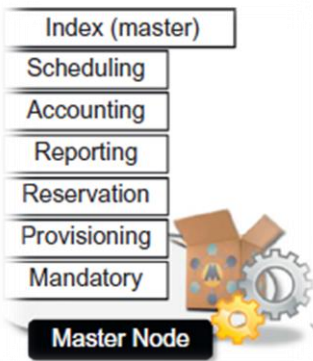
- ▶ constitutes the static deployment of Aneka that can be elastically scaled on demand when additional resources are required.

The framework consists of Dynamic Resource Provisioning • Resource Reservation • Workload Partitioning • Accounting, Monitoring, and Reporting

If the local premises offer some virtual machine management capabilities, it is

possible to provide a very efficient use of resources, thus minimizing the expenditure for application execution.

- ▶ **Administrative Console:** performs all the required *management operations*.
- ▶ **A repository:** provides storage for all the *libraries* required to lay out and install the basic Aneka platform.
 - ▶ Make up the **software image** for the *node manager and the container programs*.
 - ▶ make libraries available through a variety of communication channels, such as HTTP, FTP, common file sharing
- ▶ **The Management Console** can *manage multiple repositories* and select the one that *best suits the specific deployment*
- ▶ **Aneka daemon :** The infrastructure is deployed by taking a *collection of nodes* and installing them on the Aneka node manager.



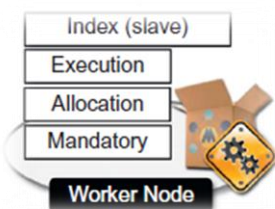
- ▶ Consists of **Remote management service :** used to *deploy and control* container instances
 - ▶ Index Service (master copy)
 - ▶ Heartbeat Service
 - ▶ Logging Service
 - ▶ Reservation Service
 - ▶ Resource Provisioning Service
 - ▶ Accounting Service
 - ▶ Reporting and Monitoring Service
 - ▶ Scheduling Services for the supported programming models

Master Node

- ▶ also provides connection to an **RDBMS facility** where **state of several services** is maintained.
- ▶ **all the scheduling services** are maintained in the **master node**.
- ▶ share the **application store** that is normally persisted on the RDBMS to provide a **fault-tolerant infrastructure**.
- ▶ master configuration is replicated in several nodes
 - ▶ To provide a highly available infrastructure based on the **failover mechanism**

Worker Nodes

- ▶ The **workforce** of the Aneka Cloud
- ▶ generally configured for the **execution of applications**.
- ▶ **mandatory services**
- ▶ **specific execution services** of each of the supported programming models in the Cloud.
- ▶ Common Configuration



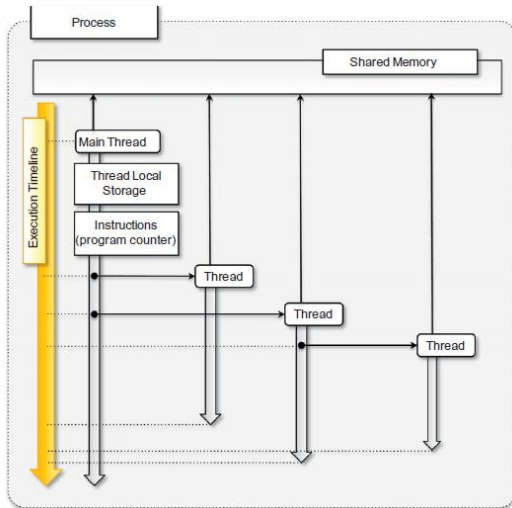
- ▶ Index Service
- ▶ Heartbeat Service
- ▶ Logging Service
- ▶ Allocation Service
- ▶ Monitoring Service
- ▶ Execution Services for the supported programming models

MODULE 3

5. a. What is a thread? Discuss different thread APIs. 6m

- Ans
- ▶ A thread identifies a **single control flow**: logical sequence of instructions within a process.
 - ▶ **What are threads to OS?**
 - ▶ Minimal building blocks for expressing running code.
 - ▶ All operations run as threads and have variable lifetimes.
 - ▶ Threads of the same process share the same execution context
 - ▶ **What concept is used in multitasking?**

- ▶ The process of execution multiple threads by stopping execution of a thread and starting execution of another thread.
- ▶ **Context switching** – save information of current thread/process in the register onto a stack and load information of thread/process to execute onto the registers.
- ▶ Running program : **main thread** which is **implicitly created** by the **compiler** or **runtime environment**.
- ▶ All threads have their own **memory space** allocated for the **entire process**.
- ▶ **Execution of process is terminated** when all the other **threads are completed**.



Thread API

- ▶ minimum set of features needed to support **multithreading**
- ▶ **POSIX Threads**
- ▶ **Portable Operating System Interface for UNIX** : set of standards related to API for portable development for UNIX OS flavors.
 - ▶ **Creation and deletion of thread**
 - ▶ **Thread synchronization : mutexes, joins, conditions**
 - ▶ Threading support in **Java and .NET**

b. With a neat diagram, compare thread life cycle in system threading and Aneka threading

8m

Ans

- ▶ Local threads belong to the **hosting process**.
- ▶ To create a thread, only necessary to provide a **pointer** to a method
- ▶ Aneka threads live in the **context** of a **distributed application**
- ▶ **multiple distributed applications** can be managed within a single process
- ▶ **thread creation** also requires the specification of the reference to the application to which the thread belongs

.Net Threading API	Aneka Threading API
<i>System.Threading</i>	<i>Aneka.Threading</i>
<i>Thread</i>	<i>AnekaThread</i>
<i>Thread.ManagedThreadId (int)</i>	<i>AnekaThread.Id (string)</i>
<i>Thread.Name</i>	<i>AnekaThread.Name</i>
<i>Thread.ThreadState (ThreadState)</i>	<i>AnekaThread.State</i>
<i>Thread.IsAlive</i>	<i>AnekaThread.IsAlive</i>
<i>Thread.IsRunning</i>	<i>AnekaThread.IsRunning</i>
<i>Thread.IsBackground</i>	<i>AnekaThread.IsBackground [false]</i>
<i>Thread.Priority</i>	<i>AnekaThread.Priority [ThreadPriority.Normal]</i>
<i>Thread.IsThreadPoolThread</i>	<i>AnekaThread.IsThreadPoolThread [false]</i>
<i>Thread.Start</i>	<i>AnekaThread.Start</i>
<i>Thread.Abort</i>	<i>AnekaThread.Abort</i>
<i>Thread.Sleep</i>	[Not provided]
<i>Thread.Interrupt</i>	[Not provided]
<i>Thread.Suspend</i>	[Not provided]
<i>Thread.Resume</i>	[Not provided]
<i>Thread.Join</i>	<i>AnekaThread.Join</i>

- ▶ Aneka threads live and execute in a **distributed environment**, so their lifecycle is different from life cycle of threads.
- ▶ Not possible to **map state value** of local threads to Aneka threads.
- ▶ **Local threads** : most of the state transitions are triggered by the developer
- ▶ In Aneka : triggered by the **Aneka Middleware**
- ▶ **Aneka threads** : has more states
- ▶ they support **file staging**, scheduled by the middleware, which can queue

- ▶ them for a period of time
- ▶ They support **reservation of nodes** : state indicating execution failure due to missing reservation credentials.

Typical thread cycle in Aneka:

- ▶ **Unstarted**
- ▶ On invoking `Start()` method moves to **Started** state
- ▶ Moves to **StagingIn** state (if there are files to upload for execution) or **Queued** state
- ▶ If there is an error, goes to **Failed** state.
- ▶ **Rejected** State is reached for invalid reservation token
- ▶ From the **Queued** state, if there is a free node, it moves to the **Running** state
- ▶ If thread generates an exception, moves to **Failed** state
- ▶ On successful completion, moves to the **Completed** state
- ▶ If output files are yet to be retrieved, it moves to **StagingOut** state
- ▶ If developer directly calls `Abort()` method, goes into the **Abort** state.

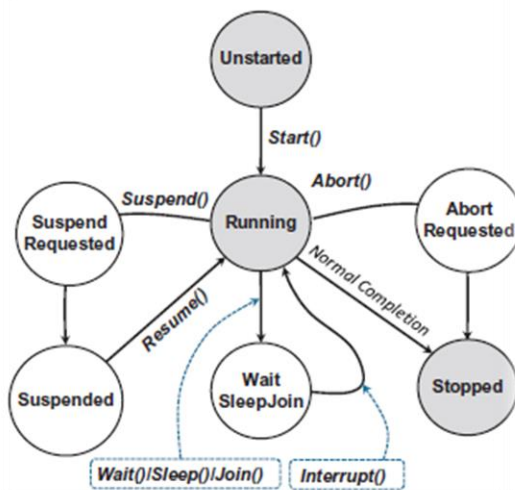


Indicate they do not have a corresponding mapping

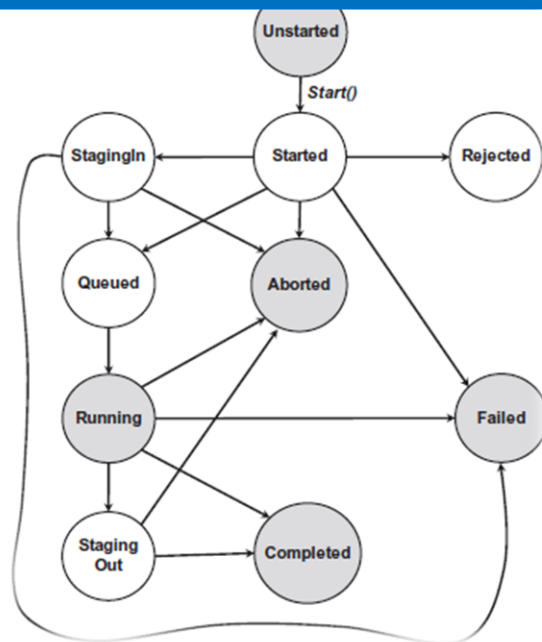


Indicate common states

System.Threading.Threadlife Life cycle



Aneka.Threading.AnekaThreadlife Life cycle



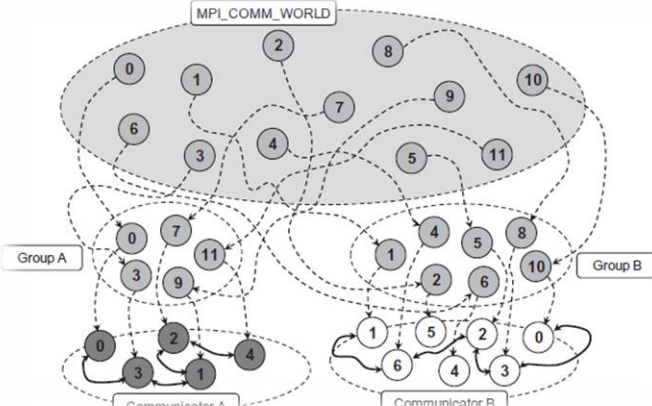
c. Explain Aneka thread application model with a listing for application creation and configuration

6m

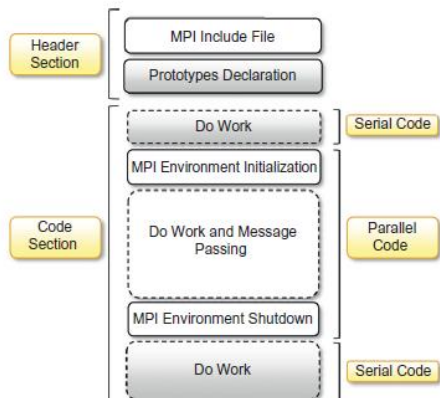
- Ans
- ▶ Thread Programming Model : programmer creates units of work as Aneka Threads.
 - ▶ **AnekaApplication<W,M>**
 - ▶ Aneka API makes strong use of **generics**
 - ▶ Categorize the support given to different programming models through template specialization.
 - ▶ To develop distributed applications :
 - ▶ **AnekaApplication<AnekaThread, ThreadManager>**
 - ▶ Class type for all distributed applications.
 - ▶ Defined in **Aneka.Threading** namespace noted in Aneka.Threading.dll library.
 - ▶ **Configuration class** : defined in **Aneka.Entity** , **Aneka.dll** library
 - ▶ Contains a set of properties that allow application class to configure middleware
 - ▶ Address of Aneka index service

	<ul style="list-style-type: none"> ▶ User credentials – required to authenticate the application with the middleware. ▶ Additional tuning parameters <pre> using System; using System.Collections.Generic; using Aneka; using Aneka.Util; using Aneka.Entity; // Aneka Thread Programming Model user classes using Aneka.Threading; private AnekaApplication <AnekaThread,ThreadManager> CreateApplication(); { Configuration conf =new Configuration(); conf.SchedulerUri = newUri("tcp://localhost:9090/Aneka"); conf.Credentials =newUserCredentials("Administrator", string.Empty); conf.UseFileTransfer = false; // we create the application instance and configure it. AnekaApplication<AnekaThread,ThreadManager> app = new AnekaApplication<AnekaThread,ThreadManager>(conf); return app; } </pre>	
--	--	--

6.a	Explain MPI reference scenario and MPI program structure with the required diagrams.	8m
------------	---	-----------

Ans	<ul style="list-style-type: none"> ▶ Message Passing Interface(MPI) is a specification for developing parallel programs that communicate by exchanging messages. ▶ Compared to earlier models, MPI introduces the constraint of communication that involves MPI tasks that need to run at the same time. ▶ MPI has originated as an attempt to create common ground from the several distributed shared memory and message-passing infrastructures available for distributed computing.  <ul style="list-style-type: none"> ▶ MPI provides developers with a set of routines that: <ul style="list-style-type: none"> ▶ 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 ▶ 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 	
-----	---	--

- ▶ It is possible to create specific groups as sub-sets of this global group—for example, for isolating all the MPI processes that belong to the same application



- ▶ Each MPI process is assigned a rank within the group to which it belongs. 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

- ▶ A common model used in MPI is the master-worker model, whereby one MPI process (usually the one with rank) 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

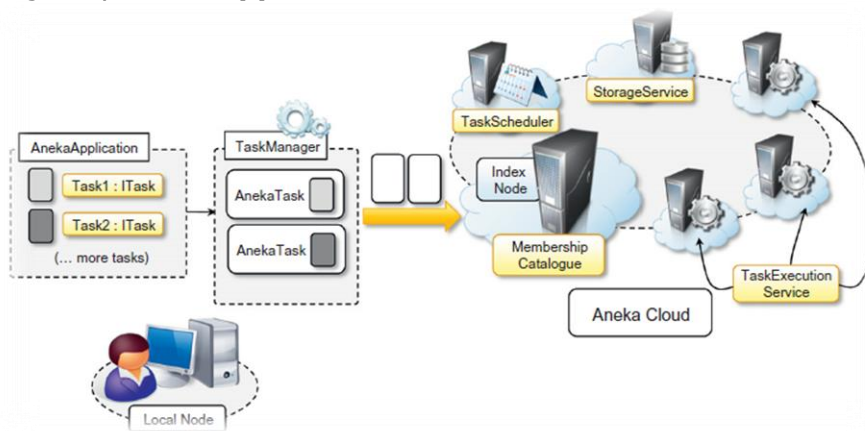
for the language. This compiler introduces additional code in order to properly manage the MPI runtime. The output of the compilation process can be run as a distributed application by using a specific tool provided with the MPI implementation

b. Explain task programming model with a neat diagram

6m

- Ans
- ▶ Task programming is realized through the abstraction of the Aneka.Tasks.ITask. AnekaApplication class

- ▶ Managed by **AnekaApplication** class



- ▶ **AnekaTask** and **TaskManager** : constitute client side view
- ▶ In middleware 4 services constitute underlying component that interacts with Aneka:
 - ▶ **Membership catalogue:** main access point and acts as a service directory to locate Task Scheduler
 - ▶ **Task Scheduler** : manages execution of task based applications
 - ▶ **ExecutionService** : for execution and monitoring task state.
 - ▶ **StorageService** : data transfer support in the form of data files, input and output files
 - ▶ **Web Service** : allows any client to submit tasks for execution

Developing applications with the task model

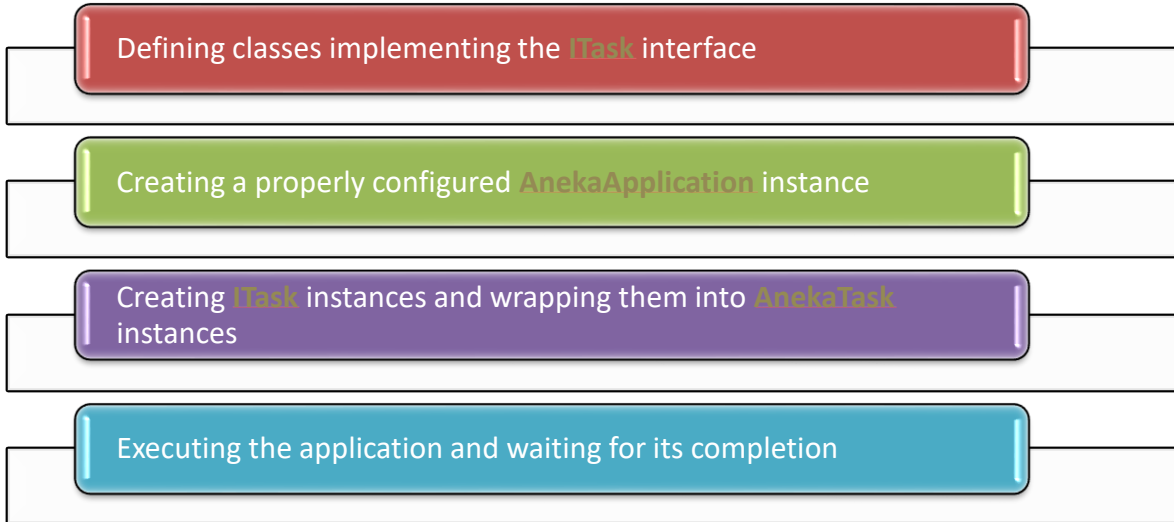
- ▶ All client side features are contained in the **Aneka.Tasks** namespace. (**Aneka.Tasks.dll**)

Aneka Task

- ▶ AnekaTask : manages tasks within Aneka
- ▶ Extends Aneka.Entity.WorkUnit class
- ▶ Provides features for embedding ITask instances
- ▶ AnekaTask: used internally

- ▶ For end users : provides facility for specifying input and output files
- ▶ ITask interface exposes only 1 method : **execute**(invoked to execute task in remote node.
- ▶ ITask provides support for native tasks – programming languages supported by .NET framework.
- ▶ Task instances need to be **serializable** : as they are created and moved over the network.

Developing applications with the task model



Controlling Task Execution:

- ▶ Operations provided to support different programming models
 - ▶ Static and dynamic task submission
 - ▶ Application state and task state monitoring
 - ▶ Event-based notification of task completion or failure

File Management

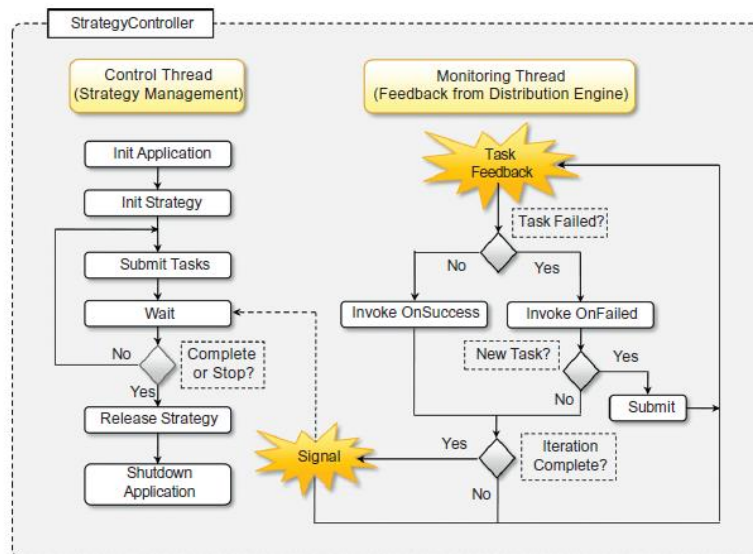
- ▶ Any model based on the WorkUnit and ApplicationBase classes has built-in support for file.
- ▶ It is possible to provide input files that are common to all the WorkUnit instances
- ▶ A FileData instance provides information about a file:
 - ▶ Its nature: whether it is a shared file, an input file, or an output file
 - ▶ Its path both in the local and in the remote file system, including a different name
 - ▶ A collection of attributes that provides other information (such as the final destination of the file or whether the file is transient or not, etc.)

c. Discuss how workflows are managed in Aneka with required diagram. 6m

Ans

- ▶ **A workflow is an automation of a business process, in whole or in 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.**
- ▶ Support for workflow in Aneka is not native but is obtained with plug-ins that allow client-based workflow managers to submit tasks to Aneka
- ▶ Two different workflow managers can leverage Aneka for task execution:
 - ▶ WorkflowEngine : leverages the task submission Web service exposed by Aneka
 - ▶ Offspring : the latter directly interacts with the Aneka programming APIs.
- ▶ The connection with Aneka is realized through the AnekaEngine : implements the operations of IDistributionEngine for the Aneka middleware and relies on the services exposed by the task model programming APIs.
- ▶ The system allows for the execution of a dynamic workflow the structure of which is defined as the workflow executes.

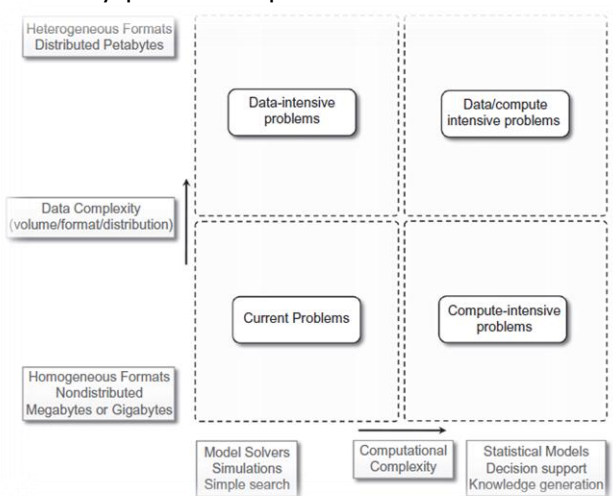
- ▶ Two different types of tasks can be defined
 - ▶ Native tasks and legacy tasks.
 - ▶ Native tasks are completely implemented in managed code.
 - ▶ Legacy tasks manage file dependencies and wrap all the data necessary for the execution of legacy programs on a remote node.
- ▶ a strategy may define shared file dependencies that are necessary to all the tasks generated by the workflow
- ▶ The dependencies among tasks are implicitly defined by the execution of the strategy by the StrategyController and the events fired by the distributed engine



- ▶ Two main execution threads control the execution of a strategy.
 - ▶ A control thread manages the execution of the strategy
 - ▶ 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.
 - ▶ 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
- ▶ If there are more tasks to be executed, they are submitted and the controller waits for feedback from the engine; otherwise, an iteration of the strategy is completed.
- ▶ 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.
- ▶ The AnekaEngine creates an instance of the AnekaApplication class for each execution of a strategy and configures the template class with a specific implementation of the TaskManager, which overrides the behaviour implemented for file management and optimizes the staging of output files.

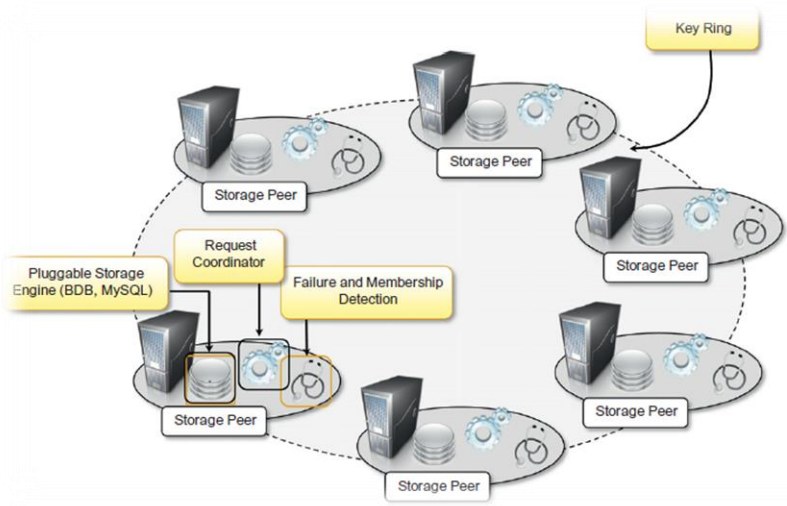
7 a. What is data intensive computing? Explain Amazon Dynamo architecture with a neat diagram. 10m

- ▶ **Concerned with the production, manipulation and analysis of large-scale data in the range of hundreds of megabytes to petabytes of data.** (Gorton et al 2010)
- ▶ **Dataset** : a collection of information elements
- ▶ **Repository** : where the datasets reside
- ▶ **Metadata** : information about the data in a dataset that is used for searching / filtering.
- ▶ Data-intensive applications not only deal with **huge volumes** of data but, very often, also exhibit **compute-intensive** properties
- ▶ Datasets are commonly persisted in **several formats** and **distributed** across different locations.
- ▶ Processing requirements **scale linearly** with the data size and they can be easily processed parallel.



Amazon Dynamo architecture

- ▶ is composed of a collection of storage peers organized in a ring that share same key space.
- ▶ Key space : partitioned among the storage 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



- ▶ Dynamo implements the capability of being an **always-writable** store, where consistency of data is resolved in the background
- ▶ which requires applications to build their own data models on top of simple building blocks provided by the store.
- ▶ there are no **referential integrity constraints**, relationships are not

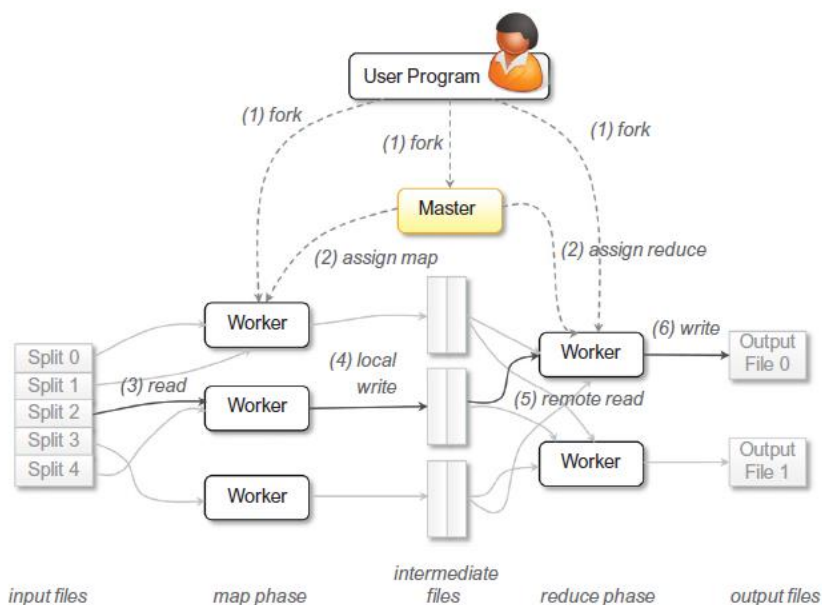
embedded in the storage model, and therefore **join operations** are not supported

- ▶ is the distributed key-value store that supports the management of information of several of the business services offered by Amazon Inc.
- ▶ **Goal** : provide an incrementally scalable and highly available storage system
 - ▶ To reach reliability at a massive scale.
 - ▶ Serve 10 million requests per day.
- ▶ **Interface** : get/put semantics
- ▶ **ACID properties** are sacrificed for a more reliable and efficient infrastructure.
- ▶ Creates an **eventually consistent model** : in the long term, all users will see the same data.

b. Explain Map-reduce computation with a neat diagram

10m

Ans



When user invokes MapReduce program the following sequence of events take place:

- (1) The MapReduce program in the user program first splits input files into M pieces and starts to make copies of the program on clusters of machines
- (2) One of the copies is the master and the rest are workers. There are M map tasks and R reduce tasks to assign.
- (3) A worker assigned a map task reads contents of corresponding split input. It parses out key-value pairs and passes the pair to user-defined Map function. The intermediate key-value is buffered in memory.
- (4) Periodically, the buffered pairs are written to local disk partitioned into R regions by the partitioning algorithm.
- (5) The reduce works read these intermediary data and sorts it by the intermediary keys such that same key are grouped together.
- (6) reduce worker iterates over sorted intermediary data. For each key, it passes intermediary values to the Reduce function. The output of Reduce function is appended to final output file.
- (7) When all map and reduce tasks are completed, master wakes up user program

11 a. Discuss the variations and extensions of map reduce.

6m

Ans

Variations

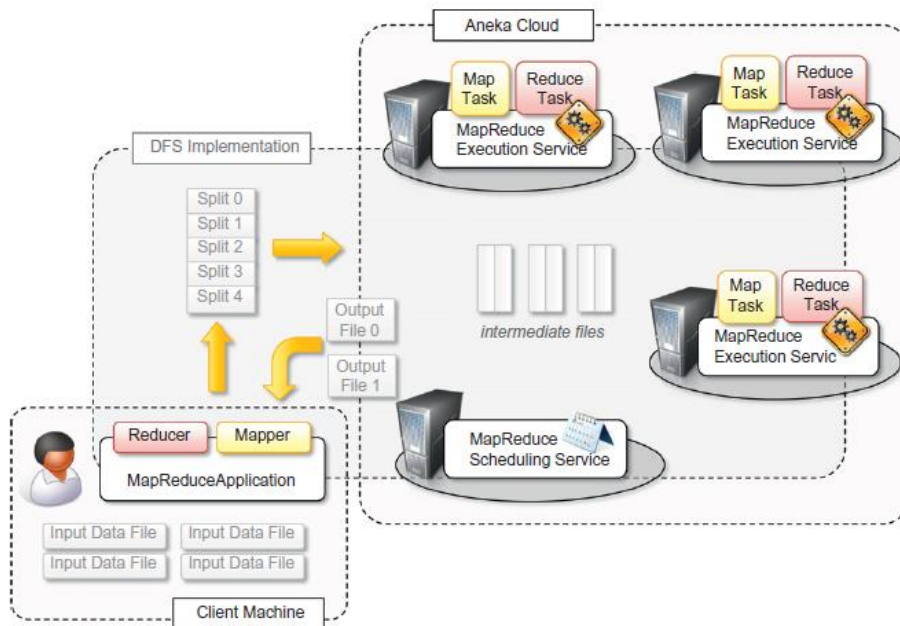
- Hadoop
- Pig
- Hive
- Map-reduce-merge
- Twister

- ▶ **Hadoop** : is a collection of software projects for reliable and scalable distributed computing
- ▶ Consists of Hadoop Distributed File System (HDFS) and Hadoop MapReduce
- ▶ Initially developed and supported by Yahoo!
- ▶ **Pig** : allows the analysis of large datasets
- ▶ Has high-level language for expressing data analysis program + infrastructure
- ▶ Infrastructure layer : compiler for high level languages that produce a sequence of MapReduce jobs.
 - ▶ Run on top of distributed infrastructures such as Hadoop.
- ▶ Developers can express data analytics program using Pig Latin – exposes SQL like interface characterized by
 - ▶ major expressiveness
 - ▶ Reduced programming effort
 - ▶ Familiar interface w.r.t. MapReduce.
- ▶ **Hive** : provides a data warehouse infrastructure on top of Hadoop MapReduce.
- ▶ Provides tools for easy data summarization, ad hoc queries, and analysis of large data sets stored in Hadoop Map Reduce files
- ▶ **Map-Reduce-Merge** : 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.
- ▶ **Twister**: is an extension of the MapReduce model that allows the creation of iterative executions of MapReduce jobs.
 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

b. Describe Aneka map reduce infrastructure with a neat diagram.

8m

Ans

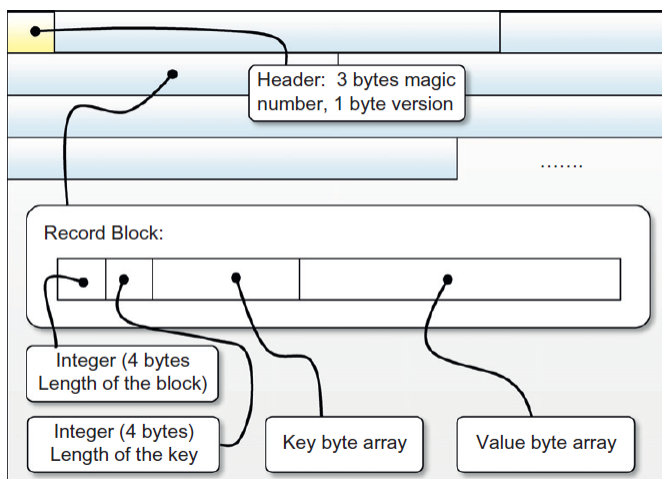


- ▶ the abstractions and runtime support for developing MapReduce applications on top of Aneka
- ▶ A **MapReduce job** in Google MapReduce or Hadoop corresponds to the execution of a **MapReduce application** in Aneka
- ▶ The application instance has components identify map and reduce functions
- ▶ Expressed as **Mapper** and **Reducer** classes
- ▶ 3 main elements :
 - ▶ **MapReduce Scheduling Service**
 - ▶ **MapReduce Execution Service**
 - ▶ specialized **distributed file system**
- ▶ **Client components** : MapReduceApplication is used to submit execution of a MapReduce job
- ▶ Task creation is taken care of by the application once user defines the map and reduce functions.
- ▶ **Mapper<K,V>**
- ▶ **Reducer<K,V>**
- ▶ **MapReduceApplication<M,R>**
- ▶ Other classes are internally used
- ▶ **Mapper<K,V> and Reducer<K,V>** are the starting point
- ▶ **Template specialization** keeps track of key and value types.
- ▶ **Generics** provide a natural method to object manipulation
- ▶ Simplify necessity of type checking and casting.
- ▶ Submission and execution of MapReduce job is performed through **MapReduceApplication<M,R>**
- ▶ map operation is implemented by overriding the abstract method **void Map(IMapInput<K,V> input)**
- ▶ **IMapInput<K,V>**
- ▶ provides access to the input key-value pair on which map operation is performed
- ▶ Requires overriding the abstract method **Reduce (IReduceInputEnumerator<V> input)**
- ▶ reduce operation is applied to a **collection of values** that are mapped to the same **key**
- ▶ the **IReduceInputEnumerator<V>** allows developers to iterate over such collections.
- ▶ MapReduceApplication<M,R> To submit, execute, and monitor the progress
- ▶ The interface of the class only exhibits MapReduce specific settings.
- ▶ Control logic is encapsulated in ApplicationBase<M>
- ▶ From this class, it is possible to set behaviour of MapReduce for current execution.

c. Discuss distributed file system support for execution of map reduce job with a neat diagram **6m**

- ▶ Does not leverage the default Storage service and data transfer.
- ▶ It uses **distributed file system** implementation.
- ▶ Because requirements for file management are significantly different w.r.t. other models.
- ▶ MapReduce is specifically designed to process large quantities of data stored in files of large dimensions.
- ▶ DFS guarantee **high availability** and **better efficiency** by means of data replication and distribution.
- ▶ Original MapReduce implementation : assumes the existence of distributed and reliable storage.
- ▶ **Retrieve location of file and file chunks** : used by the scheduler to optimize and schedule map and reduce tasks according to the location of data.
- ▶ **Access a file by means of a stream**: required for usual I/O operations to and from data files.
- ▶ In DFS, **stream** might also access a **network** if file chunk is not stored on a local node.
- ▶ Aneka provides **interfaces** that allow performing such operations.
- ▶ Provides capability to plug different file systems behind them by providing appropriate implementation.
- ▶ Current implementation provides binding to HDFS.

SeqReader and SeqWriter



- ▶ Aneka MapReduce file
- ▶ Header – 4 bytes
 - ▶ First 3 bytes represent SEQ
 - ▶ 4th byte identifies version of the file.
- ▶ Record block
 - ▶ First 8 bytes : represents length of the rest of the block
 - ▶ Remaining portion: stores the key value component of the pair.

MODULE 5

9 a. Discuss the storage services provided by AWS **12m**

- Ans
- ▶ S3 Key concepts: accessible through Representational State Transfer (REST) interface
 - ▶ The storage is organized in a two-level hierarchy
 - ▶ Stored objects cannot be manipulated like standard files
 - ▶ Content is not immediately available to users.
 - ▶ Requests will occasionally fail.

Resource Naming:

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

Subdomain form: <http://bucketname.s3.amazonaws.com/>.

Virtual hosting form: <http://bucket-name.com/>.

Buckets : A bucket is a container of objects.

- ▶ virtual drive hosted on the S3 distributed storage

- ▶ provides users with a flat store to which they can add objects.
- ▶ A bucket is located in a specific geographic location eventually replicated fault tolerance and better content distribution.
- ▶ Users create a bucket by sending a PUT request to <http://s3.amazonaws.com/> with the name of the bucket and,
- ▶ They may want to specify the availability zone, additional information about the preferred location.
- ▶ Content of a bucket can be listed by sending a GET request
- ▶ The deletion of a bucket is performed by a DELETE request

- ▶ **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
- ▶ name cannot be longer than **1,024 bytes** when encoded in **UTF-8**, and it allows almost **any character**.
- ▶ buckets do not support nesting
 - ▶ characters normally used **as path separators** are allowed.
 - ▶ This actually compensates for the lack of a **structured file system**
 - ▶ directories can be **emulated** by properly naming objects.
- ▶ Objects can be tagged with **metadata**, which are passed as **properties** of the **PUT request**
- ▶ **Access Control Policies (ACPs)** : An ACP is a **set of grant permissions**.
- ▶ attached to a resource expressed by means of **XML configuration file**.
- ▶ A policy allows defining up to **100 access rules**
- ▶ Each grants one of the available permissions to a grantee.
 - ▶ READ
 - ▶ WRITE
 - ▶ READ_ACP
 - ▶ WRITE_ACP
 - ▶ FULL_CONTROL
- ▶ Grantees can be either **single users or groups**: users can be identified by their **canonical IDs** or **email addresses** used for sign up

Amazon Elastic Block Store(EBS)

- ▶ They accommodate up to **1 TB** of space
- ▶ accessed through a **block device interface**
- ▶ Allows users to format them according to the needs of the instance they are connected to (raw storage, file system, or other)
- ▶ content of an **EBS volume survives** the instance life cycle and is persisted into **S3**
- ▶ EBS volumes can be cloned, used as boot partitions, and constitute durable storage

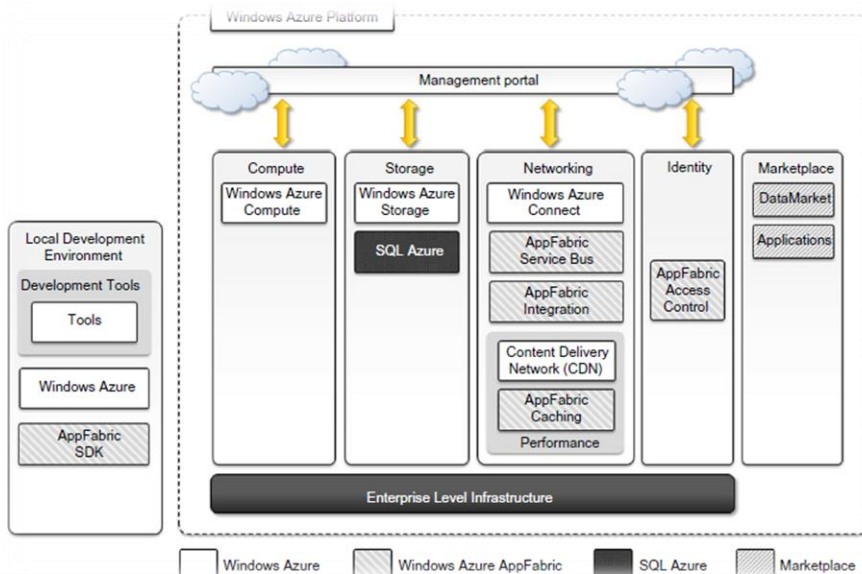
Amazon ElastiCache

- ▶ ElastiCache is an implementation of an **elastic memory cache based on a cluster of EC2 instances**.
- ▶ provides fast data access from other **EC2 instances** through a **Memcached-compatible protocol**.
- ▶ based on a **cluster of EC2 instances** running the caching software available through Web services
- ▶ An ElastiCache cluster can be **dynamically resized** according to the demand of the client applications.
- ▶ automatic **patch management** and **failure detection** and recovery of **cache nodes**

Structured Storage solutions

- ▶ **Preconfigured EC2 AMIs**: predefined templates featuring an installation of a given database management system
- ▶ **Amazon RDS**: A relational database service that relies on the EC2 infrastructure and is managed by Amazon

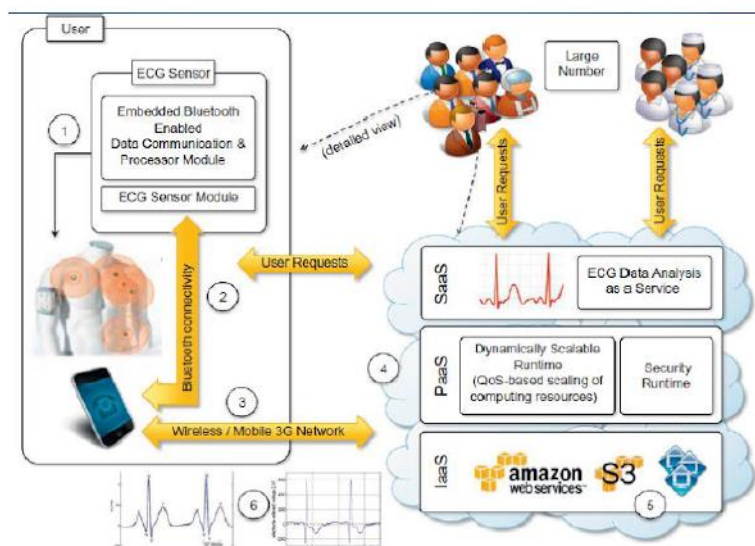
	<ul style="list-style-type: none"> ▶ 2 features : multi-AZ deployment and read replicas ▶ Optimal for Oracle and MySQL migrated to AWS ▶ Amazon SimpleDB: a lightweight, highly scalable, and flexible data storage solution for applications that do not require a fully relational model for their data <ul style="list-style-type: none"> ▶ semistructured data ▶ Based on concepts of domains, items, and attributes ▶ Provides fewer constraints on the structure of data entries which leads to improved performance. <p>SimpleDB</p> <ul style="list-style-type: none"> ▶ domains as top-level elements to organize a data store ▶ Domains are comparable to tables. ▶ Unlike tables they allow all items to have the same column structure. ▶ Each item is represented as a collection of attributes expressed as a key-value pair. ▶ Each domain can grow up to 10 GB ▶ A single user can allocate a maximum of 250 domains ▶ Clients can create, delete, modify and make snapshots of domain. ▶ Select clause supports the following test operators : =, !=, <, >, <=, >=, like, not like, between, is null, is not null, isempty() ▶ select from domain_name where every(attribute_name)='value' ▶ The select query can extend its boundaries of a single domain, hence querying a huge amount of data. 	
b.	Explain SQL Azure architecture with a neat diagram	8m
Ans	<ul style="list-style-type: none"> ▶ 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. ▶ This scalable computing environment is hosted within Microsoft datacenters and accessible through the Windows Azure Management Portal. ▶ Alternatively, developers can recreate a Windows Azure environment (with limited capabilities) on their own machines for development and testing purposes. <p>Compute Services</p> <ul style="list-style-type: none"> ▶ Web Role – hosted in IIS 7 web server. <ul style="list-style-type: none"> ▶ designed to implement scalable Web applications. ▶ Web roles represent the units of deployment of Web applications within the Azure infrastructure ▶ NET technology natively supports Web roles ▶ It is possible to develop ASP.NET (ASP.NET Web Role and ASP.NET MVC 2 Web Role) and WCF (WCF Service Web Role) applications. ▶ Worker role - designed to host compute services in Azure. <ul style="list-style-type: none"> ▶ They can be used to quickly provide compute power or to host services that do not communicate with the external world through HTTP ▶ Developing a worker role is like a developing a service. ▶ 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 – gives a finer grained control over the virtual resources.
- ▶ 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)
- ▶ upload it to Windows Azure to create compute instances on demand.

10 a. Describe how cloud computing can be applied to remote ECG monitoring with a required diagram. 10m

Ans



- (1) The patient is equipped with a wireless ECG sensor attached to their body and mobile device capable of connecting to internet.
- (2) The wireless ECG sensor module collects patients data and forwards to mobile device via bluetooth without user intervention
- (3) A client software in mobile device transmits data to ECG analysis web service in the cloud.
- (4) The analysis software carries out computations with historical record which involves comparison, classification, diagnosis of heartbeat.
- (5) Software appends latest results to the patient's historic record in a private cloud storage. Physicians will then determine based on features extracted whether heartbeat is normal, i.e. a healthy sinus

rhythm
 (6) The diagnosis is disseminated to patients mobile device and doctor/emergency services at predefined intervals
 (7) monitoring and computing process is repeated according to user's preference, hourly, daily or monthly or over a longer period of time.

b. Explain CRM and ERP implementations with three examples and the required diagrams 10m

Ans Customer relationship management (CRM) and enterprise resource planning(ERP) applications are market segments that are flourishing in the cloud, with CRM applications the more mature of the two.
 Cloud CRM applications constitute a great opportunity for small enterprises and start-ups to have fully functional CRM software without large up-front costs and by paying subscriptions
 ERP systems integrate several aspects of an enterprise: finance and accounting, human resources, manufacturing, supply chain management, project management, and CRM. Their goal is to provide a uniform view and access to all operations that need to be performed to sustain a complex organization.

SalesForce.com
 the most popular and developed CRM solution available today. The application provides customizable CRM solutions that can be integrated with additional features developed by third parties.

the platform has evolved to support the entire life cycle of a wider range of cloud applications by implementing a flexible and scalable infrastructure. 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 despite the large amounts of data that need to be crawled. Users can customize their application by leveraging the "native" Force.com application frame- work or by using programmatic APIs in the most popular programming languages.

Microsoft dynamics CRM

The system is completely hosted in Microsoft’s datacenters across the world and offers to customers a 99.9% SLA, with bonus credits if the system does not fulfill the agreement. 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 programmatically 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.

Dynamics CRM can also leverage the capability of Windows Azure for the development and integration of new features.

NetSuite

NetSuite provides a collection of applications that help customers manage every aspect of the business enterprise. 3 major products :

- ▶ NetSuite Global ERP
- ▶ NetSuite Global CRM1
- ▶ NetSuite Global Ecommerce

An all-in-one solution: NetSuite One World

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

NetSuite also provides an infrastructure and a development environment for implementing customized applications.

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.

NetSuite Business Suite components offer accounting, ERP, CRM, and ecommerce capabilities

SuiteFlex, allows integrating such capabilities into new Web applications, which are then packaged for distribution by SuiteBundler.

The entire infrastructure is hosted in the NetSuite datacenters - which provide warranties regarding application uptime and availability.