**Cloud Computing -SOLUTION**

**SECTION – A**

**1-What are the major challenges faced in cloud?**

Ans. 1-Security and privacy

2- Lock-In

3- Isolation failure

4- Management Interface Compromise

5- Insecure or incomplete data deletion

6-Consistent Performance

7-Converting Back Office Activities

8-Cost Calculations within Limits

9-Migrating Existing Applications onto the Cloud

10-Working with Poly Cloud Environments

11-Optimizing Cloud Expenses

**2-List the characteristics of cloud computing?**

Ans.2

**Characteristics of Cloud Computing**

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



**3- Write down any two advantages of SaaS?**

The advantages of Software As a Service are as:

**1.Reduced time to benefit/ Time-to-value**

**2. Lower costs/ Cheap cost –Licsnses, Maintenance, Administration, Hardware**

**3. Scalability and integration/ highly Scalable**

**4. New releases (upgrades)**

**5. Easy to use and perform proof of concepts**

**4-What is IaaS?**

Infrastructure as a service (IaaS) is a type of cloud computing model that allocates virtualized computing resources to the user through the internet. IaaS is one of the main components of cloud computing along with software as a service (SaaS) and platform as a service (PaaS). IaaS is completely provisioned and managed over the internet.

**5- What is HATEOAS?**

Hypermedia as the Engine of Application State (HATEOAS) is a component of the REST application architecture that distinguishes it from other network application architectures.

With HATEOAS, a client interacts with a network application whose application servers provide information dynamically through hypermedia.

**SECTION -B**

**2-(a) What do you understand by service oriented architecture (SOA)? How it support cloud computing?**

## Service-Oriented Architecture (SOA)

Service-Oriented Architecture helps to use applications as a service for other applications regardless the type of vendor, product or technology. Therefore, it is possible to exchange the data between applications of different vendors without additional programming or making changes to services.

The cloud computing service oriented architecture is shown in the diagram below.

* SOA uses interfaces which solves the difficult integration problems in large systems.
* SOA communicates customers, providers and suppliers with messages by using the XML schema.
* It uses the message monitoring to improve the performance measurement and detects the security attacks.
* As it reuses the service, there will be lower software development and management costs.
*  SOA has capability to adjust or modify the different external environments and large applications can be managed easily.
*  The companies can develop applications without replacing the existing applications.
*  It provides reliable applications in which you can test and debug the independent services easily as compared to large number of code.

**(b) What is the difference between process virtual machines, host VMMs and native VMMs.**

A **Process virtual machine**, sometimes called an application virtual machine, runs as a normal application inside a host OS and supports a single process. It is created when that process is started and destroyed when it exits. Its purpose is to provide a platform-independent programming environment that abstracts away details of the underlying hardware or operating system, and allows a program to execute in the same way on any platform. For example Wine software in Linux helps to run Windows application.

A **Host virtual machine** is the server component of a virtual machine , which provides computing resources in the underlying hardware to support guest virtual machine (guest VM).

Host virtual machines are created and hosted entirely on a cloud service provider infrastructure and are available to remote users over the Internet under systematic access control, defined compute and I/O resources. Host virtual machines exhibit the same functionalities as those of a guest virtual machine but are accessed in a different way.

**A hypervisor or VMM** is virtualization software that allows multiple operating systems (OS) to share the resources of the same physical server. This physical server is known as the host server. **Hypervisors** can be categorized into two main types. **The first one is called a native, bare metal or type 1 hypervisor**. The second type is called a type 2 or hosted hypervisor. a bare metal hypervisor runs directly on the physical server. It has control over the server’s resources and can allocate resources to guest operating systems or VMs as required. VMs run on top the bare metal hypervisor. Examples of bare metal hypervisors are IBM z/VM, Microsoft Hyper-V, VMWare ESXi and Citrix XenServer. It is important to remember that there are two layers of software stack in a bare metal hypervisor.

.

**(c) Briefly explain the overview of Software as Service (SaaS) and advantage of SaaS?**

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

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

Some of the SaaS applications are not customizable such as **Microsoft Office Suite.** But SaaS provides us **Application Programming Interface (API),** which allows the developer to develop a customized application.

**Characteristics**

Here are the characteristics of SaaS service model:

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

**Advantages**

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

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

**(d) Explain the objectives of Cloud lifecycles. Difference between PaaS and Iaas?**

The goal of cloud lifecycle management is to manage the dynamic nature of the cloud environment, accelerating provisioning, facilitating flexibility, and rapidly meeting the needs of the business . With the Cloud Lifecycle Management solution, organizations can deliver flexible, customizable cloud services while maintaining a structured, controlled, and dynamic IT environment. The key benefits of a cloud lifecycle management solution should include: Accelerating the delivery of cloud services in response to business needs»Automating provisioning and workflows, both for speed and cost savings»Enabling users to request flexible configurable cloud services for their specific use cases»Supporting the use of public cloud infrastructures to augment internal resources»Maximizing resource utilization by ensuring unused cloud services are reclaimed»Initial decisions around cloud lifecycle management will help lay the foundation for the technology decisions going forward – ensuring that the environment is flexible enough to address anticipated areas of growth in the future.



 The cloud lifecycle starts with a user needing to request a service from the cloud . Requestors need a self-service portal by which they can request, augment, and retire their cloud services . Driving the portal is a service catalog that aggregates the offerings available, by role, to the users . Once a request is initiated, a workflow is invoked, either fully automated or with manual steps, depending on the request and on the organization’s needs . Once approved, the service is automatically provisioned .A cloud requires full-stack provisioning — server, storage, and network resources, as well as middleware, applications and other software elements — in order to exist . This provisioning can be in any environment (virtual, cloud, or even physical) and spans server, network, and storage resources . Once provisioned, the service enters its operational phase, where the normal day-to-day activities of performance, capacity, and compliance are managed . Once a cloud service is no longer needed, users will need a mechanism to decommission that resource . Good decommissioning systems operate on-demand, at the user’s explicit request, or according to a pre-determined schedule .

**SECTION - C**

**3- (a) What is hypervisor? Discuss hardware virtualization in your organization.**

 Hypervisor is a form of virtualization software used in Cloud hosting to divide and allocate the resources on various pieces of hardware. The program which provides partitioning, isolation or abstraction is called virtualization hypervisor. Hypervisor is a hardware virtualization technique that allows multiple guest operating systems (OS) to run on a single host system at the same time. A hypervisor is sometimes also called a virtual machine manager (VMM).

**Types of Hypervisor –**

**TYPE-1 Hypervisor:**
Hypervisor runs directly on underlying host system. It is also known as “Native Hypervisor” or “Bare metal hypervisor”. It does not require any base server operating system. It has direct access to hardware resources. Examples of Type 1 hypervisors include VMware ESXi, Citrix XenServer and Microsoft Hyper-V hypervisor.



**TYPE-2 Hypervisor:**
A Host operating system runs on undrlying host system. It is also known as ‘Hosted Hypervisor”. Basically a software installed on an operating system. Hypervisor asks operating system to make hardware calls. Example of Type 2 hypervisor include VMware Player or Parallels Desktop. Hosted hypervisors are often found on endpoints like PCs.

 

**Type 1 hypervisors offer much better performance than Type 2** ones because there’s no middle layer, making them the logical choice for mission-critical applications and workloads. But that’s not to say that hosted hypervisors don’t have their place – they’re much simpler to set up, so they’re a good bet if, say, you need to deploy a test environment quickly. One of the best ways to determine which hypervisor meets your needs is to compare their performance metrics. These include CPU overhead, amount of maximum host and guest memory, and support for virtual processors. The following factors should be examined before choosing a suitable hypervisor:

 The company or in organization’s applications are the reason for the data center besides our our organization’s need - co-workers also have needs for individual system without expanding physical machines. This limitation can be sort by using hardware virtualization. A virtualization hypervisor are:

a. Flexibility
b. Scalability
c. Usability
d. Availability
e. Reliability
f. Efficiency
g. Reliable support

**2. The cost of a hypervisor:** For many buyers, the toughest part of choosing a hypervisor is striking the right balance between cost and functionality. While a number of entry-level solutions are free, or practically free, the prices at the opposite end of the market can be staggering. Licensing frameworks also vary, so it’s important to be aware of exactly what you’re getting for your money.

**3. Virtual machine performance:** Virtual systems should meet or exceed the performance of their physical counterparts, at least in relation to the applications within each server. Everything beyond meeting this benchmark is profit.

**4. Ecosystem:** It’s tempting to overlook the role of a hypervisor’s ecosystem – that is, the availability of documentation, support, training, third-party developers and consultancies, and so on – in determining whether or not a solution is cost-effective in the long term.

**5. Test for yourself:** You can gain basic experience from your existing desktop or laptop. You can run both VMware vSphere and Microsoft Hyper-V in either VMware Workstation or VMware Fusion to create a nice virtual learning and testing environment.

There are 3 main modues coordinate in order to emiulate the undrelying hardware:

1. Dispatcher
2. Allocator
3. Interpreter

**3-(b) Explain bare system architecture and compare with host architecture?**

**Type 1 Hypervisor is called a native or Bare Metal Hypervisor** while **type 2 Hypervisor is called a Host OS Hypervisor.**

**Type 1 hypervisor (bare system architecture)** is a hypervisor that runs directly on the host’s hardware to control the hardware and to manage guest operating systems while **Type 2 hypervisors (host architecture)** run on a conventional operating system just as other computer programs do. Thus, this is the main difference between Type 1 and Type 2 Hypervisor.

Functionality is another difference between Type 1 and Type 2 Hypervisor. Type 1 Hypervisor runs directly on the host’s hardware while Type 2 Hypervisor runs on an operating system similar to other computer programs.

AntsleOs, Xen, XCP-ng, Microsoft Hyper V, VMware ESX/ESXi, Oracle VM Server for x86 are some examples for Type 1 Hypervisors while VMware Workstation, VMware Player, VirtualBox, Parallel Desktop for Mac are some examples for Type 2 Hypervisors.

In conclusion, Hypervisor is capable of creating and executing virtual machines. There are two types of hypervisors as Type 1 and Type 2. The main difference between Type 1 and Type 2 Hypervisor is that Type 1 Hypervisor runs directly on the host’s hardware while Type 2 Hypervisor runs on an operating system similar to other computer programs.



 **‘Bare System Architecture’ ‘Host Architecture’**

**4-(a) Explain Iaas. How EC2 renting of Amazon works.**

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

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

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

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

Amazon Elastic Compute Cloud (Amazon EC2) is a web-based service that allows businesses to run application programs in the Amazon Web Services (AWS) public cloud. Amazon EC2 allows a developer to spin up virtual machines (VMs), which provide compute capacity for IT projects and cloud workloads that run with global AWS data centers. EC2 was the idea of engineer Chris Pinkham who conceived it as a way to scale Amazon's internal infrastructure.

To begin using EC2, developers sign up for an account at Amazon's AWS website. They can then use the AWS Management Console, the AWS Command Line Tools (CLI), or AWS Software Developer Kits (SDKs) to manage EC2.

A developer then chooses EC2 from the AWS Services dashboard and 'launch instance' in the EC2 console. At this point, they select either an Amazon Machine Image (AMI) template or create an AMI containing an operating system, application programs, and configuration settings. The AMI is then uploaded to the Amazon S3 and registered with Amazon EC2, creating an AMI identifier. Once this has been done, the subscriber can requisition virtual machines on an as-needed basis.

Data only remains on an EC2 instance while it is running, but a developer can use an Amazon Elastic Block Store volume for an extra level of durability and Amazon S3 for EC2 data backup.

VM Import/Export allows a developer to import on-premises virtual machine images to Amazon EC2, where they are turned into instances.

EC2 also offers Amazon Cloud Watch which monitors Amazon cloud applications and resources, allowing users to set alarms, view graphs, and get statistics for AWS data; and AWS Marketplace, an online store where users can buy and sell software that runs on AWS

Amazon EC2 provides different instance types, sizes and pricing structures designed for different computing and budgetary needs. In addition to general purpose instances, Amazon EC2 offers an instance type for compute, memory, accelerated computing, and storage-optimized workloads. AWS limits how many instances a user can run in a region at a time, depending on the type of instance. Each instance type comes with different size options corresponding to the CPU, memory and storage needs of each enterprise.

On-Demand instances allow a developer to create resources as needed and to pay for them by the hour. Reserved instances (RIs) provide a price discount in exchange for one and three-year contract commitments -- a developer can also opt for a convertible RI, which allows for the flexibility to change the instance type, operating system or tenancy. There's also an option to purchase a second-hand RI from the Amazon EC2 reserved instances marketplace. A developer can also submit a bid for spare Amazon EC2 capacity, called Spot instances, for a workload that has a flexible start and end time. If a business needs dedicated physical server space, a developer can opt for EC2 dedicated hosts, which charge hourly and let the business use existing server-bound software licenses, including Windows Server and SQL Server.

**4-(b) Discuss the case study of Saas.**

Cloud application services, or Software as a Service (SaaS), represent the largest cloud market and are still growing quickly. SaaS uses the web to deliver applications that are managed by a third-party vendor and whose interface is accessed on the clients’ side. Most SaaS applications can be run directly from a web browser without any downloads or installations required, although some require plugins.

Because of the web delivery model, SaaS eliminates the need to install and run applications on individual computers. With SaaS, it’s easy for enterprises to streamline their maintenance and support, because everything can be managed by vendors: applications, runtime, data, middleware, OSes, virtualization, servers, storage and networking.

Popular SaaS offering types include email and collaboration, customer relationship management, and healthcare-related applications. Some large enterprises that are not traditionally thought of as

software vendors have started building SaaS as an additional source of revenue in order to gain a competitive advantage.

**Google Docs as case study:**

Google Docs is a SaaS application that delivers the basic office automation capabilities with support for collaborative editing over the Web. The application is executed on top of the Google distributed computing infrastructure, which allows the system to dynamically scale according to the number of users using the service.

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. It supports collaborative editing over the Web for most of the applications included in the suite. This eliminates tedious emailing and synchronization tasks when documents need to be edited by multiple users. By being stored in the Google infrastructure, these documents are always available from anywhere and from any device that is connected to the Internet. Moreover, the suite allows users to work offline if Internet connectivity is not available. Support for various formats such as those that are produced by the most popular desktop office solutions allows users to easily import and move documents in and out of Google Docs, thus eliminating barriers to the use of this application.

Google Docs is a good example of what cloud computing can deliver to end users: ubiquitous access to resources, elasticity, absence of installation and maintenance costs, and delivery of core functionalities as a service.

There are three major types of cloud services: IaaS, PaaS, and SaaS. We can compare them by taking daily transportation as a simple case study example. IaaS, PaaS, and SaaS towards transportation:

* **On-premises IT infrastructure is like owning a car.**When you buy a car, you’re responsible for its maintenance, and upgrading means buying a new car.
* **IaaS is like leasing a car.**When you lease a car, you choose the car you want and drive it wherever you wish, but the car isn’t yours. Want an upgrade? Just lease a different car!
* **PaaS is like taking a taxi.**You don’t drive a taxi yourself, but simply tell the driver where you need to go and relax in the back seat.
* **SaaS is like going by bus.**Buses have assigned routes, and you share the ride with other passengers.

**5-(a) Discuss publish subscribe model with example?**

****

The Publish Subscribe model allows messages to be broadcast to different parts of a system asynchronously. A sibling to a message queue, a message topic provides a lightweight mechanism to broadcast asynchronous event notifications, and endpoints that allow software components to connect to the topic in order to send and receive those messages. To broadcast a message, a component called a publisher simply pushes a message to the topic.

Unlike message queues, which batch messages until they are retrieved, message topics transfer messages with no or very little queuing, and push them out immediately to all subscribers. All components that subscribe to the topic will receive every message that is broadcast, unless a message filtering policy is set by the subscriber.

The subscribers to the message topic often perform different functions, and can each do something different with the message in parallel. The publisher doesn’t need to know who is using the information that it is broadcasting, and the subscribers don’t need to know who the message comes from. This style of messaging is a bit different than message queues, where the component that sends the message often knows the destination it is sending to.

**5-(b) What is web services? Discuss principles of REST towards cloud computing.**

Web service is a standardized medium to propagate communication between the client and server applications on the World Wide Web.

A web service is a software module which is designed to perform a certain set of tasks.

* The web services can be searched for over the network and can also be invoked accordingly.
* When invoked the web service would be able to provide functionality to the client which invokes that web service.

Web Service Architecture Diagram

There are mainly two types of web services.

1. SOAP web services.
2. RESTful web services.

**SOAP (Simple Object Access Protocol)** SOAP is known as a transport-independent messaging protocol. SOAP is based on transferring XML data as SOAP Messages. Each message has something which is known as an XML document.

REST (REpresentational State Transfer) is an architectural style for developing web services. REST is popular due to its simplicity and the fact that it builds upon existing systems and features of the internet's Hypertext Transfer Protocol (HTTP) in order to achieve its objectives, as opposed to creating new standards, frameworks and technologies.

A primary benefit of using REST, both from a client and server's perspective, is REST-based interactions happen using constructs that are familiar to anyone who is accustomed to using the internet's HTTP.

An example of this arrangement is REST-based interactions all communicate their status using standard HTTP status codes. So, a 404 means a requested resource wasn't found; a 401 code means the request wasn't authorized; a 200 code means everything is OK; and a 500 means there was an unrecoverable application error on the server.

**Principles /Architecture of REST:-**

REST defines **6 architectural constraints/Principles** which make any web service – a true RESTful API.

1. Uniform interface
2. Client–server
3. Stateless
4. Cacheable
5. Layered system
6. Code on demand (optional)

#### Uniform interface

As the constraint name itself applies, you MUST decide APIs interface for resources inside the system which are exposed to API consumers and follow religiously. A resource in the system should have only one logical URI and that should provide a way to fetch related or additional data. It’s always better to **synonymise a resource with a web page**.

Any single resource should not be too large and contain each and everything in its representation. Whenever relevant, a resource should contain **links (HATEOAS) pointing to relative URIs** to fetch related information.

Also, the resource representations across system should follow certain guidelines such as naming conventions, link formats or data format (xml or/and json).

All resources should be accessible through a common approach such as HTTP GET and similarly modified using a consistent approach.

#### Client–server

This essentially means that client application and server application MUST be able to evolve separately without any dependency on each other. A client should know only resource URIs and that’s all. Today, this is normal practice in web development so nothing fancy is required from your side. Keep it simple.

Servers and clients may also be replaced and developed independently, as long as the interface between them is not altered.

#### Stateless

Roy fielding got inspiration from HTTP, so it reflects in this constraint. Make all client-server interaction stateless. Server will not store anything about latest HTTP request client made. It will treat each and every request as new. No session, no history.

If client application needs to be a stateful application for the end user, where user logs in once and do other authorized operations thereafter, then each request from the client should contain all the information necessary to service the request – including authentication and authorization details.

No client context shall be stored on the server between requests. The client is responsible for managing the state of the application.

#### Cacheable

In today’s world, caching of data and responses is of utmost important wherever they are applicable/possible. The webpage you are reading here is also a cached version of the HTML page. Caching brings performance improvement for client side, and better scope for scalability for a server because the load has reduced.

In REST, caching shall be applied to resources when applicable and then these resources MUST declare themselves cacheable. Caching can be implemented on the server or client side.

Well-managed caching partially or completely eliminates some client-server interactions, further improving scalability and performance.

#### Layered system

REST allows you to use a layered system architecture where you deploy the APIs on server A, and store data on server B and authenticate requests in Server C, for example. A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way.

#### Code on demand (optional)

Well, this constraint is optional. Most of the time you will be sending the static representations of resources in form of XML or JSON. But when you need to, you are free to return executable code to support a part of your application e.g. clients may call your API to get a UI widget rendering code. It is permitted.