Cc merged

Cloud Computing

UNIT
4,5

- IaaS Recap
- VM Management
  - Why VM Management?
  - VM management Tools
  - VM Provisioning
  - VM Migration Techniques
  - Live Migration Example

What is OpenStack?

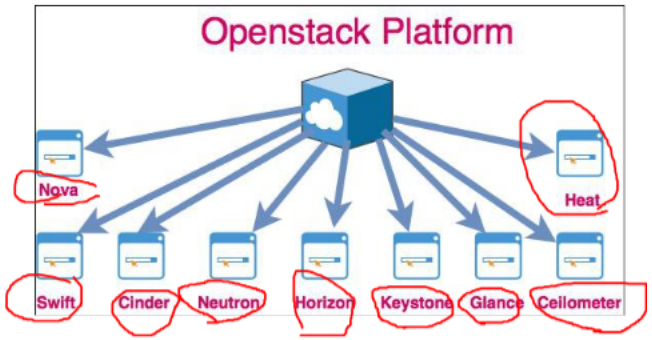
OpenStack is a **cloud operating system** that controls **large pools of compute, storage, and networking resources**.

Managed through **a dashboard** that gives administrators control while empowering their users to provision resources through a web interface.

OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds.

Backed by some of the biggest companies in software development and hosting, as well as thousands of individual community members, many think that OpenStack is the future of cloud computing.

Managed by **OpenStack Foundation a non profit organization**.



Differences

In principle, the main difference between AWS and OpenStack is that the former already exists, while the latter one you have to build yourself.

Since building cloud infrastructure from scratch entails significant upfront investments (setting up a data centre, hardware purchase, cloud deployment consulting fee, etc.), AWS is usually a more compelling option at the beginning of the cloud migration journey.

At the end of the day, all you need to do is to create an account on AWS and attach a credit card, and you can start using cloud resources right away.

AWS will charge you based on the actual resource consumption.

Differences

On the other hand, those costs(AWS) can quickly become significant as the number of workloads continues to increase.

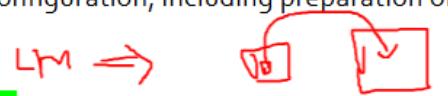
Over time, it may turn out that the aggregated recurring cost of using AWS will exceed the cost of OpenStack deployment.

Of course, running cloud infrastructure on-premises also comes with some recurring costs (hosting facilities, power consumption, staff salary, etc.), but these are lower when running workloads in the long-term and at scale, according to Canonical's Cloud Pricing Report from 2021.

Anatomy of Cloud

Virtual Infrastructure (VI) management—the management of virtual machines distributed across a pool of physical resources—becomes a key concern when building an IaaS cloud and **poses a number of challenges**.

In traditional physical resources, machines require a fair amount of configuration, including preparation of the machine's software environment and network configuration.



In a **virtual infrastructure**, this configuration must be done **on-the-fly,** with as little time between the time the VMs are requested and the time they are available to the users.

This is further complicated by the need to configure groups of VMs that will provide a specific service (e.g., an application requiring a Web server and a database server).

Additionally, a **virtual infrastructure manager** must be capable of **allocating resources efficiently**, taking into **account an organization's goals** (such as minimizing power consumption and other operational costs) and **reacting to changes in the physical infrastructure**.

Virtual infrastructure management in **private clouds** has to deal with an additional problem:

Unlike large IaaS cloud providers such as Amazon, **private clouds typically do not have enough** resources to provide the illusion of "infinite capacity."

The **immediate provisioning scheme used in public clouds**, where resources are provisioned at the moment they are requested, is **ineffective in private clouds**.

Support for additional provisioning schemes is required for applications that require resources at specific times, such as **best-effort provisioning** and **advance reservations** to guarantee quality of service (QoS).

Thus, **efficient resource allocation algorithms** and **policies** and the ability to combine both **private and public cloud resources**, resulting in a hybrid approach, become even more important.

What is Resource Management

RM is a process that deals with the procurement and release of resources.

Virtualization techniques are used for flexible and on-demand resource provisionir

To do so, for each received task, either a new VM is created or it is placed on the existing VM of the same user.

Once the task is completed, all the acquired resources are released which become parts of the free resource pool.

Resource assignment is performed on the basis of Service Level Agreement (SLA) that is agreed between the service provider and the customer.

SLA contains details of the service level that is required by a tenant. Moreover, it contains information about the payment process and SLA violation penalty.

What is Resource Management

Cloud resource management requires complex policies and decisions for multi-objective optimization.

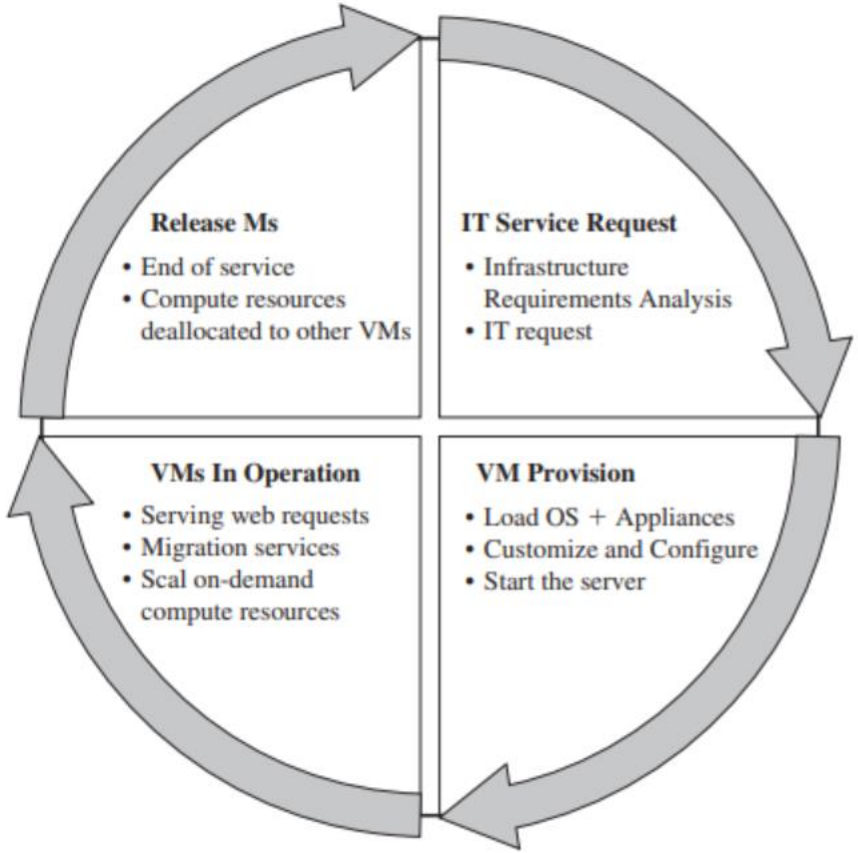
Effective resource management is extremely challenging due to the scale of the cloud infrastructure and to the unpredictable interactions of the system with a large population of users.

The scale makes it impossible to have accurate global state information and the large user population makes it nearly impossible to predict the type and the intensity of the system workload.

VM Life Cycle

Virtual Machine Life Cycle

- **The cycle starts by a request delivered to the IT department, stating the requirement for creating a new server for a particular service.**
- **This request is being processed by the IT administration to start seeing the servers' resource pool, matching these resources with requirements**
- **Starting the provision of the needed virtual machine.**
- **Once it provisioned and started, it is ready to provide the required service according to an SLA(Service Level agreement).**
- **Virtual is being released; and free resources.**



VM Provisioning process

- Server provisioning is defining server's configuration based on the organization requirements, a H/W, and S/W component (processor, RAM, storage, networking, operating system, applications, etc.).

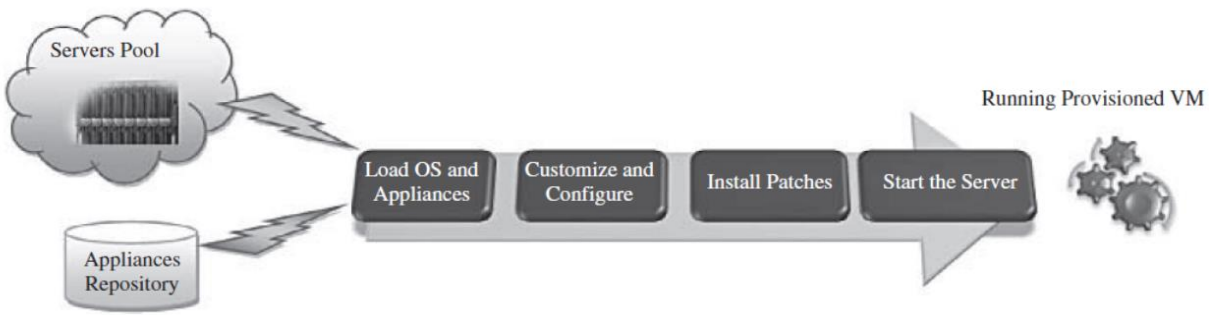
VMs can be provisioned by

- Manually installing an OS,
- Using a preconfigured VM template,
- Cloning an existing VM, or importing a physical server or a
- Server from another hosting platform.
- Physical servers can also be virtualized and provisioned using P2V (Physical to Virtual)

VM Provisioning process

Steps to Provision VM -

- Select a server from a pool of available servers along with the appropriate OS template you need to provision the virtual machine.
- Load the appropriate software.
- Customize and configure the machine (e.g., IP address, Gateway) to an associated network and storage resources.
- Finally, the virtual server is ready to start with its newly loaded S/W.



VM Provisioning using templates

After creating a virtual machine by virtualizing a physical server, or by building a new virtual server in the virtual environment, a template can be created out of it.

Most virtualization management vendors (VMware, XenServer, etc.) provide the data center's administration with the ability to do such tasks

Provisioning from a template reduces the time required to create a new virtual machine.

- Administrators can create different templates for different purposes.

For example –

- Vagrant provision tool using VagrantFile (template file) - Demo
- Heat – Orchestration Tool of openstack (Heat template in YAML format) - Demo

This enables the administrator to quickly provision a correctly configured virtual server on demand.

Provisioning from a template is an invaluable feature, because it reduces the time required to create a new virtual machine.

What is VM Migration

The process of moving a virtual machine from one host server or storage location to another;

There are different techniques of VM migration-

- - Hot/live migration,
- - Cold/regular migration, and
- - Live storage migration of a virtual machine.

In this process, all key machines' components, such as CPU, storage disks, networking, and memory, are completely virtualized, thereby facilitating the entire state of a virtual machine to be captured by a set of easily moved data files.

- Migration can be categorized as cold or non-live migration and live migration.
- Based on granularity, the migration can be divided into single and multiple migrations.
- The design and continuous optimization and improvement of live migration mechanisms are striving to minimize downtime and live migration time.
- The downtime is the time interval during the migration service is unavailable due to the need for synchronization.
- For a single migration, the migration time refers to the time interval between the start of the pre-migration phase to the finish of post-migration phases that instance is running at the destination host.
- On the other hand, the total migration time of multiple migrations is the time interval between the start of the first migration and the completion of the last migration.

What is VM Live Migration

Live migration can be defined as the movement of a virtual machine from one physical host to another while being powered on.

When it is properly carried out, this process takes place without any noticeable effect from the end user's point of view (a matter of milliseconds).

One of the most significant advantages of live migration is the fact that it facilitates proactive maintenance in case of failure, because the potential problem can be resolved before the disruption of service occurs.

Live migration can also be used for load balancing in which work is shared among computers in order to optimize the utilization of available CPU resources.

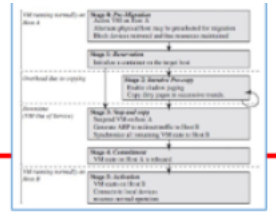
Live Migration Xen Hypervisor

The steps of live migration's mechanism and how memory and virtual machine states are being transferred, through the network, from one host A to another host B: •

The Xen hypervisor is an example for this mechanism.

The migration process has been viewed as a transactional interaction between the two hosts involved:

Live Migration Xen Hypervisor



Stage 0: Pre-Migration. An active virtual machine exists on the physical host A.

Stage 1: Reservation. A request is issued to migrate an OS from host A to host B (a precondition is that the necessary resources exist on B and on a VM container of that size).

Stage 2: Iterative Pre-Copy. During the first iteration, all pages are transferred from A to B. Subsequent iterations copy only those pages dirtied during the previous transfer phase.

Stage 3: Stop-and-Copy. Running OS instance at A is suspended, and its network traffic is redirected to B. CPU state and any remaining inconsistent memory pages are then transferred. At the end of this stage, there is a consistent suspended copy of the VM at both A and B. The copy at A is considered primary and is resumed in case of failure.

Stage 4: Commitment. Host B indicates to A that it has successfully received a consistent OS image. Host A acknowledges this message as a commitment of the migration transaction. Host A may now discard the original VM, and host B becomes the primary host.

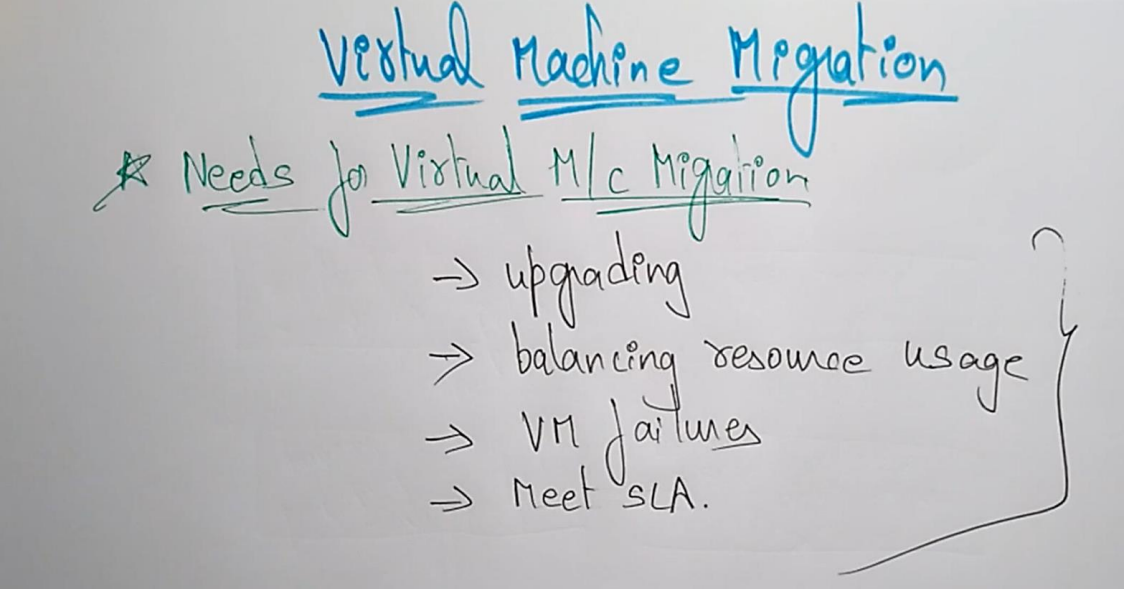
Stage 5: Activation. The migrated VM on B is now activated. Post-migration code runs to reattach the device's drivers to the new machine and advertise moved IP addresses.

FOR Safety- not there in portion

Live Migration Xen Hypervisor

Memory and storage transmission can be categorized into three phases:

- Push phase where the instance is still running in the source host while memory pages and disk block or writing data are pushed through the network to the destination host.
- Stop-and-Copy phase where the instance is stopped, and the memory pages or disk data is copied to the destination across the network. At the end of the phase, the instance will resume at the destination.
- Pull phase where the new instance executes while pulling faulted memory pages when it is unavailable in the source from the source host across the network.



Live Migration Technique

Post-migration code runs to reattach the device's drivers to the new machine and advertise moved IP addresses.

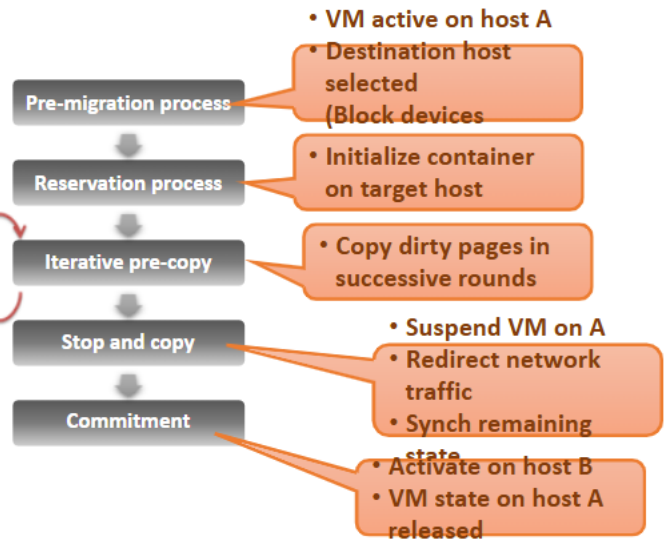
This approach to failure management ensures that at least one host has a consistent VM image at all times during migration:

1) Original host remains stable until migration commits and that the VM may be suspended and resumed on that host with no risk of failure.
2) A migration request essentially attempts to move the VM to a new host and on any sort of failure, execution is resumed locally, aborting the migration.

Challenges of live migration :

- VMs have lots of state in memory
- Some VMs have soft real-time requirements

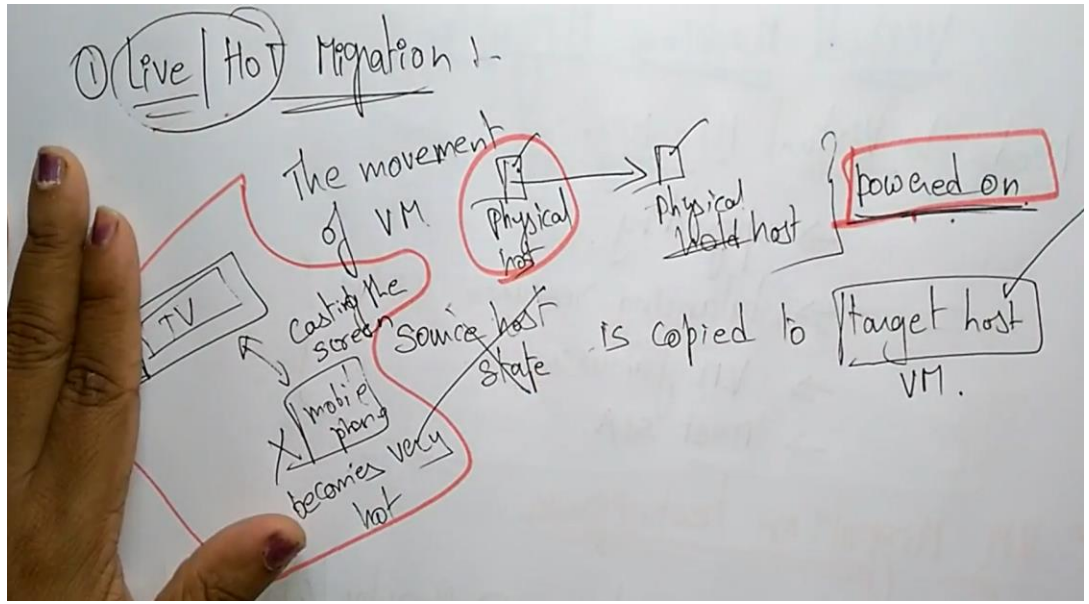
Live Migration Technique



Pre-migration and Post-migration phases are handling the computing and network configuration.

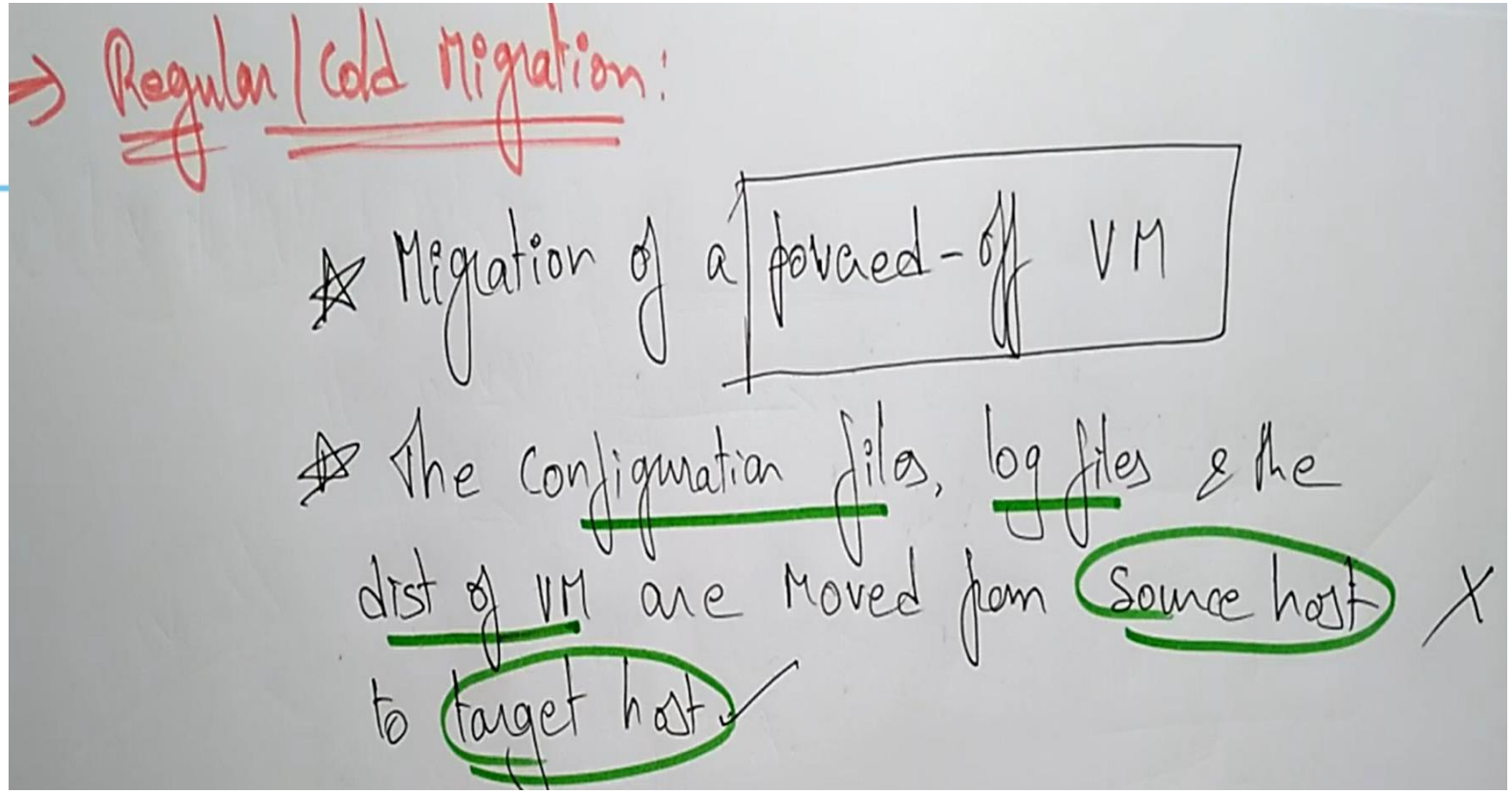
During the pre-migration phase, migration management software creates instance's virtual interfaces (VIFs) on the destination host, updates interface or ports binding, and networking management software, such as OpenStack Neutron server, configures the logical router.

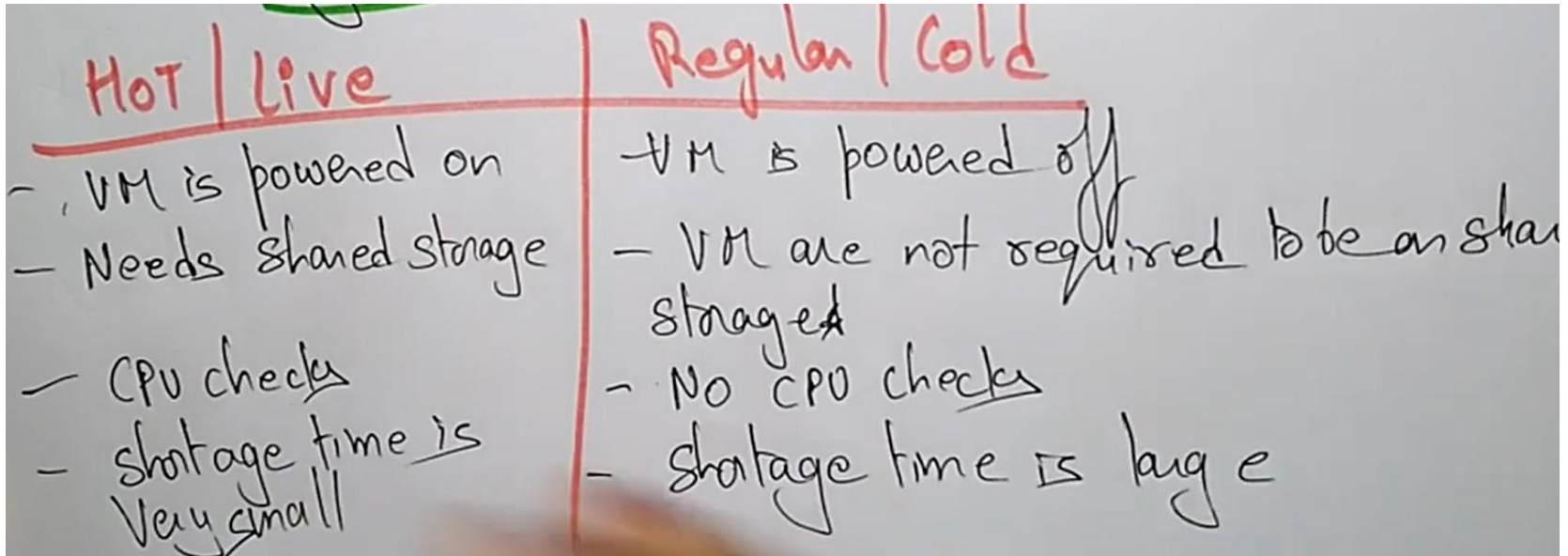
During the post-migration phase, migration management software updates port or interface states and rebinds the port with networking management software and the VIF driver unplugs the instance's virtual ports on the source host.



Cold Migration

- **Cold migration** is the migration of a powered-off virtual machine.
- With cold migration, you have the option of moving the associated disks from one data store to another. The virtual machines are not required to be on a shared storage.
- It's important to highlight that the **two main differences between live migration and cold migration** are that live migration **needs a shared storage** for virtual machines in the server's pool, but cold migration does not;
- Also, in live migration for a virtual machine between two hosts, there would be **certain CPU compatibility checks** to be applied; while in cold migration this checks do not apply.





Storage Migration

- This kind of migration constitutes moving the virtual disks or configuration file of a running virtual machine to a new data store without any interruption in the availability of the virtual machine's service
- 1. Throughout most of the move operation, disk reads and writes go to the source virtual hard disk.
- 2. While reads and writes occur on the source virtual hard disk, the disk contents are copied to the new destination virtual hard disk.
- 3. After the initial disk copy is complete, disk writes are mirrored to both the source and destination virtual hard disks while outstanding disk changes are replicated.
- 4. After the source and destination virtual hard disks are completely synchronized, the virtual machine switches over to using the destination virtual hard disk.
- 5 The source virtual hard disk is deleted.

CLOUD COMPUTING

UNIT 6

Agenda

- SaaS Recap
- Capacity management in Cloud computing
- Virtual Infrastructure management
- RESERVIOR Project
- Distributed management of virtual machines
- Reservation-based provisioning of virtualized resource
- Provisioning to meet SLA commitments
- Scheduling models and algorithms
- Haieza project

Software as a Service - Summary

Why SaaS

Capability

SaaS provides the following
- Hosted , Finished Product
- Subscription to Services

Characteristics

Not owned but subscribed to from an external service provider.
Designed to be Multi Tenant
Customer Configuration instead of Application configuration
Centralized management

Enabler : Web Service
✓ Accessibility & Portability

Models

SaaS can be obtained as
1. Managed Service
2. Monthly Subscription

Benefit

No need to purchase Software or Licenses upfront
Centralized management helps SaaS Vendors
Lower TCO for users

True SaaS Applicability

- **Enterprise Software Application**
  - Perform business functions
  - Organize internal and external information
  - Share data among internal and external users
  - The most standard type of software applicable to SaaS model
  - Example: Saleforce.com CRM application, Siebel On-demand application

Why Capacity Management in Cloud?

Capacity Management is

- Strategic and Proactive
- Assists in managing service quality
- Assists in managing cost expenditures
- Aligns business and IT
- Match needs and cost during growth

Following are **five characteristics of Cloud**

•They provide on-demand provisioning of computational resources;

- They use virtualization technologies to lease these resources;
- They provide public and simple remote interfaces to manage those resources;
- They use a pay-as-you-go cost model, typically charging by the hour;
- They operate data centers large enough to provide a seemingly unlimited number of resources to their clients

What are the Challenges?

Capacity on Demand- Too Little, Too Much or Just Right?

Automation and Control

Security Protecting your loved ones"

Scalability- In / Up or Out?

Old Habits – The Potential Risks

Importance of Capacity Management

When **Capacity** is **Too Little** (under capacity)

- Application Sizing not performed
- Controlled by limits
- Lower usage costs
- Service Levels affected – by how much?
- High impact on business?
  - Loss of service
  - SLA breach penalties

When **Capacity** is **Too Much** (over capacity)

- Unlimited access to resources
- Service Levels unaffected
- Costs – higher resource usage, software licensing
- Impacts on other services
  - Poor performance
  - Wasted resources (multi virtual CPU (*vSMP*) & single-threaded applications)
  - VM Sprawl
  - Increased pressure to manage virtual resources

Find the Right Balance

We need to have the **Right Capacity**?

- Application Sizing performed
- Efficient use of IaaS
- Acceptable Service Levels
  - continuous review and adjustment
- Controlled by shares, limits and reservations
- Continuous monitoring and tuning
  - Configuration adjustments (CPU, Memory)
  - VM consolidation
  - Power off unused ESX hosts
- Right Balance
  - Find the equilibrium (service / cost)

How To ? – Find the Right Balance

Automation and Control

- Rapid Elasticity- Guest Migration and Portability, Templates, Golden Host
- Resource Pools (limits, shares and reservations)
- Load Balancing
  - DRS (Automatic, Partial or Manual)
- Affinity
  - VM to VM or VM to Host
  - CPU
    - Fine grained tweaking for performance gains – Single Threaded Apps
    - Licensing

Protecting your loved ones"

- Critical business applications
- Service Levels must be met
- Highest Priority (shares, unlimited), CPU affinity
- Must guarantee resources (reservations)
- High Availability Clusters
  - VMware - Fault Tolerance
- Trade offs
  - "just in case" capacity management could impact on other services
  - Significant impact? Think about scaling out or up .

Old Habits – Potential Risks

- Gartner – *"Through 2015, more than 70% of private cloud implementations will fail to deliver operational, energy and environmental efficiencies."*
- Infrastructure or application hugging
  - Silo mentality "what's mine is mine"
- Lack of resource sharing
  - Through lack of trust and confidence
- "Just in case" capacity planning
  - Leads to over provisioning

What is Virtual Infrastructure Management

Virtual Infrastructure (VI) management—the management of virtual machines distributed across a pool of physical resources—becomes a key concern when building an IaaS cloud and **poses a number of challenges**.

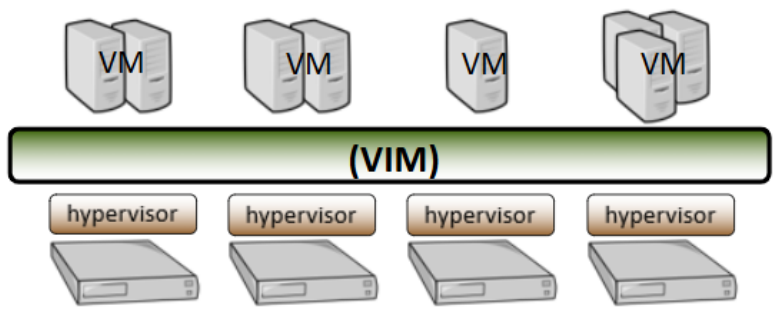
- In traditional physical resources, virtual machines require a fair amount of configuration, including preparation of the machine's software environment and network configuration.
- In a virtual infrastructure, this configuration must be done on-the-fly, with as little time between the time the VMs are requested and the time they are available to the users.

- This is further complicated by the need to configure groups of VMs that will provide a specific service (e.g., an application requiring a Web server and a database server).
- Additionally, a virtual infrastructure manager must be capable of allocating resources efficiently, taking into account an organization's goals (such as minimizing power consumption and other operational costs) and reacting to changes in the physical infrastructure.

- Virtual infrastructure management in **private clouds** has to deal with an additional problem:
- **Unlike** large IaaS cloud providers such as Amazon, private clouds typically do not have enough resources to provide the illusion of "infinite capacity."
- The immediate provisioning scheme used in public clouds, where resources are provisioned at the moment they are requested, is **ineffective in private clouds**.

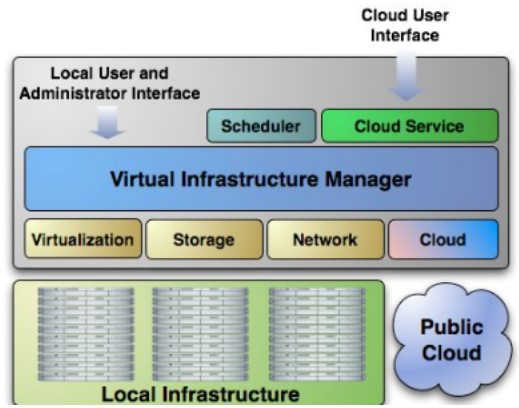
What is a Virtual Infrastructure Manager?

- A VIM runs on top of a hypervisor in a virtualized environment. The hypervisor allocates and manages virtual machines. The VIM deals with the allocation of resources in the virtual infrastructure. They include computational resources (processors), storage, and network resources. Virtual infrastructure management allows the allocation to happen based on current requirements, rather than being statically allocated.
- The VIM carries out several tasks:
- Allocating resources in accordance with traffic engineering rules.
- Support for defining operational rules.
- Definition of hub-to-facility mapping.
- Providing information for provisioning virtual infrastructure orchestration (VIO).
- VMs are great!!...but something more is needed
  - Where did/do I put my VM? (***scheduling & monitoring***)
  - How do I provision a new cluster node? (***clone***)
  - What IP addresses are available? (***networking***)
- Provide a ***uniform view*** of the resource pool
- ***Life-cycle management*** and monitoring of VM
- The VIM should ***integrate*** Image, Network and Virtualization



Why a Virtual Infrastructure Manager?

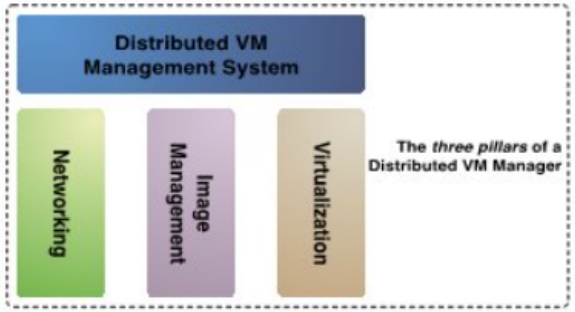
- Dynamic deployment and re-placement of virtual machines on a pool of physical resources
- Transform a rigid distributed physical infrastructure into a flexible and agile virtual infrastructure



- Backend of Public Cloud: Internal management of the infrastructure
- Private Cloud: Virtualization of cluster or data-center for internal users
- Cloud Interoperation: On-demand access to public clouds

Enter – Distributed Management of Virtual Resources

- The cloud ecosystem is inherently distributed. Resources are spread around.
- Location also plays an important role in the efficient selection / optimal selection & placement of resources.



- The problem of efficiently selecting or scheduling computational resources is well known. However, the state of the art in **VM-based resource scheduling** follows a **static approach**, where resources are initially selected using a greedy allocation strategy, with minimal or no support for other placement policies.
- To efficiently schedule resources, **VI managers** must be able to support **flexible and complex scheduling policies** and must *leverage* (use) the ability of VMs to suspend, resume, and migrate.

So What is the Solution

- Managing VMs in a pool of distributed physical resources is a key concern in IaaS clouds, requiring the use of a virtual infrastructure manager.
- **OpenNebula** is capable of managing groups of interconnected VMs—with support for the Xen, KVM, and VMWare platforms—within data centers and private clouds that involve a large amount of virtual and physical servers.
- **OpenNebula** can also be used to build hybrid clouds by interfacing with remote cloud sites.

What is OpenNebula

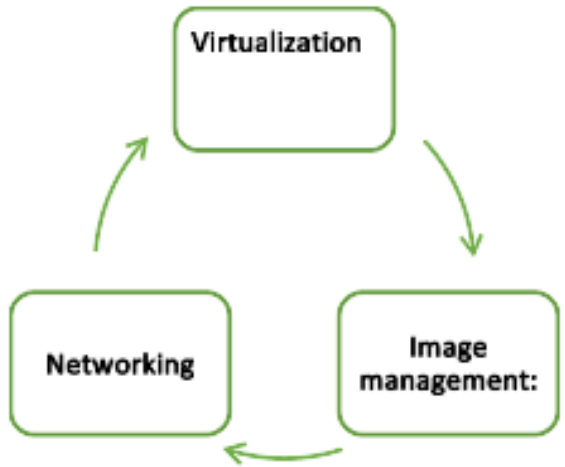
- *OpenNebula is a simple, feature-rich and flexible solution for the management of virtualized data centers.*
- *It enables private, public and hybrid clouds. Here are a few facts about this solution.*
- OpenNebula is an open source cloud middleware solution that manages heterogeneous distributed data center infrastructures.
- It is designed to be a simple but feature-rich, production-ready, customizable solution to build and manage enterprise clouds—simple to install, update and operate by the administrators; and simple to use by end users.
- OpenNebula combines existing virtualization technologies with advanced features for multi-tenancy, automated provisioning and elasticity.
- A built-in virtual network manager maps virtual networks to physical networks.
- Distributions such as Ubuntu and Red Hat Enterprise Linux have already integrated OpenNebula.
- OpenNebula supports Xen, KVM and VMware hypervisors.

- **Private Cloud** to simplify and optimize internal operations
- Hybrid Cloud to supplement the capacity of the Private Cloud
- Public Cloud to expose your Private to external users

Stages of VM Life Cycle in OpenNebula

- **The life cycle of a VM within OpenNebula follows several stages:**
  - **Resource Selection**: allowing site administrators to configure the scheduler to prioritize the resources that are more suitable for the VM
  - **Resource Preparation:** The disk images of the VM are transferred to the target physical resource. During the boot process, the VM is contextualized, a process where the disk images are specialized to work in a given environment.
  - **VM Creation**: The VM is booted by the resource hypervisor
  - **VM Migration**: The VM potentially gets migrated to a more suitable resource(e.g.,to optimizethe power consumption ofthe physical resources)
  - **VM Termination**: When the VM is going to shut down, OpenNebula can transfer back its disk images to a known location. This way, changes in the VM can be kept for a future use.

- **Within OpenNebula, a VM is modeled as having the following attributes:**
  - A capacity in terms of memory and CPU
  - A set of NICs attached to one or more virtual networks
  - A set of disk images
  - A state file (optional) or recovery file

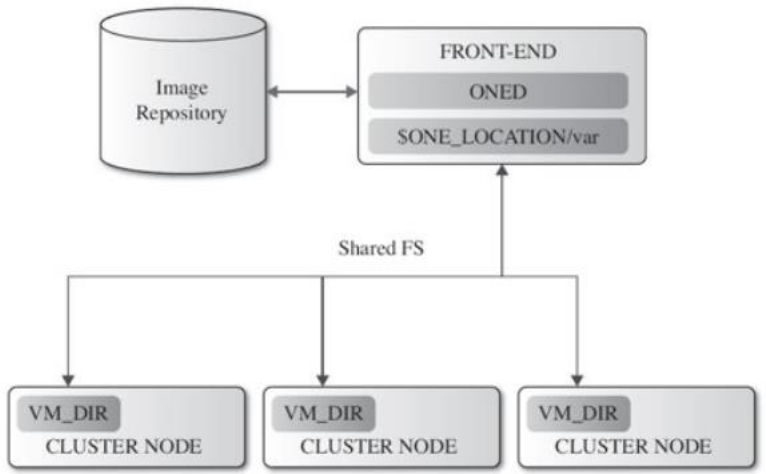


VM Management in OpenNebula

- OpenNebula manages VMs by interfacing with a physical resource's hypervisor, such as Xen, KVM, or VMWare, **Hyper-V** to control (e.g., boot, stop, or shutdown) the VM;
- using a set of **pluggable drivers** that decouple the managing process from the underlying technology.
- Thus, whenever the core needs to manage a VM, it uses **high-level commands** such as "start VM," "stop VM," and so on, which are translated by the drivers into commands that the virtual machine manager can understand. By decoupling the OpenNebula core from the virtualization technologies through the use of a **driver-based architecture**, adding support for additional virtual machine managers only requires writing a driver for it.

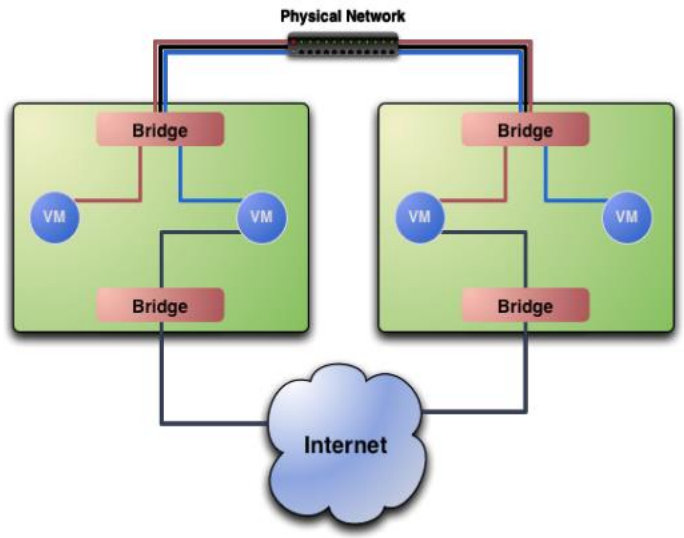
Image Management in OpenNebula

- Transferring the VM images from an image repository to the selected resource and by creating on-the-fly temporary images
- What is image?
  - Virtual disk contains the OS and other additional software
- Image management model



Networking OpenNebula

- In general, services deployed on a cloud require several interrelated VMs, with a virtual application network (VAN) being the primary link between them.
- OpenNebula dynamically creates these VANs and tracks the MAC addresses leased in the network to the service VMs.
- **physical cluster** as a set of hosts with one or more network interfaces, each of them connected to a different physical network.



Benefits of OpenNebula

For the Infrastructure Manager

- Centralized management of VM workload and distributed infrastructures
- Support for VM placement policies: balance of workload, server consolidation…
- Dynamic resizing of the infrastructure
- Dynamic partition and isolation of clusters
- Dynamic scaling of private infrastructure to meet fluctuating demands
- Lower infrastructure expenses combining local and remote Cloud resources

For the Infrastructure User

- Faster delivery and scalability of services
- Support for heterogeneous execution environments
- Full control of the lifecycle of virtualized services management

Features of OpenNebula

Feature	Function
Internal Interface	• Unix-like CLI for fully management of VM life-cycle and physical boxes • XML-RPC API and libvirt virtualization API
Scheduler	• Requirement/rank matchmaker allowing the definition of workload and resource-aware allocation policies • Support for advance reservation of capacity through Haizea
Virtualization Management	• Xen, KVM, and VMware • Generic libvirt connector (VirtualBox planned for 1.4.2)
Image Management	• General mechanisms to transfer and clone VM images
Network Management	• Definition of isolated virtual networks to interconnect VMs
Service Management and Contextualization	• Support for multi-tier services consisting of groups of inter-connected VMs, and their auto-configuration at boot time
Security	• Management of users by the infrastructure administrator
Fault Tolerance	• Persistent database backend to store host and VM information
Scalability	• Tested in the management of medium scale infrastructures with hundreds of servers and VMs (no scalability issues has been reported)
Flexibility and Extensibility	• Open, flexible and extensible architecture, interfaces and components, allowing its integration with any product or tool

What is a Cloud Workload

- A cloud workload is a specific application, service, capability or a specific amount of work that can be run on a cloud resource. Virtual machines, databases, containers, Hadoop nodes and applications are all considered cloud workloads.
- A workload is **a collection of resources and code that delivers business value, such as a customer-facing application or a backend process**. A workload might consist of a subset of resources in a single cloud account or be a collection of multiple resources spanning multiple cloud accounts.
- Examples of workloads are **marketing websites, e-commerce websites, the back-ends for a mobile app, analytic platforms**, etc. Workloads vary in levels of architectural complexity, from static websites to architectures with multiple data stores and many components.

Where do we Run Workloads?

Workload deployment -- determining where and how the workload runs -- is an essential part of workload management. Today, an enterprise can choose to deploy a workload on premises, as well as to a cloud.

Traditionally, workloads are deployed in the enterprise data center, which contains all of the server, storage, network, services and other infrastructure required to operate the workload. The business owns the data center facility and computing resources and fully controls the provisioning, optimization, and maintenance of those resources. The enterprise establishes policies and practices for the data center and workload deployment in order to meet prevailing business goals and regulatory obligations.

With the rise of the internet, cloud computing is now a viable alternative for many on-premises workload deployments.

The challenge for any business is deciding just where to deploy a given workload. Today, most general-purpose workloads can operate successfully in the public cloud, and, increasingly, applications are designed and developed to run natively and solely in a public cloud.

Workload Challenges

Technologically, the most demanding workloads may struggle in the public cloud. Some workloads require high-performance network storage or depend on internet throughput.

For example, database clusters that need high throughput and low latency may be unsuited to the cloud -- and the cloud provider may offer high-performance database services as an alternative. Applications that rely on low latency or are not designed for distributed computing infrastructures are usually kept on premises.

Technical issues aside, a business may decide to keep workloads on premises for business continuance or regulatory reasons.

Cloud clients have little actual insight into the underlying hardware and other infrastructure that hosts the workloads and data. That can be problematic for businesses obligated to meet data security and other regulatory requirements such as clear auditing and proof of data residency. Keeping those sensitive workloads in the local data center allows the business to control its own infrastructure and implement the necessary auditing and controls.

Workload Challenges

Cloud providers are also independent businesses that serve their own business interests and may not be able to meet an enterprise's specific uptime or resilience expectations for a workload. Outages happen and may last for hours -- even days -- adversely affecting client businesses and their customer base. Consequently, organizations often opt to keep critical workloads in the local data center where dedicated IT staff can maintain them.

Some organizations implement a hybrid cloud strategy that mixes on-premises, private cloud and public cloud services. This provides flexibility to run workloads and manage data where it makes the most sense, for reasons ranging from costs to security to governance and compliance. This presents tradeoffs -- for example, an organization may keep sensitive data and workloads in its own data center to preserve more direct control over them, but it also takes on more security responsibilities for them.

Amdahl's Law

In computer programming, Amdahl's law is that, in a program with parallel processing , a relatively few instructions that have to be performed in sequence will have a limiting factor on program speedup such that adding more processors may not make the program run faster.
This is generally an argument against parallel processing for certain applications and, in general, against overstated claims for parallel computing. Others argue that the kinds of applications for which parallel processing is best suited tend to be larger problems in which scaling up the number of processors does indeed bring a corresponding improvement in throughput and performance.

Amdahl's Law Formula

$$S_{max} = \frac{1}{(1-p)+\frac{p}{s}}$$

Cloud Workload – An Anatomy

Key Terms to Understand

Lease: A lease is defined as a contract between the Cloud Service Provider and the end user to facilitate the usage of resources available with the CSP to execute a workload of the end user.

Application: One or more business process encapsulated in code which requires computing resources to execute and provide business value.

Job: A Work Breakdown Structure of an application. An application may contain several jobs, and all of them need to be executed to complete execution of the application.

Tasks: Steps that need to be performed to complete a job. Task can be sequential or parallel.

When we talk about cloud workload we are referring to the completion of a specific job which provides some business values.

Resource Allocation vs Task Scheduling

- ***It might look similar, but there are lots of differences***
- ***RA➔ Identifying and allocating a resource of a type of work load***
- ***TS➔ Ability to shuffle and manage tasks to operate efficiently on the available resources.***

Scheduling Techniques

- While a VI manager like OpenNebula can handle all the minutiae of managing VMs in a pool of physical resources, **scheduling these VMs efficiently is a different and complex matter.**
- **Immediate provisioning model** is used by commercial cloud providers, such as Amazon, since their data centers' capacity is assumed to be infinite.
- Best-effort provisioning where requests have to be queued and prioritized
- **Advance provisioning** where resources are pre-reserved so they will be guaranteed to be available at a given time period.
- However, when managing a private cloud with limited resources, an immediate provisioning model is insufficient.
- **A lease-based** resource **provisioning model** that can act as a scheduling back-end for OpenNebula, supporting other **provisioning models** other than the immediate provisioning models in existing cloud providers. In particular, Haizea adds support for both best-effort provisioning and advance reservations, when managing a finite number of resources

Scheduling Techniques – Existing Approaches

- Additionally, advance reservations lead to utilization problems caused by the need to vacate resources before a reservation can begin.
- Traditional job schedulers are unable to efficiently schedule workloads combining both best-effort jobs and advance reservations.
- However, advance reservations can be supported more efficiently by using a scheduler capable of preempting running jobs at the start of the reservation and resuming them at the end of the reservation.
- Preemption can also be used to run large parallel jobs (which tend to have long queue times) earlier, and it is specially relevant in the context of urgent computing, where resources have to be provisioned on very short notice and the likelihood of having jobs already assigned to resources is higher.
- While preemption can be accomplished by canceling a running job, the least disruptive form of preemption is check pointing, where the preempted job's entire state is saved to disk, allowing it to resume its work from the last checkpoint.

Scheduling Techniques – Existing Approaches

- Additionally, advance reservations lead to utilization problems caused by the need to vacate resources before a reservation can begin.
- Traditional job schedulers are unable to efficiently schedule workloads combining both best-effort jobs and advance reservations.
- However, advance reservations can be supported more efficiently by using a scheduler capable of preempting running jobs at the start of the reservation and resuming them at the end of the reservation.
- Preemption can also be used to run large parallel jobs (which tend to have long queue times) earlier, and it is specially relevant in the context of urgent computing, where resources have to be provisioned on very short notice and the likelihood of having jobs already assigned to resources is higher.
- While preemption can be accomplished by canceling a running job, the least disruptive form of preemption is check pointing, where the preempted job's entire state is saved to disk, allowing it to resume its work from the last checkpoint.

- Additionally, some schedulers also support job migration, allowing check-pointed jobs to restart on other available resources, instead of having to wait until the preempting job or reservation has completed.
- •Check-pointing-based preemption, requires the job's executable itself to be checkpointable. An application can be made checkpointable by explicitly adding that functionality to an application (application-level and library-level checkpointing) OR transparently by using OS-level checkpointing, where the operating system (such as Cray, IRIX, and patched versions of Linux using BLCR [17]) checkpoints a process, without rewriting the program or relinking it with checkpointing libraries.
- •Thus, a job scheduler capable of checkpointing-based preemption and migration could be used to checkpoint jobs before the start of an advance reservation, minimizing their impact on the schedule.
- •However, the application and library-level checkpointing approaches burden the user with having to modify their applications to make them checkpointable, imposing a restriction on the software environment. On the other hand, OS-level checkpointing is a more appealing option, but still imposes certain software restrictions on resource consumers.

- An alternative approach to supporting advance reservations was proposed by Nurmi et al. [18], which introduced "virtual advance reservations for queues" (VARQ).
- •This approach overlays advance reservations over traditional job schedulers by first predicting the time a job would spend waiting in a scheduler's queue and then submitting a job (representing the advance reservation) at a time such that, based on the wait time prediction, the probability that it will be running at the start of the reservation is maximized.
- •Since no actual reservations can be done, VARQ jobs can run on traditional job schedulers, which will not distinguish between the regular best-effort jobs and the VARQ jobs.
- •Although this is an interesting approach that can be realistically implemented in practice (since it does not require modifications to existing scheduler), it still depends on the job abstraction.

Scheduling Techniques – VM Overheads

Virtualization technologies are a key enabler of many features found in IaaS clouds. Virtual machines are also an appealing vehicle for implementing efficient reservation of resources due to:

- Ability to be suspended,
- Potentially migrated,
- Resumed without modifying any of the applications running inside the VM.

However, virtual machines also raise additional challenges related to the overhead of using VMs:

- **Preparation Overhead.** When using VMs to implement reservations, a VM disk image must be either prepared on-the-fly or transferred to the physical node where it is needed. Since a VM disk image can have a size in the order of gigabytes, this preparation overhead can significantly delay the starting time of leases. This delay may, in some cases, be unacceptable for advance reservations that must start at a specific time.
- **Runtime Overhead**. Once a VM is running, scheduling primitives such as **checkpointing** and **resuming** can incur in significant overhead since a VM's entire memory space must be saved to disk, and then read from disk. Migration involves transferring this saved memory along with the VM disk image. Similar to deployment overhead, this overhead can result in noticeable delays.

Reservation Based Provisioning

- A particularly interesting problem when provisioning virtual infrastructures is **how to deal with situations where the demand for resources is known beforehand**—for example, when an experiment depending on some complex piece of equipment is going to run from 2 pm to 4 pm, and computational resources must be available at exactly that time to process the data produced by the equipment.
- Commercial clouds do have **infinite resources** to handle this situation. On the other hand, when dealing with **finite capacity**, a different approach is needed. However, the intuitively simple solution of reserving the resources beforehand is not so simple, because it is known to cause resources to be underutilized, due to the difficulty of scheduling other requests around an inflexible reservation.

Provisioning to meet SLA

- IaaS clouds can be used to deploy services that will be consumed by users other than the one that has deployed the services.
- There is a distinction between the **cloud consumer** (i.e., the **service owner**; for instance, the company that develops and manages the applications) and the **end users** of the resources provisioned on the cloud (i.e., the **service user**; for instance, the users that access the applications).
- Furthermore, **service owners** will enter into **service-level agreements (SLAs)** with their end users, covering guarantees such as the timeliness with which these services will respond.
- Requirements are formalized in infrastructure SLAs between **the service owner** and **cloud provider**, separate from the high-level **SLAs between the service owner** and **its end users**.
- In many cases, either the service owner is not resourceful enough to perform an exact service sizing or service workloads are hard to anticipate in advance.
- Therefore, to protect high-level SLAs, the cloud provider should cater for **elasticity on demand**.
- **Scaling and de-scaling** of an application is best managed by the application itself. The reason is that in many cases, resource allocation decisions are **application-specific** and are being driven by the **application level metrics**.

What is Haizea Scheduler

When managing a private cloud with limited resources, **an immediate provisioning model is insufficient**. A **lease-based resource provisioning model** that can act as a scheduling back-end for OpenNebula, **supporting other provisioning models other than the immediate provisioning models in existing cloud providers**. In particular, Haizea adds support for both best-effort provisioning and advance reservations, **when managing a finite number of resources**. The remainder of this section describes Haizea's leasing model and the algorithms Haizea uses to schedule these leases.

- We define a **lease** as "**a negotiated and renegotiable agreement between a resource provider and a resource consumer, where the former agrees to make a set of resources available to the latter, based on a set of lease terms presented by the resource consumer**."
- The terms must encompass the following:
  - the hardware resources required by the resource consumer, such as CPUs, memory, and network bandwidth;
  - a software environment required on the leased resources;
  - and an availability period during which a user requests that the hardware and software resources be available.

Solution – Haizea Scheduler

The Haizea project (http://haizea.cs.uchicago.edu/) was created to develop a scheduler that can efficiently support advance reservations efficiently by using the suspend/resume/migrate capability of VMs, but minimizing the overhead of using VMs.

The fundamental resource provisioning abstraction in Haizea is the lease, with three types of lease currently supported:

Advanced reservation leases, where the resources must be available at a specific time.

Best-effort leases, where resources are provisioned as soon as possible and requests are placed on a queue if necessary.

Immediate leases, where resources are provisioned when requested or not at all.

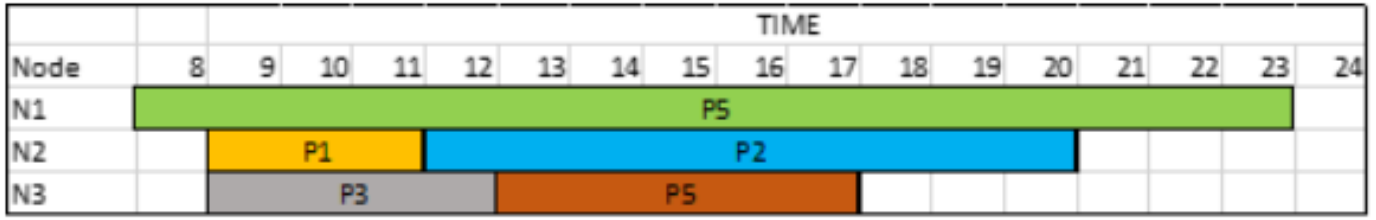
Leasing Schedule – Best Effort Lease (BEL)

- **Best-effort leases** are scheduled using a **queue**. When a best-effort lease is requested, the lease request is placed at the end of the queue, which is periodically evaluated using a backfilling algorithm to determine if any leases can be scheduled.

- The scheduler does this by **first checking the earliest possible starting time** for the lease on each physical node, which will depend on the required disk images. **For example**, if some physical nodes have cached the required disk image, it will be possible to start the lease earlier on those nodes.

- Once these earliest starting times have been determined, the scheduler chooses the nodes that allow the lease to start the soonest.

- The use of **VM suspension/resumption** allows the best-effort leases to be scheduled **even if there are not enough resources available for their full requested duration**.



e.g. If we have three nodes N1, N2 and N3 and five processes have to be scheduled with entry time and duration of resources as follows – Best effort queue

Process	Entry time	Time required	Node	Start time		
P1	9am	3 hours	N2	9am	Ended at 12pm	
P2	10am	10 hours	N2	12pm	Ended at 8pm	
P3	9.30am	4 hours	N3	9.30am	Ended at 1.30pm	
P4	11am	5hours	N3	1.30pm	Ended at 6.30pm	
P5	8am	15hours	N1	8am	Ended at 11pm	

Leasing Schedule – Advanced Reservation (AR)

- **Advance reservations**, on the other hand, **do not go through a queue, since they must start at either the requested time or not at all**.
- Thus, scheduling this type of lease is relatively simple, because it mostly involves checking if there are enough resources available during the requested interval.
- However, the scheduler must also check **if any associated overheads can be scheduled** in such a way that the lease can still start **on time**.
- **For preparation overhead**, the scheduler determines if the required images can be transferred on time.
- These transfers are scheduled using an **Earliest Deadline First (EDF) algorithm**, where **the deadline for the image transfer is the start time of the advance reservation lease**.
- **For runtime overhead**, the scheduler will attempt to schedule the lease without having to preempt other leases; if preemption is unavoidable. The necessary suspension operations are scheduled; if they can be performed on time.

e.g. If we have three nodes N1, N2 and N3 and five processes have to be scheduled with entry time and duration as given below (last example) with P2 being advanced reservation. (HW next slide)

Process	Entry time	Time required	Node	Start time		
P1	9am	3 hours	N2	9am	Ended at 12pm	
P2	10am	10 hours	N1	10am	Ended at 8pm	
P3	9.30am	4 hours	N3	9.30am	Ended at 1.30pm	
P4	11am	5hours	N3	1.30pm	Ended at 6.30pm	
P5	8am	15hours	N1	8am pre-empted at 10am	Resumed on N2 at 12pm	Ended at 1 am

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS

- If temporal behavior of services with respect to resource demands is highly predictable, then capacity can be efficiently scheduled using reservations.

- In this section we focus on **less predictable elastic workloads**. For these workloads, **exact scheduling of capacity may not be possible**. Rather than that, **capacity planning and optimizations are required**.

- IaaS providers perform two complementary management tasks:

  (1) **Capacity planning** to make sure that SLA obligations are met as contracted with the service providers and;

  (2) **Continuous optimization** of resource utilization in specific workload to make the most efficient use of the existing capacity.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
- Infrastructure SLA's

- IaaS can be regarded as a giant virtual hardware store, where computational resources such as virtual machines (VM), virtual application networks (VAN) and virtual disks (VD) can be **ordered on demand in the matter of minutes or even seconds**.
- Chandra et al. [29] quantitatively study advantages of **fine-grain resource allocation** in a shared hosting platform. As this research suggests, **fine-grain temporal and spatial resource allocation** may lead to substantial improvements in capacity utilization.
- Amazon EC2 [1] offers small, large, and extra large general-purpose VM instances and **high-CPU**, **medium** and **extra large** instances. It is possible that more instance types (**e.g., I/O high, memory high, storage high, etc.**) will be added in the future should a demand for them arise. Other IaaS providers—for example, GoGrid [3] and FlexiScale [4]—follow similar strategy.

- Thus, to deploy a service on a cloud, **a service provider orders suitable virtual hardware** and installs its application software on it.
- From the IaaS provider, a given service configuration is a virtual resource array of black box resources, which correspond to the number of instances of resource type.
- For example, a typical three-tier application may contain **ten general-purpose small instances** to run Web front-ends, **three large instances** to run an application server cluster with load balancing and redundancy, and **two large instances** to run a replicated database.
- A risk mitigation mechanism to protect user experience in the IaaS model is offered by infrastructure SLAs (i.e., the SLAs formalizing capacity availability) signed between service provider and IaaS provider.

- There is **no universal approach to infrastructure SLAs**. As the IaaS field matures and more experience is being gained, some methodologies may become more popular than others. Also some methods may be more suitable for specific workloads than other. There are three main approaches as follows.
- **No SLAs.** This approach is based on two premises: **(a)** Cloud always has spare capacity to provide on demand, and **(b)** services are not QoS sensitive and can withstand moderate performance degradation. This methodology is best suited for the best effort workloads.
- **Probabilistic SLAs.** These SLAs allow **us to trade capacity availability for cost of consumption**. Probabilistic SLAs specify clauses that determine availability percentile for contracted resources computed over the SLA evaluation period. **The lower the availability percentile, the cheaper the cost of resource consumption.** This type of SLA is **suitable for small and medium businesses** and **for many enterprise grade applications**.
- **Deterministic SLAs.** These are, in fact, probabilistic SLAs where **resource availability percentile is 100%**. These SLAs are most stringent and difficult to guarantee. From the provider's point of view, they do not admit capacity multiplexing. Therefore this is the most costly option for service providers, which may be applied for critical services.

- We will focus on probabilistic SLAs, however, because they represent the more interesting and flexible option and lay the foundation for the rest of discussion on **statistical multiplexing of capacity**.
- Before we can proceed, we need to define the concept, **elasticity rules**, which are are scaling and de-scaling policies that guide transition of the service from one configuration to another to match changes in the environment. The main motivation for defining these policies stems from the pay-as-you-go billing model of IaaS clouds. The service owner is interested in paying only for what is really required to satisfy workload demands minimizing the over-provisioning overhead. There are **three types of elasticity rules**:
  - **Time-driven:** These rules **change the virtual resources array in response to a timer event**. These rules are useful for predictable workloads—for example, for services with well-known business cycles.
  - **OS Level Metrics-Driven:** These rules react on predicates defined in terms of the OS parameters (see Amazon Auto-scaling Service). These **auto-scaling policies are useful for transparently scaling and de-scaling services**. The problem is, however, that in many cases **this mechanism is not precise enough**.
  - **Application Metrics-Driven:** This is a unique RESERVOIR offering that **allows an application to supply application-specific policies** that will be transparently executed by IaaS middleware in reacting on the monitoring information supplied by the service-specific monitoring probes running inside VMs.

CLOUD COMPUTING

Unit 7

Agenda

- Capacity Management Recap
- Multi Tenancy in the Cloud
- 4 Models of Multi Tenancy

Cloud security issues

What is Multi Tenancy?

Multi-tenancy is an architectural pattern

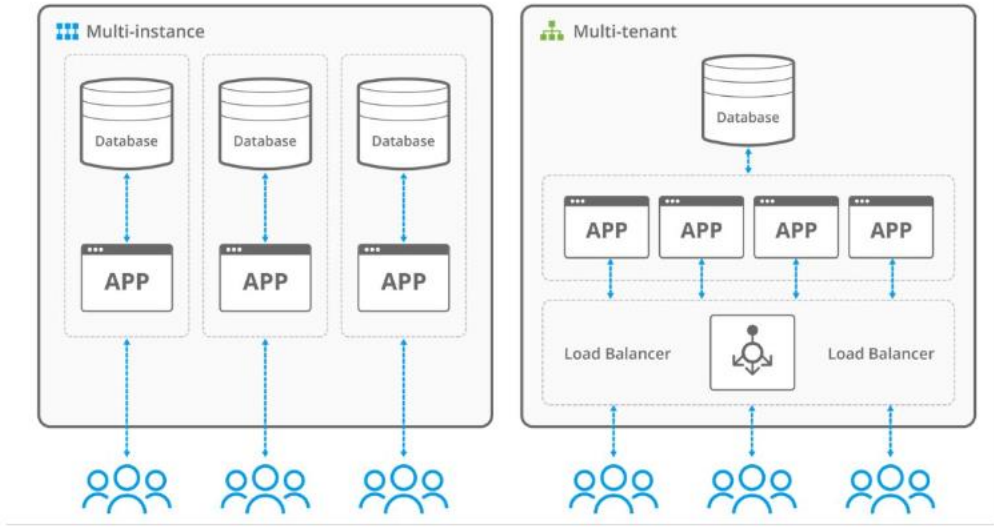
A single instance of the software is run on the service provider's infrastructure

Multiple tenants access the same instance.

In contrast to the multi-user model, multi-tenancy requires customizing the single instance according to the multi-faceted requirements of many tenants.

- Multi-tenancy is an architecture in which a **single instance** of a **software application** serves **multiple customers**. Each customer is called a tenant. Tenants may be given the ability to customize some parts of the application, such as color of the user interface (UI) or business rules, **but they cannot customize the application's code**.

Multi Tenant vs Multi Instance



- In the multi tenant architecture, the application is redesigned to handle the resource sharing between the multiple tenants.
- For example, SalesForce.com (service provider) hosts the CRM application using their infrastructure.
- A company who wants to use this hosted CRM application for their business is the customer and the employees of the companies to whom the company provides privileges to access the CRM application are the actual users of the application

IAAS – MT Model

In (IaaS), a single physical or virtual infrastructure is shared among multiple tenants. Each tenant is provided with its own logical environment, which includes compute resources such as virtual machines (VMs), storage, networking, and other infrastructure components. There are several ways to implement a multi-tenant deployment model for IaaS:

1.Virtual machine isolation: In this model, each tenant is provided with a set of virtual machines that are completely isolated from other tenants. This approach provides strong security and isolation, but can be less efficient than other approaches.

2.Network isolation: In this model, each tenant is provided with its own virtual network that is isolated from other tenants. This approach provides better performance and efficiency than virtual machine isolation, but may require more complex network management.

3.Resource pool isolation: In this model, each tenant is provided with a dedicated set of compute resources such as CPU, memory, and storage that are isolated from other tenants. This approach provides good performance and resource allocation, but may be less secure than other approaches.

4.Hybrid model: In this model, a combination of the above models is used to meet the specific requirements of each tenant. For example, some tenants may require strong security and isolation, while others may prioritize performance and efficiency.

PAAS – MT Model

1.Application-level isolation: In this model, each tenant's application is isolated from other tenants' applications at the application level. This can be achieved through containerization or virtualization, and may also involve isolating the application's data and configuration settings.

2.Tenant-level isolation: In this model, each tenant is provided with a dedicated instance of the platform, which is isolated from other tenants. This approach provides strong security and isolation, but can be less efficient than other approaches.

3.Resource pool isolation: In this model, each tenant is provided with a dedicated set of compute resources such as CPU, memory, and storage that are isolated from other tenants. This approach provides good performance and resource allocation, but may be less secure than other approaches.

4.Hybrid model: In this model, a combination of the above models is used to meet the specific requirements of each tenant. For example, some tenants may require strong security and isolation, while others may prioritize performance and efficiency.

SAAS – MT Model

There are several ways to implement a multi-tenant deployment model for SaaS:

1.Database-level isolation: In this model, each tenant's data is stored in a separate database schema, which provides complete isolation and security. This approach can be efficient and scalable, but may require complex database management.

2.Application-level isolation: In this model, each tenant's data is kept separate at the application level, which may involve partitioning data and configuration settings. This approach can be more flexible and easier to manage than the database-level isolation model, but may be less secure.

3.Hybrid model: In this model, a combination of the above models is used to meet the specific requirements of each tenant. For example, some tenants may require complete data isolation, while others may be willing to share some components of the application.

Benefits

Cost Savings –

- An application instance usually incurs a certain amount of memory and processing overhead which can be substantial when multiplied by many customers, especially if the customers are small.
- As the single instance is shared between multiple tenants this cost overhead can be segregated between multiple tenants.

Data aggregation–

- In non MT architecture, the data for different customers will be located in different database schemas and pulling information from all of them can be a very cumbersome task.
- In MT architecture, instead of collecting data from multiple data sources, with potentially different database schemas, all data for all customers is stored in a single database schema.
- Thus, running queries across customers, mining data, and looking for trends is much simpler.

Release management –

- MT simplifies the release management process.
- In a traditional release management process, packages containing code and database changes are distributed to client desktop and/or server machines.
- These packages then have to be installed on each individual machine.
- With the multitenant model, the package typically only needs to be installed on a single server. This greatly simplifies the release management process.

Characteristics of MT Architecture

Customization –

- Multi-tenant applications are typically required to provide a high degree of customization to support each target organization's needs. Customization typically includes the following aspects:
  - ❖ **Branding**: allowing each organization to customize the look-and-feel of the application to match their corporate branding (often referred to as a distinct "skin").
  - ❖ **Workflow**: accommodating differences in workflow to be used by a wide range of potential customers.
  - ❖ **Extensions to the data model**: supporting an extensible data model to give customers the ability to customize the data elements managed by the application to meet their specific needs.
  - ❖ **Access control**: letting each client organization independently customize access rights and restrictions for each user.

Quality of service

- Multitenant applications are expected to provide adequate isolation of security, robustness and performance between multiple tenants which is provided by the layers below the application in case of multi-instance applications

MT- Different Levels

- Implementing the highest degree of resource sharing for all resources may be prohibitively expensive in development effort and complexity of the system.
- A balanced approach, where there is fine grained sharing of resources only for important resources, may be the optimum approach.
- The four levels of multi-tenancy are described in the following list for any given resource in a cloud system, the appropriate level could be selected
  - **Custom instances**
  - **Configurable instances**
  - **Configurable, multi-tenant efficient instances**
  - **Scalable, configurable, multi tenant efficient resources**

Custom instances

- Lowest level of MT
- Each customer has own custom version of application
- Different versions of application are running differently
- Extremely difficult to manage as needs dedicated support for each customer

Configurable instances

- Same version of application is shared between the customers with customizations specific to each customer
- Different instances of same application are running
- Supports customization like logos on the screen, tailor made workflows
- Managing application is better that custom instances approach as only one copy needs to be managed
- Upgrades are simple and seamless

Configurable, multi-tenant efficient instances

- Same version of application is shared between the customers through a single instance of application
- More efficient usage of the resources
- Management is extremely simple

Scalable, configurable, multi tenant efficient resources

- All features of level 3 are supported.
- Application instances are installed on cluster of computers allowing it to scale as per demand.
- Maximum level of resource sharing is achieved.
- Example, Gmail

Security in MT

- The key challenge in multi-tenancy is the secure sharing of resources. A very important technology to ensure this is authentication.
- Clearly each tenant would like to specify the users who can log in to the cloud system. Unlike traditional computer systems, the tenant would specify the valid users, but authentication still has to be done by the cloud service provider.
- Two basic approaches can be used: **a centralized authentication system** or a **decentralized authentication system** .
- In the **centralized system**, all authentication is performed using a centralized user data base. The cloud administrator gives the tenant's administrator rights to manage user accounts for that tenant. When the user signs in, they are authenticated against the centralized database.

- In the **decentralized system**, each tenant maintains their own user data base, and the tenant needs to deploy a federation service that interfaces between the tenant's authentication framework and the cloud system's authentication service.
- **Decentralized authentication** is useful if **single sign-on is important**, since centralized authentication systems will require the user to sign on to the central authentication system in addition to signing on to the tenant's authentication system.
- However, **decentralized authentication systems** have the **disadvantage that they need a trust relationship between the tenant's authentication system and the cloud provider's authentication system**. Given the self-service nature of the cloud (i.e., it is unlikely that the cloud provider would have the resources to investigate each tenant, and ensure that their authentication infrastructure is secure), centralized authentication seems to be more generally applicable.

Multi Tenancy: Resource Sharing

- Two major resources that need to be shared are storage and servers.
- The basic principles for sharing of these resources are described first.
- This is followed by a deeper discussion that focuses on the question of how these resources can be shared at a fine granularity, while allowing the tenants to customize the data to their requirements.

Sharing storage resources:

- In a multi-tenant system, many tenants share the same storage system. Cloud applications may use two kinds of storage systems: File systems and databases, where the term database is used to mean not only relational databases, but NoSQL databases as well.
- The discussion is focused on sharing data for different users in a database. The focus is also on multi-tenant efficient approaches where there is only one instance of the database which is shared among all the tenants

Resource Sharing Approaches

STORAGE Sharing Scenarios in MT

Dedicated tables per tenant

- We discuss only the approach where only one instance of database is shared between the tenants.
- Each tenant has separate copy of table
- Only tenant is given the privileges to access these tables, no other can access it
- Customizations can be easily added to the tenant's tables

Shared table approach

- Tables are shared between the tenants
- Needs to isolate between the tenants data in different rows using the unique tenant id assigned to each tenant
- More space efficient than dedicated table approach
- Needs more compute resources as it needs to use view to make join to retrieve tenant specific data
- Metadata table needs to be maintained for tenant's information
- Managing customization is difficult

Tenant 1 Employee Table

EmpId	EmpName	EmpDept	Salary
1	P	IT	10
2	Q	Sales	20
3	R	IT	30

Tenant 2 Employee Table

EmpId	EmpName	EmpDept	Salary
11	A	Manu	234
22	B	Sales	245
33	c	Retail	335

Data Table

Tenant Id	EmpId	EmpName	EmpDept	Salary
1	1	P	IT	10
1	2	Q	Sales	20
1	3	R	IT	30
2	11	A	Manu	234
2	22	B	Sales	245
2	33	C	Retail	335

Metadata table

Tenant Id	Tenant Name
1	Tenant1
2	Tenant2

Support for Customization

Customization:

- It is important for the cloud infrastructure to support customization of the stored data, since it is likely that different tenants may want to store different data in their tables.
- For example, an automobile repair shop, different shops may want to store different details about the repairs carried out.
- Three methods for doing this are described in the next slides. It is to be noted that **difficulties for customization** occur only in the **shared table method**.
- In the dedicated table method, each tenant has their own table, and therefore can have different schema.

Pre Allocated Columns:

- In shared table approach, it's very complex to provide support for the customizations.
- Each tenant might have his unique requirements to store data in the tables and using shared table approach, managing such requirements needs to come up with proper data architecture.

Pre allocated columns

- Fixed number of columns is reserved for custom columns
- If the numbers of custom column are too less than the reserved custom columns then space are wasted
- If the numbers of custom columns are more than the reserved custom columns then customer will feel restricted

Name-Value Pairs:

- The major problem with the pre-allocated columns technique is that there could be a lot of wasted space.
- If the number of columns is too low, then users will feel constrained in their customizations.
- However, if the number is too big, there will be a lot of space wastage.

Name-Value Pairs

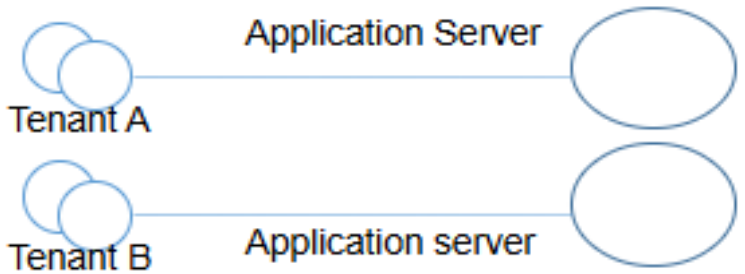
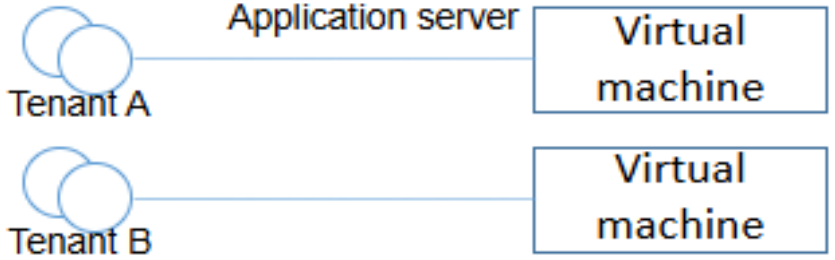
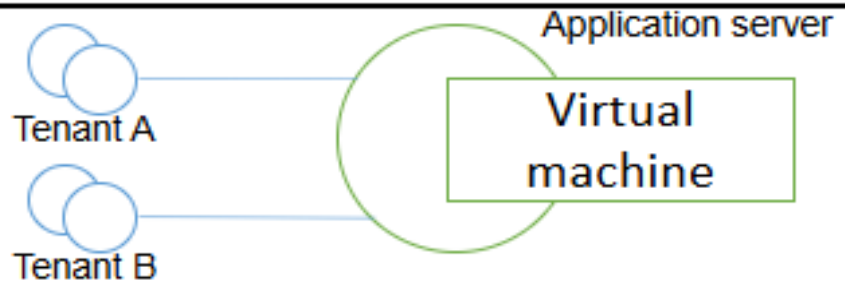
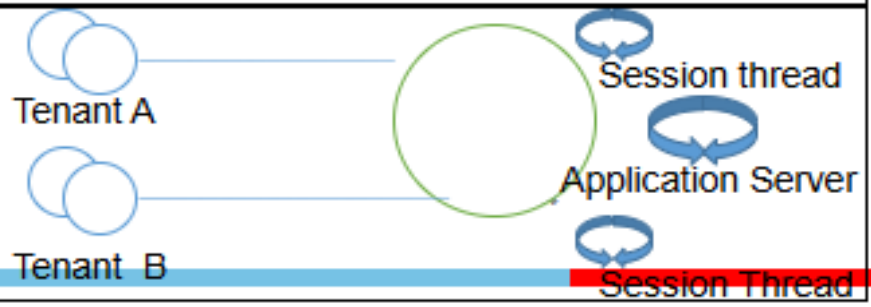
- A metadata table for tenant is maintained
- A data table is specified with standard common columns and has extra column at end to point to another name value pair table
- Name value pair table (aka pivot table) actually includes the custom fields for this record.
- The actual custom fields are stored with data type and other metadata information.
- Space efficient as compared to pre allocated columns method
- Joins are involved to fetch tenant specific data

XML method :

- The last column of database table is reserved for storing the information in XML format

- Let's consider Tenants have following table structure that needs to be mapped to the **Name-Value pair table structure**.
- Each tenant table has:
- standard common fields and
- few custom fields having varying data type

Multi Tenancy: Resource Sharing

Fully isolated Application server Each tenant accesses an application server running on a dedicated servers.	
Virtualized Application Server Each tenant accesses a dedicated application running on a separate virtual machine.	
Shared Virtual Server Each tenant accesses a dedicated application server running on a shared virtual machine.	
Shared Application Server The tenant shared the application server and access application resources through separate session or threads.	

Cloud Security Issues

Loss of Control in the Cloud

- Consumer's loss of control
  - Data, applications, resources are located with provider
  - User identity management is handled by the cloud
  - User access control rules, security policies and enforcement are managed by the cloud provider
  - Consumer relies on provider to ensure
    - Data security and privacy
    - Resource availability
    - Monitoring and repairing of services/resources

Lack of Trust in the Cloud

- Trusting a third party requires taking risks
- Defining trust and risk
  - Opposite sides of the same coin (J. Camp)
  - People only trust when it pays (Economist's view)
  - Need for trust arises only in risky situations
- Defunct third party management schemes
  - Hard to balance trust and risk
  - e.g. Key Escrow (Clipper chip)
  - Is the cloud headed toward the same path?

Multi-tenancy Issues in the Cloud

- Conflict between tenants' opposing goals
  - Tenants share a pool of resources and have opposing goals
- How does multi-tenancy deal with conflict of interest?
  - Can tenants get along together and 'play nicely' ?
  - If they can't, can we isolate them?
- How to provide separation between tenants?
- Cloud Computing brings new threats
  - Multiple independent users share the same physical infrastructure
  - Thus an attacker can legitimately be in the same physical machine as the target

Taxonomy of Fear - CIA

- Confidentiality
  - Fear of loss of control over data
    - Will the sensitive data stored on a cloud remain confidential?
    - Will cloud compromises leak confidential client data
  - Will the cloud provider itself be honest and won't peek into the data?
- Integrity
  - How do I know that the cloud provider is doing the computations correctly?
  - How do I ensure that the cloud provider really stored my data without tampering with it?
- Availability
  - Will critical systems go down at the client, if the provider is attacked in a Denial of Service attack?
  - What happens if cloud provider goes out of business?
  - Would cloud scale well-enough?
  - Often-voiced concern
    - Although cloud providers argue their downtime compares well with cloud user's own data centers

Threat Model

- A threat model helps in analyzing a security problem, design mitigation strategies, and evaluate solutions

- Steps:
  - Identify attackers, assets, threats and other components
  - Rank the threats
  - Choose mitigation strategies
  - Build solutions based on the strategies

- Basic components
  - Attacker modeling
    - Choose what attacker to consider
      - insider vs. outsider?
      - single vs. collaborator?
    - Attacker motivation and capabilities
  - Attacker goals
  - Vulnerabilities / threats

Data Security and Storage

- Several aspects of data security, including:
  - Data-in-transit
    - Confidentiality + integrity using secured protocol
    - Confidentiality with non-secured protocol and encryption
  - Data-at-rest
    - Generally, not encrypted , since data is commingled with other users' data
    - Encryption if it is not associated with applications?
      - But how about indexing and searching?
  - Processing of data, including multitenancy
    - For any application to process data

PART III. POSSIBLE SOLUTIONS

Minimize Lack of Trust: Policy Language

- Consumers have specific security needs but don't have a say-so in how they are handled
  - Currently consumers cannot dictate their requirements to the provider (SLAs are one-sided)
- Standard language to convey one's policies and expectations
  - Agreed upon and upheld by both parties
  - Standard language for representing SLAs
- Create policy language with the following characteristics:
  - Machine-understandable (or at least processable),
  - Easy to combine/merge and compare

Minimize Lack of Trust: Certification

- Certification
  - Some form of reputable, independent, comparable assessment and description of security features and assurance
    - Sarbanes-Oxley, DIACAP, DISTCAP, etc
- Risk assessment
  - Performed by certified third parties
  - Provides consumers with additional assurance

Minimize Loss of Control: Monitoring

- Cloud consumer needs situational awareness for critical applications
  - When underlying components fail, what is the effect of the failure to the mission logic
  - What recovery measures can be taken
    - by provider and consumer

- Requires an application-specific run-time monitoring and management tool for the consumer
  - The cloud consumer and cloud provider have different views of the system
  - Enable both the provider and tenants to monitor the components in the cloud that are under their control

Minimize Loss of Control: Monitoring (Cont.)

- Provide mechanisms that enable the provider to act on attacks he can handle.
  - infrastructure remapping
    - create new or move existing fault domains
  - shutting down offending components or targets
    - and assisting tenants with porting if necessary
  - Repairs
- Provide mechanisms that enable the consumer to act on attacks that he can handle
  - application-level monitoring
  - RAdAC (Risk-adaptable Access Control)
  - VM porting with remote attestation of target physical host
  - Provide ability to move the user's application to another cloud

Conclusion

- Cloud computing is sometimes viewed as a reincarnation of the classic mainframe client-server model
  - However, resources are ubiquitous, scalable, highly virtualized
  - Contains all the traditional threats, as well as new ones

- In developing solutions to cloud computing security issues it may be helpful to identify the problems and approaches in terms of
  - Loss of control
  - Lack of trust
  - Multi-tenancy problems

What is SLA or Service Level Agreement

Describes a set of non functional requirements of the service.

Example : RTO time – Return to Operation Time if case of failure

SLO – Service Level Objective. That is, the objective to be achieved.

KPI – Key Performance Indicators

Service Level Objective:

Objective of service quality that has to be achieved.

Set of measurable KPIs with thresholds to decide if the objective is fulfilled or not.

- Attainable
- Repeatable
- Measurable
- Understandable
- Meaningful
- Controllable
- Affordable
- Mutually acceptable

SLA Role in High Availability

What is High Availability

Driven by SLA (Service Level Agreement)

High Availability must conform to SLA. Goal here is to meet promised quality

Examples: Service is available 99.95%.

Net Banking : Guarantees banking 24 x7 . There are published down times called " SYSTEM MAINTENANCE" window

Duronto Express : Promises point to point service with only Service halts

What deters HA

Service outage which was unplanned because of Server or Human Error

Service Disruption by Planned Maintenance windows (Downtime)

Service Performance degradation due to lack of infrastructure or failure of critical components during peak load time

Bottom Line : Need effective Capacity Planning to meet promised QoS or in other words maintain HA thus adhering to the SLA's

Steps for HA

Steps to achieve high availability

- **Build for server failure**
    - Have redundant servers which can be made online
- **Build for zone failure**
    - Having DR sites in case of failure to switch over to backup site
- **Build for Cloud failure**
    - Plan for your Cloud setup to be robust and contained. Errors should not cascade, having fire door policy to contain threats or errors which affect the ecosystem
- **Automating and testing**
    - Test & Test again, low manual interference

SLA vs SLO

The SLA is the entire agreement that specifies

- what service is to be provided,
- how it is supported,
- times,
- locations,
- costs,
- performance, and
- responsibilities of the parties involved.

SLOs are specific measurable characteristics of the SLA such as

- availability,
- throughput,
- frequency,
- response time, or quality.

SLIs are actual measure of the SLO and serves as a benchmark to compare against the promised SLO

- availability,
- throughput,
- frequency,
- response time, or quality.

SLA

In the early days of web-application deployment**, performance of the application** at peak load was a single important criterion for provisioning server resources.

Provisioning in those days involved deciding hardware configuration**,** determining the number of physical machines**,** and acquiring them upfront so that the overall business objectives could be achieved.

The web applications were **hosted** on these dedicated individual servers within enterprises' own server rooms. These web applications were used to provide different kinds of e-services to various clients.

Due to the increasing **complexity of managing the huge Data centres,** enterprises started outsourcing the application hosting to the infrastructure providers. They would procure the hardware and make it available for application hosting.

It necessitated the enterprises to enter into a **legal agreement with the infrastructure service providers** to guarantee a minimum quality of service (QoS).

Typically**,** the **QoS parameters** are related to the availability of the system CPU**,** data storage**,** and network for efficient execution of the application at peak loads.

This legal agreement is known as the service-level agreement (SLA)

Traditional SLO Management Approaches

Load balancing – is to distribute the incoming request onto a set of physical machines, each hosting a replica of an application, so that the load on the machines is equally distributed. Front end node (node facing the client) receives the incoming requests and distributes these requests to different physical machines for further execution. Back end nodes serves the incoming requests.

Class agnostic load balancing – the front end machine are agnostic to nature of request. This means that the front end machine is neither aware of the type of client from which the request originates nor aware of the request category.

Class aware load balancing – The front end node additionally inspect the type of client making the request and/or the type of service requested before deciding which back end node should service the request.

Admission Control – These algorithms plays important role in deciding the set of requests that should be admitted into the application server when the server experiences very heavy loads. The objective of admission control mechanisms is to police the incoming requests and identify when the system faces overload situations.

Request based algorithms – reject new requests if the servers are running to their capacity. The disadvantage is that client's session may consist of multiple requests that are not necessarily unrelated. Some requests are rejected even if there are others that are honored.

Session based algorithms – ensure that longer sessions are completed and any new sessions are rejected. Once a session is admitted into the server, all future requests belonging to that session are admitted as well, even though new sessions are rejected by system.

SLA Types

Infrastructure SLA – Infrastructure provider manages and offers guarantees on availability of the infrastructure, namely server machine, power, network connectivity and so on. Enterprises manage their applications that are deployed on these server machines. The machines are leased to customers and are isolated from machines of other customers.

For example, SLAs can be

Hardware availability – 99% uptime in a calendar month

Power availability – 99.99% of the time in calendar month

Data center network availability – 99.99% of the time in calendar month

Application SLA – In application co-hosting model, the server capacity is available to the applications based solely on their resource demands. Hence the service providers are flexible in allocating and de-allocating computing resources among the co-located applications. Therefore the service providers are also responsible for ensuring to meet their customers application SLOs.

For example, SLAs can be

Web site response time (max of 3.5 sec per user request), Latency of web server (max of 0.2 sec per request), Latency of DB (max of 0.5 sec per query)

Infrastructure SLA vs Capacity Management

If temporal behavior of services with respect to resource demands is highly predictable, then capacity can be efficiently scheduled using reservations.

For **less predictable elastic workloads**., **exact scheduling of capacity may not be possible**.

Rather than that, **capacity planning and optimizations are required**.

- IaaS providers perform two complementary management tasks:

  (1) **Capacity planning** to make sure that SLA obligations are met as contracted with the service providers and;

  (2) **Continuous optimization** of resource utilization in specific workload to make the most efficient use of the existing capacity.

Thus, to deploy a service on a cloud, **a service provider orders suitable virtual hardware** and installs its application software on it.

From the IaaS provider, a given service configuration is a virtual resource array of black box resources, which correspond to the number of instances of resource type.

SLA Life Cycle

Each SLA goes through a sequence of steps starting from identification of terms and conditions, activation and monitoring of the stated terms and conditions, and eventual termination of contract once the hosting relationship ceases to exist. Such a sequence of steps is called SLA life cycle and consists of the following five phases:

1. Contract definition
2. Publishing and discove
3. Negotiation
4. Operationalization
5. De-commissioning

Contract Definition: Generally, service providers define a set of service offerings and corresponding SLAs using standard templates. These service offerings form a catalog. Individual SLAs for enterprises can be derived by customizing these base SLA templates.

Publication and Discovery. Service provider advertises these base service offerings through standard publication media, and the customers should be able to locate the service provider by searching the catalog. The customers can search different competitive offerings and shortlist a few that fulfill their requirements for further negotiation.

Negotiation: Once the customer has discovered a service provider who can meet their application hosting need, the SLA terms and conditions needs to be mutually agreed upon before signing the agreement for hosting the application. For a standard packaged application which is offered as service, this phase could be automated.

For customized applications that are hosted on cloud platforms, this phase is manual. The service provider needs to analyze the application's behavior with respect to scalability and performance before agreeing on the specification of SLA. At the end of this phase, the SLA is mutually agreed by both customer and provider and is eventually signed off. SLA negotiation can utilize the WS-negotiation specification

Operational: SLA operation consists of SLA monitoring, SLA accounting, and SLA enforcement.

SLA monitoring involves measuring parameter values and calculating the metrics defined as a part of SLA and determining the deviations.

On identifying the deviations, the concerned parties are notified. SLA Accounting involves capturing and archiving the SLA adherence for compliance.

As part of accounting, the application's actual performance and the performance guaranteed as a part of SLA is reported.

De-commissioning : SLA decommissioning involves termination of all activities performed under a particular SLA when the hosting relationship between the service provider and the service consumer has ended.

SLA specifies the terms and conditions of contract termination and specifies situations under which the relationship between a service provider and a service consumer can be considered to be legally ended

SLA LC Management – Cloud Applications

SLA management of applications hosted on cloud platforms involves five phases.

1. Feasibility
2. On-boarding
3. Pre-production
4. Production
5. Termination

Feasibility Analysis

Managed Service Providers(MSP) conducts the feasibility study of hosting an application on their cloud platforms.

This study involves three kinds of feasibility:

(a) **Technical feasibility**,

(b) **Infrastructure feasibility**

(c) **Financial feasibility**.

The technical feasibility of an application implies determining the following:

1. Ability of an application to scale out.
2. Compatibility of the application with the cloud platform being used within the MSP's data center.
3. The need and availability of a specific hardware and software required for hosting and running of the application.
4. Preliminary information about the application performance and whether they can be met by the MSP.

Performing the infrastructure feasibility involves determining the availability of infrastructural resources in sufficient quantity so that the projected demands of the application can be met.

The financial feasibility study involves determining the approximate cost to be incurred by the MSP and the price the MSP charges the customer so that the hosting activity is profitable to both of them.

A feasibility report consists of the results of the above three feasibility studies. The report forms the basis for further communication with the customer. Once the provider and customer agree upon the findings of the report, the outsourcing of the application hosting activity proceeds to the next phase, called "on boarding" of application.

Only the basic feasibility of hosting an application has been carried in this phase. However, the detailed runtime characteristics of the application are studied as part of the on-boarding activity

On-boarding of Application:

Once the customer and the MSP agree in principle to host the application based on the findings of the feasibility study, the application is moved from the customer servers to the hosting platform.

Moving an application to the MSP's hosting platform is called on-boarding.

As part of the on-boarding activity, the MSP understands the application runtime characteristics using runtime profilers*.

*** Profiling is a method of gathering performance data in any development or deployment scenario. This is for developers and system administrators who want to gather information about application performance**

On-boarding :

Packing of the application for deploying on physical or virtual environments. Application packaging is the process of creating deployable components on the hosting platform (could be physical or virtual).Open Virtualization Format (OVF) standard is used for packaging the application for cloud platform.

The packaged application is executed **directly on the physical servers** to capture and analyze the application performance characteristics. It allows **the functional validation of customer's application**. Besides, it provides a **baseline performance value** for the application in **non virtual environment**.

This can be used as **one of the data points for customer's performance expectation** and for **application SLA**. Additionally, it helps to identify the nature of application—that is, whether it is CPU-intensive or I/O intensive or network-intensive and the potential performance bottlenecks.

On-boarding :

The application is **executed on a virtualized platform** and the application performance characteristics are **noted again**. Important performance characteristics like the application's **ability to scale (out and up) and performance bounds** (minimum and maximum performance) are noted.

Based on the measured performance characteristics, **different possible SLAs are identified**. The **resources required** and the costs involved for each SLA are also computed.

Once the customer agrees to the set of SLAs and the cost, the MSP starts creating different policies required by the data center for automated management of the application. This implies that the management system should automatically infer the amount of system resources that should be allocated/de-allocated to/from appropriate components of the application when the load on the system increases/decreases.

Pre-Production

Once the determination of policies is completed as discussed in previous phase, the application is hosted in a simulated production environment.

It facilitates the customer to verify and validate the MSP's findings on application's runtime characteristics and agree on the defined SLA. Once both parties agree on the cost and the terms and conditions of the SLA, the customer sign-off is obtained. On successful completion of this phase the MSP allows the application to go on-live.

Production

In this phase, the application is made accessible to its end users under the agreed SLA.

However, there could be situations when the managed application tends to behave differently in a production environment compared to the preproduction environment.

This in turn may cause sustained breach of the terms and conditions mentioned in the SLA. Additionally, customer may request the MSP for inclusion of new terms and conditions in the SLA.

If the application SLA is breached frequently or if the customer requests for a new non-agreed SLA, the on-boarding process is performed again. In the case of the former, on-boarding activity is repeated to analyze the application and its policies with respect to SLA fulfillment. In case of the latter, a new set of policies are formulated to meet the fresh terms and conditions of the SLA.

Termination

When the customer wishes to withdraw the hosted application and does not wish to continue to avail the services of the MSP for managing the hosting of its application, the termination activity is initiated.

On initiation of termination, all data related to the application are transferred to the customer and only the essential information is retained for legal compliance. This ends the hosting relationship between the two parties for that application, and the customer sign-off is obtained.

CLOUD COMPUTING

UNIT 8

Cloud Computing – Engagements in the Cloud

- Serverless Computing
- Backend as a Service (BaaS)
- Mobile BaaS (MBaaS)
- Function as a Service (FaaS)

Future Directions in Cloud

- After Migration, What Next
- Evolution of EDGE computing
- Multi-clouds, a de facto standard

Course Wrap Up

- Review BEL & AR & IR Leases
- Exam Pattern discussion

BaaS Defined

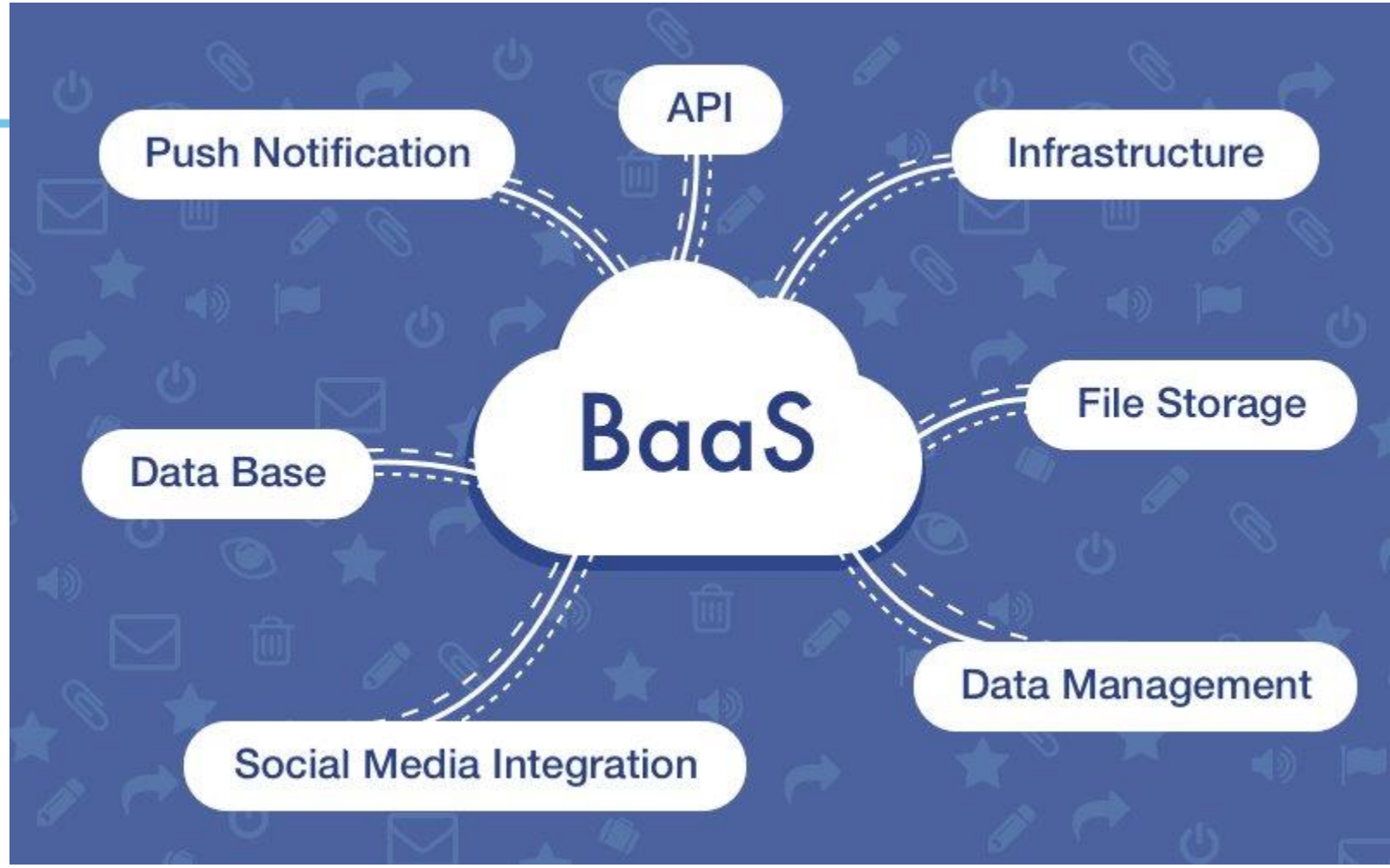
A platform that

- automates backend side development
- takes care of the cloud infrastructure

App teams 

- outsource the responsibilities of running and maintaining servers to a third party
- focus on the frontend or client-side development

Provides a set of tools to help developers to create a backend code speedily with help of ready to use features.



BaaS Architecture

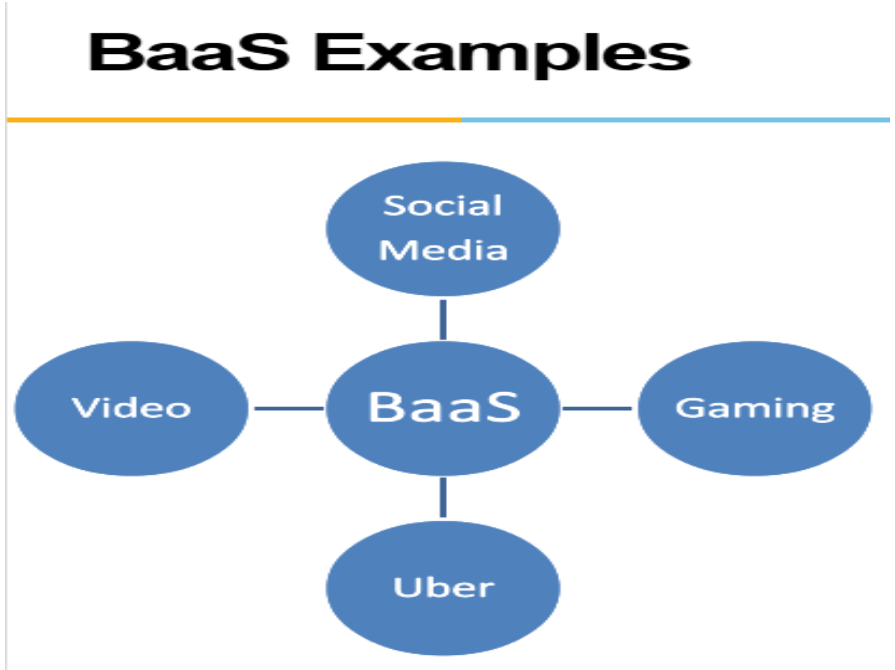
The **first layer** is the foundation and contains the database servers.

A **database cluster** will have at least two servers to replicate data and a backup routine to retrieve data.

Most BaaS providers use NoSQL databases on their technology stacks due to scaling flexibility, but there is a growing trend to use SQL databases like Postgres.

The **second layer** is the **application cluster** and contains multiple servers to process requests. The quantity of servers fluctuates throughout the time of the day, and auto-scaling procedures are necessary to fulfill the group with the correct amount of servers.

The **third layer** connects the application servers to the Internet, and it's composed of load balancers and CDNs.



BaaS vs IaaS

Imagine you would like to **build a new software project** and that you will **not use a BaaS**. The first step before you start developing the backend side code is to **set up the servers**. Here is how it will work:

•Login on AWS or any other cloud.

•Go to Instances

•Launch Instance

•Select the Operating System

•Instance Size, Type

•Configure Instance Details

- Number of instances ,Network, IP, Monitoring, Other settings like Auto Scaling, IAM, etc

•Add Storage

•Security Settings

All right, your **instance is up and running**, and now you can start coding! **Not really**! That is only the **first step** of the process, and you will still need to **install the web-server, database, framework, etc**.

After all that is done, you can start coding. The **time** to perform this process can range from a **few hours** (for a small project with skilled backend developers) to more than a **day for large environments**.

This same process using a **backend as a service will be done with a few clicks and take no more than a few minutes**.

BaaS Pro & Con

Advantages

Speedy Development

Reduced Development price

Serverless, and no need to manage infrastructure

Disadvantages

Less flexible as compared to custom coding / deployments

Less customization in comparison to a custom backend

Vendor lock-in possible

Serverless Computing

Serverless computing is a method of providing **backend services on an as-used basis**.

A serverless provider allows users to write and deploy code without the hassle of worrying about the underlying infrastructure.

A company that gets backend services from a serverless vendor is **charged** based on **their computation** and do not have to **reserve and pay for a fixed amount of bandwidth or number of servers**, as the service is auto-scaling.

Note that *despite the name serverless, physical servers are still used but developers do not need to be aware of them.*

BaaS vs Serverless

There is some overlap between **BaaS and serverless computing**, because in both the developer only has to write their application code and doesn't think about the backend. In addition, many BaaS providers also offer serverless computing services. However, there are significant operational differences between applications built using BaaS and a true serverless architecture.

How the application is constructed

The backends of serverless applications are broken up into functions, each of which responds to events and performs one action only. BaaS server-side functionalities, meanwhile, are constructed however the provider wants, and developers don't have to concern themselves with coding anything other than the frontend of the application.

When code runs

Serverless architectures are event-driven, meaning they run in response to events. Each function only runs when it is triggered by a certain event, and it does not run otherwise. Applications built with BaaS are usually not event-driven, meaning that they require more server resources.

Where code runs

Serverless functions can be run from anywhere on any machine, as long as they are still in communication with the rest of the application, which makes it possible to incorporate edge computing into the application's architecture by running code at the network's edge. BaaS is not necessarily set up to run code from anywhere, at any time (although it can be, depending on the provider).

How the application scales

Scalability is one of the biggest differentiators separating serverless architectures from other kinds of architecture. In serverless computing, the application automatically scales up as usage increases. The cloud vendor's infrastructure starts up ephemeral instances of each function as necessary. BaaS applications are not set up to scale in this way unless the BaaS provider also offers serverless computing and the developer builds this into their application.

Serverless on AWS

innovate achieve lead

Serverless is a way to describe the services, practices, and strategies that enables building more agile applications to foster innovations and responses to the changes faster

With serverless computing, infrastructure management tasks like capacity provisioning and patching are handled by AWS, developer can focus on only writing code that serves customers

Serverless services like AWS Lambda come with
automatic scaling
built-in high availability
and a pay-for-value billing model

Lambda is an event-driven compute service that enables to run code in response to events from over 150 natively-integrated AWS and SaaS sources all without managing any servers

Serverless Advantages

1. No server management

Serverless computing does not mean that there are no servers, but it definitely means that developers or companies do not have to worry about these servers. These servers are managed by vendors and developers can focus on expanding their application without being worried about their server capacity.

2. Lower cost

This is, obviously, very cost effective because a developer is only paying for the server space they are using.

3. Flexible scalability

This is one of the important advantages of serverless computing. A developer does not have to worry about scalability of their web application because serverless computing has the ability to scale according to traffic volumes.

4. Fewer things to worry

A company or developer does not have to worry about security, patch, bugs and many other things because their servers are managed by vendors.

Serverless Disadvantages

1. Not suitable for long term tasks:

Serverless are a good option for short term or real time tasks. If a task takes longer time to complete then the company might end up paying more for compute time. For example, if a large file has to be uploaded that takes more time then it will require additional functions till it's complete and the developer ends up paying more for that.

2. High non-performance penalty:

A business pays for what they use, but if they end up not using these functions then there will be a high non-performance penalty. These functions may suffer from cold start penalty as well and can be very slow when used after a certain period of time.

3. Vendors lock in:

When a business depends on a vendor for all its backend services for a web application, it ends up losing control over their hardware, updates and run times. If they want to switch then it gets very difficult. A business or developer has to re-engineer if they wish to switch to another service provider.

4. Security concerns:

New security concerns are introduced when a vendor provides servers to a business. This can be a huge problem if the application contains personal or sensitive information like credit card details. This happens because companies are not given their own physical servers, vendors will be running code for many customers on a single server.

FaaS

Function-as-a-Service (FaaS) is a serverless way to execute modular pieces of code on the edge. FaaS lets developers write and update a piece of code on the fly, which can then be executed in response to an event, such as a user clicking on an element in a web application. This makes it easy to scale code and is a cost-efficient way to implement microservices.

Microservice

The approach of building an application out of a set of modular components is known as microservice architecture.

Dividing an application into microservices is appealing to developers because it means they can create and modify small pieces of code which can be easily implemented into their codebases.

This is in contrast to monolithic architecture, in which all the code is interwoven into one large system.

With large monolithic systems, even a minor changes to the application requires a hefty deploy process.

FaaS eliminates this deploy complexity. Using serverless code like FaaS, web developers can focus on writing application code, while the serverless provider takes care of server allocation and backend services.

FaaS Benefits

Focus more on code, not infrastructure:

With FaaS, can divide the server into functions that can be scaled automatically and independently so don't have to manage infrastructure

This allows to focus on the app code and can dramatically reduce time-to-market

Pay only for the resources you use, when you use them:

With FaaS, you pay only when an action occurs

When the action is done, everything stops—no code runs, no server idles, no costs are incurred

FaaS is, therefore, cost-effective, especially for dynamic workloads or scheduled tasks

FaaS also offers a superior total-cost-of-ownership for high-load scenarios

Scale up or down automatically:

With FaaS, functions are scaled automatically, independently, and instantaneously, as needed

When demand drops, FaaS automatically scales back down

Get all the benefits of robust cloud infrastructure:

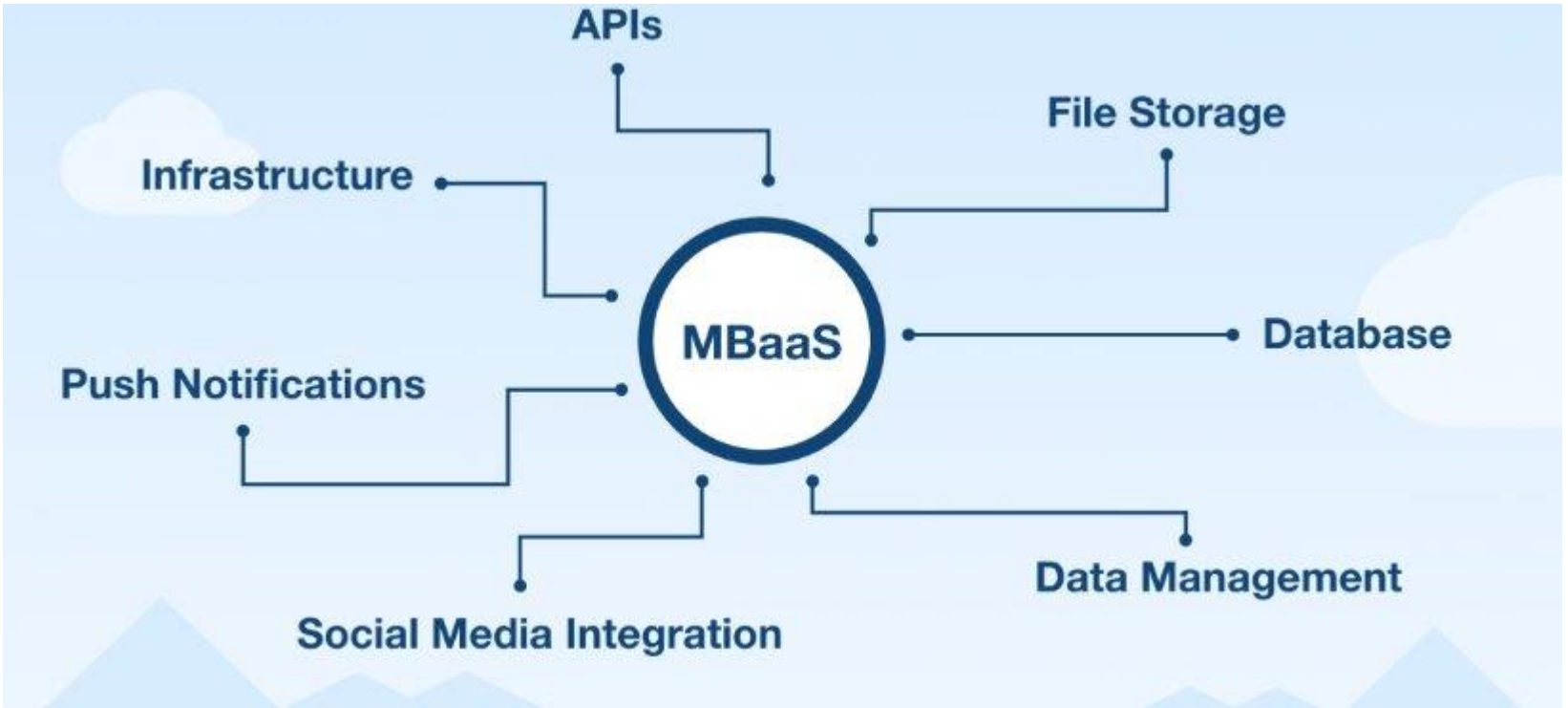
FaaS offers inherent high availability because it

is spread across multiple availability zones per geographic region

can be deployed across any number of regions without incremental costs

MBaaS

mBaaS – Mobile backend as a service helps developers in linking their applications, either mobile or websites, to the cloud via application programming interfaces (API). Other than this, the mobile backend as a service is also making grounds for helping developers in accelerating backend development, improve user management, provides push notifications, and much more. Common mBaaS features include database graphical interface, APIs, email verification, reset password, push-notifications, and more.



MBaaS Architecture

The first layer - Database

Is the foundation and contains the database servers

A database cluster has at least two servers to replicate data and a backup routine to retrieve data

The second layer - Application

Is the application cluster and contains multiple servers to process requests

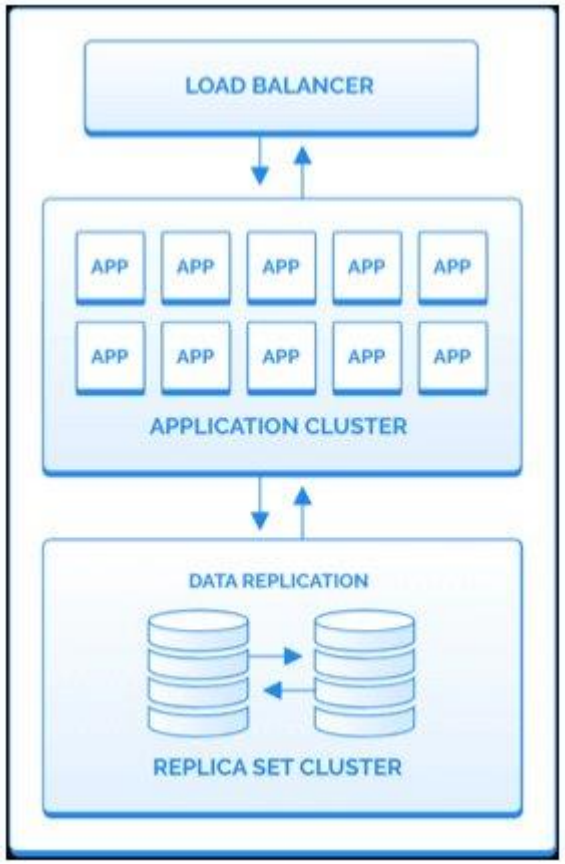
Quantity of servers fluctuates throughout the time of the day

Auto-scaling procedures are necessary to fulfill the correct number of servers

The third layer - Gateway

Connects the application servers to the Internet

Composed of load balancers and CDNs



EDGE Computing

Edge computing is a networking philosophy focused on bringing computing as close to the source of data as possible in order to reduce latency and bandwidth use.

In simpler terms, edge computing means running fewer processes in the cloud and moving those processes to local places, such as on a user's computer, an IoT device, or an edge server.

Bringing computation to the network's edge minimizes the amount of long-distance communication that has to happen between a client and server.

What is the Edge?

For Internet devices, the network edge is where the device, or the local network containing the device, communicates with the Internet. The edge is a bit of a fuzzy term; for example a user's computer or the processor inside of an IoT camera can be considered the network edge, but the user's router, ISP, or local edge server are also considered the edge. The important takeaway is that the edge of the network is geographically close to the device, unlike origin servers and cloud servers, which can be very far from the devices they communicate with.

EDGE Computing Use Cases

Edge computing can be incorporated into a wide variety of applications, products, and services. A few possibilities include:

- Security system monitoring: As described above.
- IoT devices: Smart devices that connect to the Internet can benefit from running code on the device itself, rather than in the cloud, for more efficient user interactions.
- Self-driving cars: Autonomous vehicles need to react in real time, without waiting for instructions from a server.
- More efficient caching: By running code on a CDN edge network, an application can customize how content is cached to more efficiently serve content to users.
- Medical monitoring devices: It is crucial for medical devices to respond in real time without waiting to hear from a cloud server.
- Video conferencing: Interactive live video takes quite a bit of bandwidth, so moving backend processes closer to the source of the video can decrease lag and latency.

1.Multi-cloud: The use of multiple cloud providers is becoming increasingly common, as companies seek to avoid vendor lock-in and take advantage of the unique capabilities of different cloud platforms.

2.Serverless Computing: Serverless computing is gaining in popularity, as it allows developers to write and run applications without having to worry about managing infrastructure.

3.Edge Computing: With the growth of IoT devices and real-time applications, there is a need for computing resources to be located closer to the source of data. Edge computing involves processing data at or near the edge of the network, rather than sending it back to the cloud.

4.Artificial Intelligence and Machine Learning: Cloud providers are incorporating AI and machine learning capabilities into their platforms, making it easier for developers to build intelligent applications.

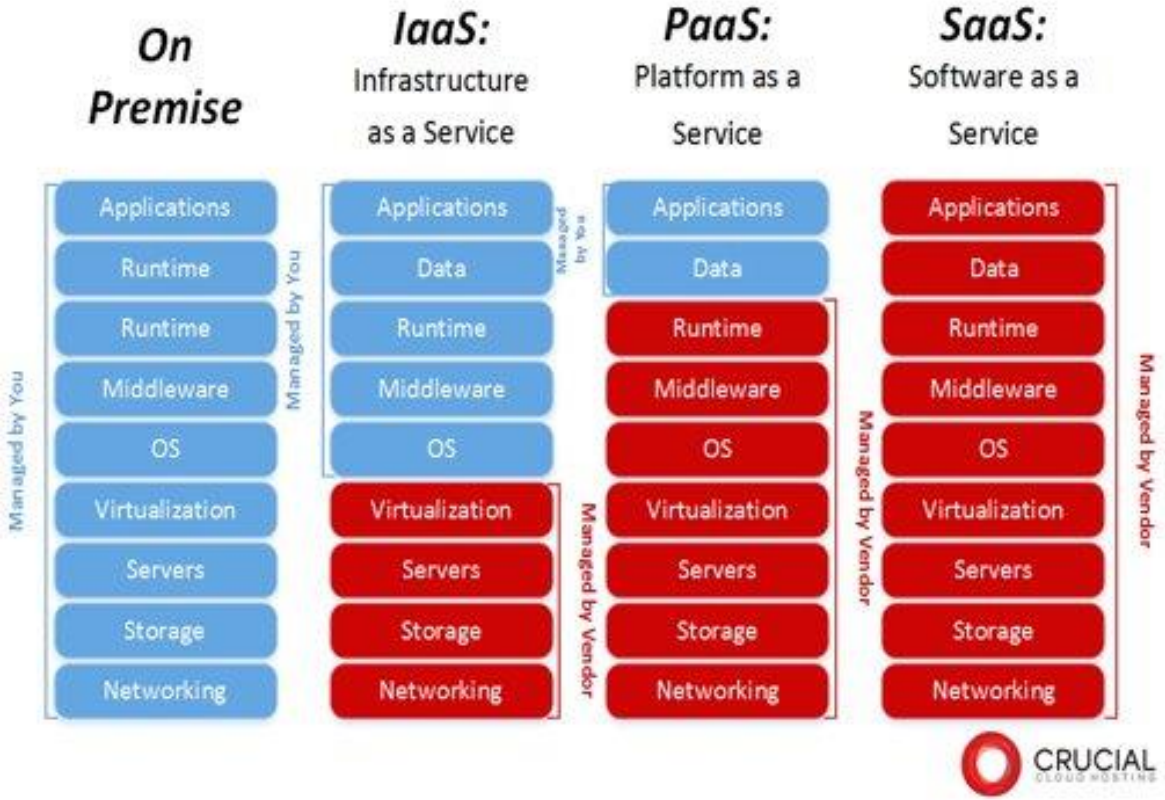
5.Quantum Computing: Quantum computing is still in its early stages, but it has the potential to transform cloud computing by providing a significant boost in computing power.

6.Green Computing: With the growing concern for the environment, cloud providers are making efforts to reduce their carbon footprint by using renewable energy and implementing energy-efficient technologies.

7.Hybrid Cloud: As more companies adopt cloud computing, hybrid cloud solutions that combine on-premises and cloud-based infrastructure are becoming increasingly popular.

8.Security and Compliance: With the increasing number of data breaches and cyber threats, cloud providers are investing in advanced security and compliance measures to protect their customers' data.

Course area	IaaS	PaaS	SaaS
Cloud computing	– Virtualization – Software defined networks – Software defined data centres – Cloud storage	– Cloud based simulation – Development of cloud software (Google App Engine, Windows Azure, Amazon)	– Communication apps – Cloud storage apps – Social computing apps – Apps management – Web hosting service
Mobile technologies	– Software defined radio networks	– SMS API – Mobile application development – Mobile agents – Mobile cloud applications	– Mobile commerce apps – M-payment apps – Mobile learning apps – Mobile social apps
Internet of Things	– Software defined wireless sensor networks – Smart environments	– API for accessing the sensor data – API for context-aware applications – API for wearable computing applications	– Software for smart device management
Big data	– Environment for Hadoop projects – Environment for MongoDB projects	– Map-reduce API	– Data analysis – Visualization
IT management	– Cloud management	– Salesforce PaaS – Heroku PaaS	– Project management software – CRM software
Computer simulation and virtual reality	– Resources for simulation execution – Environment for rendering	– API for developing 3D models	– Web simulation software – 3D modelling software tools



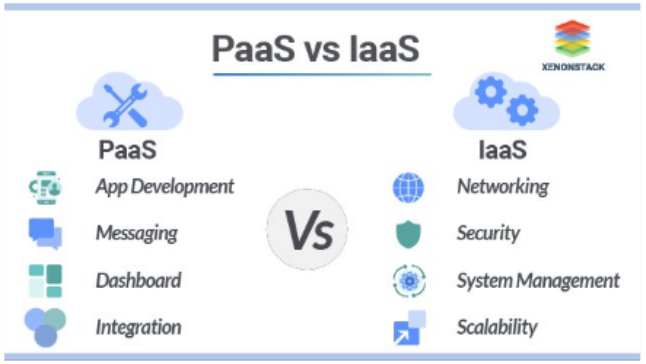
Building Blocks - PaaS

- PaaS providers can assist developers from the conception of their original ideas to the creation of applications, and through to testing and deployment.
- Below are some of the features that can be included with a PaaS offering:
  - ❑ Operating system
  - ❑ Server-side scripting environment
  - ❑ Database management system
  - ❑ Server Software
  - ❑ Support
  - ❑ Storage
  - ❑ Network access
  - ❑ Tools for design and development
  - ❑ Hosting

PaaS vs IaaS

PaaS, which is similar in many ways to Infrastructure as a Service, is differentiated from IaaS by the addition of value added services and comes in two distinct flavours;

1. A collaborative platform for software development, focused on workflow management regardless of the data source being used for the application. An example of this approach would be Heroku, a PaaS that utilizes the Ruby on Rails development language.
2. A platform that allows for the creation of software utilizing proprietary data from an application. This sort of PaaS can be seen as a method to create applications with a common data form or type. An example of this sort of platform would be the Force.com. PaaS from Salesforce.com which is used almost exclusively to develop applications that work with the Salesforce.com CRM



PaaS Disadvantages

•Since users rely on a provider's infrastructure and software, vendor lock-in can be an issue in PaaS environments.

•Other risks associated with PaaS are provider downtime or a provider changing its development roadmap.

•If a provider stops supporting a certain programming language, users may be forced to change their programming language, or the provider itself. Both are difficult and disruptive steps.

PaaS Advantages

Advantages

•Users don't have to invest in physical infrastructure

• PaaS allows developers to frequently change or upgrade operating system features. It also helps development teams collaborate on projects.

•Makes development possible for 'non-experts'

•Teams in various locations can work together

•Security is provided, including data security and backup and recovery.

•Adaptability; Features can be changed if circumstances dictate that they should.

•Flexibility; customers can have control over the tools that are installed within their platforms and can create a platform that suits their specific requirements. They can 'pick and choose' the features they feel are necessary.

AZURE SREVICES

Compute Services

- **Virtual Machine**

  This service enables you to create a virtual machine in Windows, Linux or any other configuration in seconds.

- **Cloud Service**

  This service lets you create scalable applications within the cloud. Once the application is deployed, everything, including provisioning, load balancing, and health monitoring, is taken care of by Azure.

- **Service Fabric**

  With service fabric, the process of developing a microservice is immensely simplified. Microservice is an application that contains other bundled smaller applications.

- **Functions**

  With functions, you can create applications in any programming language. The best part about this service is that you need not worry about hardware requirements while developing applications because Azure takes care of that. All you need to do is provide the code.

Networking

- **Azure CDN**

  Azure CDN (Content Delivery Network) is for delivering content to users. It uses a high bandwidth, and content can be transferred to any person around the globe. The CDN service uses a network of servers placed strategically around the globe so that the users can access the data as soon as possible.

- **Express Route**

  This service lets you connect your on-premise network to the Microsoft cloud or any other services that you want, through a private connection. So, the only communications that will happen here will be between the enterprise network and the service that you want.

- **Virtual network**

  The virtual network allows you to have any of the Azure services communicate with one another privately and securely.

- **Azure DNS**

  This service allows you to host your DNS domains or system domains on Azure.

AZURE SERVICES

Storage

- **Disk Storage**

  This service allows you to choose from either HDD (Hard Disk Drive) or SSD (Solid State Drive) as your storage option along with your virtual machine.

- **Blob Storage**

  This service is optimized to store a massive amount of unstructured data, including text and even binary data.

- **File Storage**

  This is a managed file storage service that can be accessed via industry SMB (server message block) protocol.

- **Queue Storage**

  With queue storage, you can provide stable message queuing for a large workload. This service can be accessed from anywhere in this world.

Ten design principles for Azure applications

Article • 07/19/2022 • 2 minutes to read • 11 contributors

Follow these design principles to make your application more scalable, resilient, and manageable.

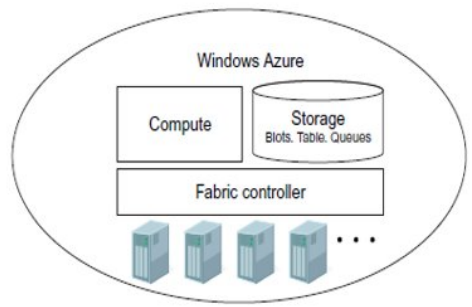
- **Design for self healing.** In a distributed system, failures happen. Design your application to be self healing when failures occur.
- **Make all things redundant.** Build redundancy into your application, to avoid having single points of failure.
- **Minimize coordination.** Minimize coordination between application services to achieve scalability.
- **Design to scale out.** Design your application so that it can scale horizontally, adding or removing new instances as demand requires.
- **Partition around limits.** Use partitioning to work around database, network, and compute limits.
- **Design for operations.** Design your application so that the operations team has the tools they need.
- **Use managed services.** When possible, use platform as a service (PaaS) rather than infrastructure as a service (IaaS).
  - **Use an identity service.** Use an identity as a service (IDaaS) platform instead of building or operating your own.
- **Use the best data store for the job.** Pick the storage technology that is the best fit for your data and how it will be used.
- **Design for evolution.** All successful applications change over time. An evolutionary design is key for continuous innovation.
- **Build for the needs of business.** Every design decision must be justified by a business requirement.

Azure Introduction

- Windows Azure provides a **platform to develop applications** using a range of available technologies and programming languages.
- It offers to create and deploy applications using .net platform, which is Microsoft's own application development technology. In addition to .net, there are many more technologies and languages supported. For example, Java, PHP, Ruby, Oracle, Linux, MySQL, Python.
- Windows Azure applications are scaled by creating **multiple instances** of the application.
- The **number of instances** needed by the **application is specified by the developer** while hosting the applications.
- If traffic is increased or decreased on the website or web application, it can be managed easily by logging in to Windows Azure management portal and specifying the instances. Load balancing can also be automated which would allow Azure to make the decision itself as when to assign more resources to application.
- Web applications support .net, java, python, php and node.js. Tasks such as scaling, and backups can be easily automated.
- A new feature called 'webjobs' is available, which is a kind of batch processing service. Webjobs can also be scaled and scheduled.
- The mobile application platforms supported are Xamarin iOS, Xamarin Android and IOS.
- Azure platform is developed in such a way that developers need to **concentrate on only the development part** and need not **worry about other technical stuff outside their domain**. Thus most of the administrative work is done by Azure itself.

Azure Runtime Environment

- The Windows Azure runtime environment provides a scalable compute and storage hosting environment along with management capabilities. It has three major components: Compute, Storage and the Fabric Controller
- The hosting environment of Azure is called the **Fabric Controller**. It has a pool of individual systems connected on a network and automatically manages resources by load balancing and geo-replication. It manages the application lifecycle without requiring the hosted apps to explicitly deal with the scalability and availability requirements. Each physical machine hosts an Azure agent that manages the machine.
- The **Azure Compute Service** provides a Windows-based environment to run applications written in the various languages and technologies supported on the Windows platform.
- The Windows **Azure storage service** provides scalable storage for applications running on the Windows Azure in multiple forms. It enables storage for binary and text data, messages and structured data through support for features called Blobs, Tables, Queues and Drives.



Azure fabric Controller

- Fabric Controller is a significant part of Windows Azure architecture.
- Inside the data centre, there are many machines or servers aggregated by a switch. We can say that fabric controller is a brain of the azure service that analyses the processes and makes decisions.
- **Fabrics** are group of machines in Microsoft's data centre which are aggregated by a switch. The group of these machines is called **cluster**.
- Each cluster is managed and owned by a **fabric controller**. They are replicated along with these machines. It manages everything inside those machines, for e.g., load balancers, switches, etc. Each machine has a **fabric agent** running inside it and fabric controller can communicate with each fabric agent
- When a user chooses one of the virtual machine, the operating system, patch updates and software updates are performed by fabric controller. It decides where the new application should run which is one of the most important functions of Fabric Controller. It also selects the physical server to optimize hardware utilization
- When a new application is published in Azure, an application configuration file written in XML is also attached. The fabric controller reads those files in Microsoft datacenter and makes the setting accordingly

Azure Components

- When the **system is running**, services are **monitored and one can access event logs, trace**/ debug data, performance counters, IIS web server logs, crash dumps, and other log files.
- **This information can be saved in Azure storage**. Note that there is **no debugging capability** for running cloud applications, but **debugging is done from a trace**.
- Like most PaaS services, Windows Azure defines a programming **model** specific to the platform, which is called the **Web role**- **Worker role** model.
- **Cloud Service Role**: In Azure, a Cloud Service Role is a collection of managed, load-balanced, Platform-as-a-Service virtual machines that work together to perform common tasks. Cloud Service Roles are managed by **Azure fabric controller** and provide the **ultimate combination of scalability, control, and customization**
- **Web Role** is a Cloud Service role in Azure that is configured and customized to **run web applications developed on programming languages** / technologies that are **supported by Internet Information Services** (IIS), such as ASP.NET, PHP, Windows Communication Foundation and Fast CGI
- **Worker Role** is any role in Azure that **runs applications and services level tasks**, which generally do not **require IIS**. In Worker Roles, IIS is not installed by default. They are mainly used to **perform supporting background processes** along with Web Roles and do tasks such as automatically compressing uploaded images, run scripts when something changes in database, get new messages from queue and process and more.

SaaS Architecture

Run by

- Bandwidth technologies
- The cost of a PC has been reduced significantly with more powerful computing but the cost of application software has not followed
- Timely and expensive setup and maintenance costs
- Licensing issues for business are contributing significantly to the use of illegal software and piracy.

Scalable

- Multitenant efficient
- Configurable

Scaling the application - maximizing concurrency, and using application resources more efficiently

- i.e. optimizing locking duration, statelessness, sharing pooled resources such as threads and network connections, caching reference data, and partitioning large databases.

Multi-tenancy – important architectural shift from designing isolated, single-tenant applications

- One application instance must be able to accommodate users from multiple other companies at the same time
- All transparent to any of the users.
- This requires an architecture that maximizes the sharing of resources across tenants
- is still able to differentiate data belonging to different customers.

Configurable - a single application instance on a single server has to accommodate users from several different companies at once

- To customize the application for one customer will change the application for other customers as well.
- Traditionally customizing an application would mean code changes
- Each customer uses metadata to configure the way the application appears and behaves for its users.
- Customers configuring applications must be simple and easy without incurring extra development or operation costs

SaaS Model Comparison

Traditional	Software as a Service
Designed for customers to install, manage and maintain.	Designed from the outset up for delivery as Internet-based services.
Architect solutions to be run by an individual company in a dedicated instantiation of the software.	Designed to run thousands of different customers on a single code.
Infrequent, major upgrades every 18-24 months, sold individually to each installed base customer.	Frequent, "digestible" upgrades every 3-6 months to minimize customer disruption and enhance satisfaction.
Version control Upgrade fee	Fixing a problem for one customer fixes it for everyone
Streamlined, repeatable functionality via Web services, open APIs and standard connectors	May use open APIs and Web services to facilitate integration, but each customer must typically pay for one-off integration work.

SaaS User Benefits

- **Lower Cost of Ownership**
  - The software is paid when it is consumed, no large upfront cost for a software license Salesforce.com has a best-of-breed CRM system for $59.00 per user per month, with no upfront
  - Since no hardware infrastructure, installation, maintenance, and administration, budgeting is easy
  - The software is available immediately upon purchasing
- **Focus on Core Competency**
  - The IT saving on capital and effort allows the customer to remain focused on their core competency and utilize resources in more strategic areas.
- **Access Anywhere**
  - Users can use their applications and access their data anywhere they have an Internet connection and a computing device
  - This enhances the customer experience of the software and makes it easier for users to get work done fast
- **Freedom to Choose (or Better Software)**
  - The pay-as-you-go (PAYG) nature of SaaS enables users to select applications they wish to use and to stop using those that no longer meet their needs. Ultimately, this freedom leads to better software applications because vendors must be receptive to customer needs and wants.

- **New Application Types**
  - Since the barrier to use the software for the first time is low, it is now feasible to develop applications that may have an occasional use model. This would be impossible in the perpetual license model. If a high upfront cost were required the number of participants would be much smaller.
- **Faster Product Cycles**
  - Product releases are much more frequent, but contain fewer new features than the typical releases in the perpetual license model because the developer know the environment the software needs to run
  - This new process gets bug fixes out faster and allows users to digest new features in smaller bites, which ultimately makes the users more productive than they were under the previous model.