Introduction

Distributed Computing

- Is a computing system consists of multiple computers or processor machines connected through a network, which can be homogeneous or heterogeneous, but run as a single system.
- The CPUs in a distributed system can be physically close together and connected by a local network, or they can be geographically distant and connected by a wide area network.
- The heterogeneity in a distributed system supports any number of possible configurations in the processor machines, such as mainframes, PCs, workstations, and minicomputers.
- The goal of distributed computing is to make such a network work as a single computer.
- Distributed computing systems are advantageous over centralized systems due to..

- Scalability
- Redundancy or replication

High Performance Computing(HPC)

- A pool of processors connected /networked with other resources like memory, storage, and input and output devices, and the deployed software is enabled to run in the entire system of connected components.
- Processor machines can be of homogeneous or heterogeneous type.
- The legacy meaning of HPC- Supercomputers; it is not true in present-day computing scenarios.
- Examples of HPC -a small cluster of desktop computers or personal computers (PCs) to the fastest supercomputers.
- HPC systems are normally found in those applications where it is required to use or solve scientific problems.

Parallel Computing

- Parallel computing is one of the facets of HPC.
- A set of processors work cooperatively to solve a computational problem.
- These processor machines or CPUs are mostly of homogeneous type.

- In serial or sequential computers, the following apply:
- It runs on a single computer/processor machine having a single CPU.
- A problem is broken down into a discrete series of instructions.
- Instructions are executed one after another.
- In parallel computing, since there is simultaneous use of multiple processor machines, the following apply:
- It is run using multiple processors (multiple CPUs).
- A problem is broken down into discrete parts that can be solved concurrently.
- Each part is further broken down into a series of instructions.
- Instructions from each part are executed simultaneously on different processors.
- An overall control/coordination mechanism is employed.

Cluster Computing

- Consists of a set of the same or similar type of processor machines connected using a dedicated network infrastructure.
- All processor machines share resources such as a common home directory and have a software such as a message passing interface (MPI) implementation installed to allow programs to be run across all nodes simultaneously.
- This is also a kind of HPC category. The individual computers in a cluster can be referred to as nodes.
- The reason to realize a cluster as HPC is due to the fact that the individual nodes can work together to solve a problem larger than any computer can easily solve. And, the nodes need to communicate with one another in order to work cooperatively and meaningfully together to solve the problem in hand.

Grid Computing

- The computing resources in most of the organizations are underutilized but are necessary for certain operations.
- The idea of grid computing is to make use of such non-utilized computing power by the needy organizations, and thereby the return on investment (ROI) on computing investments can be increased.
- It is a network of computing or processor machines managed with a kind of software such as middleware, in order to access and use the resources remotely.
- The managing activity of grid resources through the middleware is called grid services. Grid services provide access control, security, access to data including digital libraries and databases, and access to large-scale interactive and long-term storage facilities.

- Grid computing is more popular due to the following:

- Its ability to make use of unused computing power, and thus, it is a cost-effective solution (reducing investments, only recurring costs)
- As a way to solve problems in line with any HPC-based application
- Enables heterogeneous resources of computers to work cooperatively and collaboratively to solve a scientific problem

Cloud Computing

- The computing trend moved toward cloud from the concept of grid computing, particularly when large computing resources are required to solve a single problem, using the ideas of computing power as a utility and other allied concepts.
- The potential difference between grid and cloud is that grid computing supports leveraging several computers in parallel to solve a particular application,
- while cloud computing supports leveraging multiple resources, including computing resources, to deliver a unified service to the end user.
- In cloud computing, the IT and business resources, such as servers, storage, network, applications, and processes, can be dynamically provisioned to the user needs and workload.
- In addition, while a cloud can provision and support a grid, a cloud can also support nongrid environments, such as a three-tier web architecture running on traditional or Web 2.0 applications.

Need for Cloud Computing

- The main reasons for the need and use of cloud computing are - convenience and reliability.
- In the past, we used to save files in USB flash drive, external hard drive, or CD and bring that device to a different place.
- Instead, saving a file to the cloud (e.g. Dropbox/OneDrive/Google Drive) ensures that we will be able to access it with any computer that has an Internet connection.
- The cloud also makes it much easier to share a file with friends, making it possible to collaborate over the web.
- There is always a risk that someone may try to gain access to our personal data, and therefore, it is important to choose an access control with a strong password and pay attention to any privacy settings for the cloud service.

Definition – Cloud Computing

- The formal definition of cloud computing comes from the National Institute of Standards and Technology (NIST):

  "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

5-4-3 Principles of Cloud Computing

- The 5-4-3 principles put forth by NIST describe

(a) the five essential characteristic features that promote cloud computing,

(b) the four deployment models that are used to narrate the cloud computing opportunities for customers while looking at architectural models, and

(c) the three important and basic service offering models of cloud computing.

Essential Characteristics

1. On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.
2. Broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and personal digital assistants [PDAs]).

3. Elastic resource pooling: The provider's computing resources are pooled to serve multiple consumers using a multitenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

   There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify the location at a higher level of abstraction (e.g., country, state, or data center).

4. Rapid elasticity: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

5. Measured service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

Deployment models

1. Private cloud: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.
2. Public cloud: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
3. Community cloud: The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.

4. Hybrid cloud: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

Service Offering Models

- IaaS
- PaaS
- SaaS

IaaS: Infrastructure as a Service

- cloud-based services, pay-as-you-go for services such as storage, networking, and virtualization.

When to use IaaS:

- Startups and small companies may prefer IaaS to avoid spending time and money on purchasing and creating hardware and software.
- Larger companies may prefer to retain complete control over their applications and infrastructure, but they want to purchase only what they actually consume or need.
- Companies experiencing rapid growth like the scalability of IaaS, and they can change out specific hardware and software easily as their needs evolve.
- Anytime you are unsure of a new application's demands, IaaS offers plenty of flexibility and scalability.

Cont.

- IaaS is fully self-service for accessing and monitoring computers, networking, storage, and other services.
- IaaS allows businesses to purchase resources on-demand and as-needed instead of having to buy hardware.
- These cloud servers are typically provided to the organization through a dashboard or an API, giving IaaS clients complete control over the entire infrastructure.
- As opposed to SaaS or PaaS, IaaS clients are responsible for managing aspects such as applications, runtime, OSes, middleware, and data.

IaaS Advantages

IaaS offers many advantages, including:

- The most flexible cloud computing model
- Easy to automate deployment of storage, networking, servers, and processing power
- Hardware purchases can be based on consumption
- Clients retain complete control of their infrastructure
- Resources can be purchased as-needed
- Highly scalable

IaaS Characteristics

- Resources are available as a service
- Cost varies depending on consumption
- Services are highly scalable
- Multiple users on a single piece of hardware
- Organization retain complete control of the infrastructure
- Dynamic and flexible

IaaS Limitations/Concerns

- Security - While the customer is in control of the apps, data, middleware, and the OS platform, security threats can still be sourced from the host or other virtual machines (VMs)
- Legacy systems operating in the cloud - While customers can run legacy apps in the cloud, the infrastructure may not be designed to deliver specific controls to secure the legacy apps.
- Internal resources and training
- Multi-tenant security - the vendor is required to ensure that other customers cannot access data deposited to storage assets by previous customers

PaaS: Platform as a Service

- PaaS delivers a framework for developers that they can build upon and use to create customized applications.

When to use PaaS:

- To streamline workflows when multiple developers are working on the same development project.
- PaaS is particularly beneficial if you need to create customized applications. This cloud service also can greatly reduce costs and it can simplify some challenges that come up if you are rapidly developing or deploying an app.

PaaS Advantages

- Simple, cost-effective development and deployment of apps
- Scalable
- Highly available
- Developers can customize apps without the headache of maintaining the software
- Significant reduction in the amount of coding needed
- Automation of business policy
- Easy migration to the hybrid model

PaaS Characteristics

- Builds on virtualization technology, so resources can easily be scaled up or down as your business changes
- Provides a variety of services to assist with the development, testing, and deployment of apps
- Accessible to numerous users via the same development application
- Integrates web services and databases

PaaS Limitations/Concerns

- Data Security
- Integrations
- Vendor lock-in
- Customization of legacy systems
- Runtime issues
- Operational limitations

SaaS: Software as a Service

-Utilizes the internet to deliver applications, which are managed by a third-party vendor, to its users.

- A majority of SaaS applications run directly through your web browser, which means they do not require any downloads or installations on the client side.

<u>When to use SaaS:</u>

- Start-ups or small companies that need to launch ecommerce quickly and don't have time for server issues or software
- Short-term projects that require quick, easy, and affordable collaboration
- Applications that aren't needed too often, such as tax software
- Applications that need both web and mobile access

SaaS Advantages

- Reduce the time and money spent on tedious tasks such as installing, managing, and upgrading software.

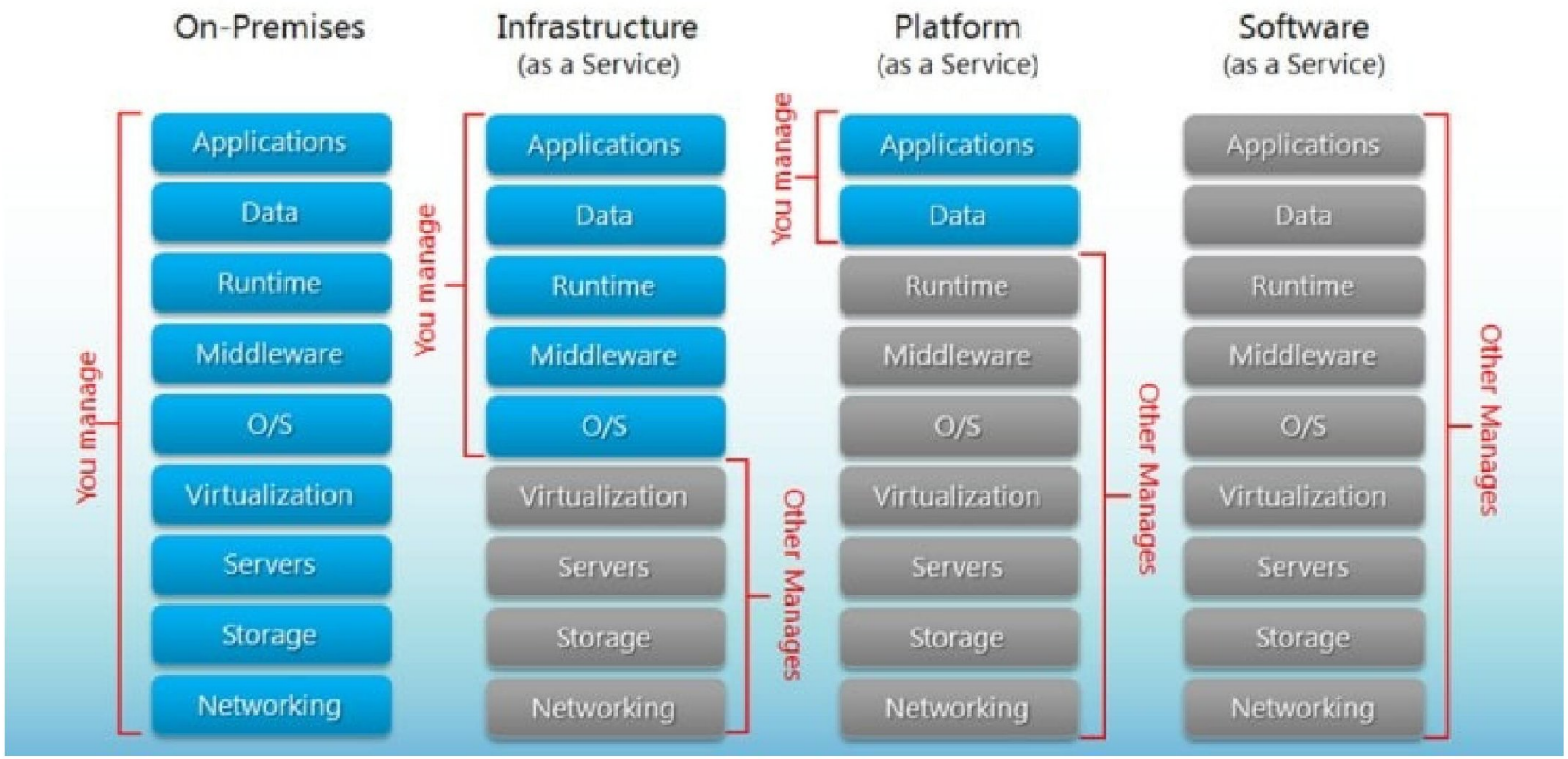
SaaS Characteristics

- Managed from a central location
- Hosted on a remote server
- Accessible over the internet
- Users not responsible for hardware or software updates

SaaS Limitations/Concerns

- Interoperability
- Vendor lock-in
- Lack of integration support
- Data security
- Customization
- Lack of control
- Feature limitations
- Performance and downtime

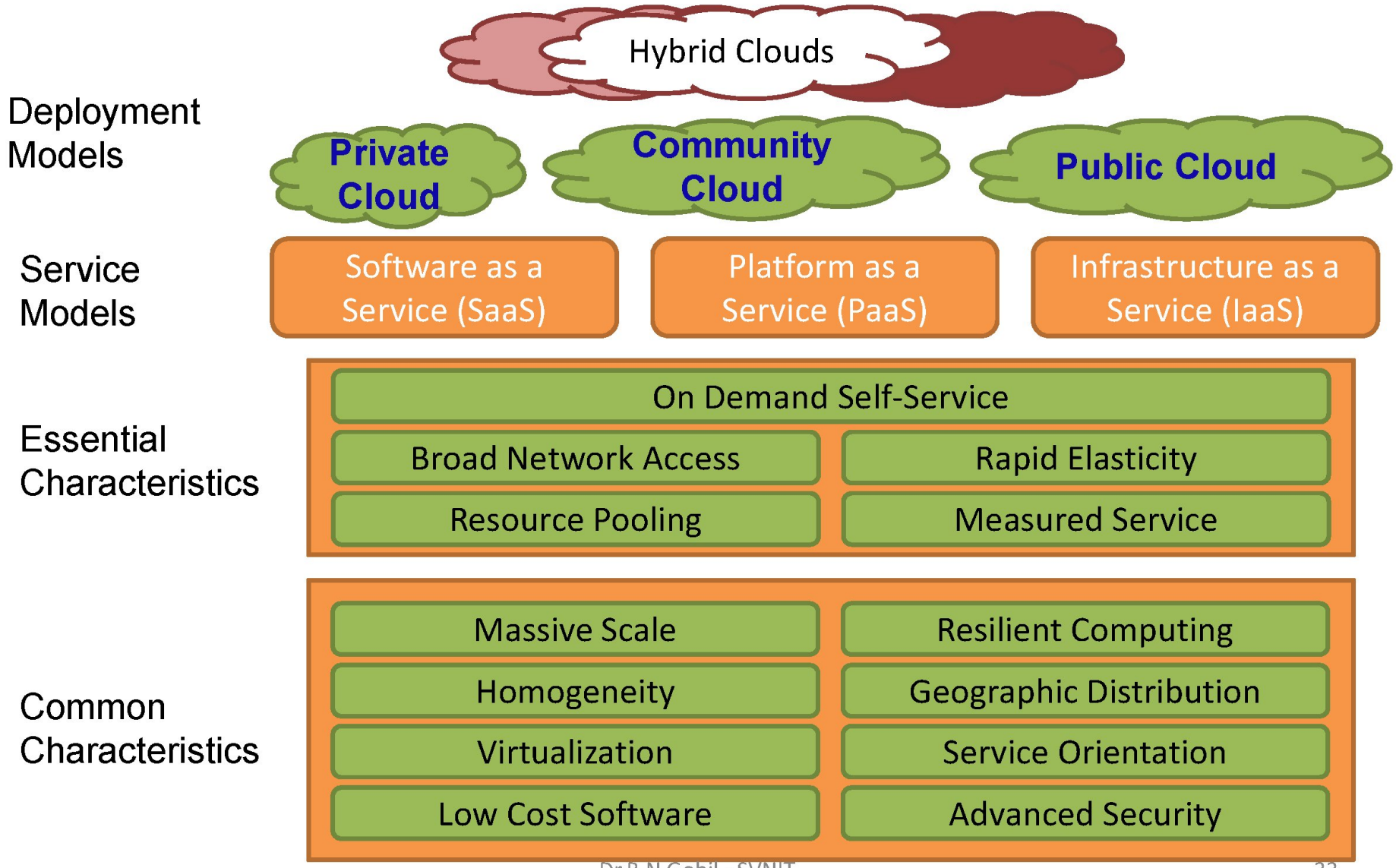
Service Offering Models



Examples

- SaaS examples: Google Apps(Gmail, Google Talk, Google Calendar, Google Docs, Google Videos and Google Cloud Connect), Salesforce(CRM), Dropbox, DocuSign, Slack(messages), GoToMeeting.
- PaaS examples: AWS Elastic Beanstalk, Windows Azure (mostly used as PaaS), Google App Engine, OpenShift – container platform.
- IaaS examples: AWS EC2, Rackspace Cloud, Google Compute Engine (GCE).

The NIST Cloud Def. Framework



NIST reference architecture

- A template description of the architecture, probably defined at different levels of abstraction
  - Highly abstract showing different functionalities
  - Lower level showing methods performing specific task
- Vendor-neutral description
- A conceptual model for discussing the technical requirements and operations of cloud computing
- A blueprint to guide developers in the design of (cloud) services and applications
  - Blueprint: compositions of interconnected services implementing reusable logic for building applications), list of functions and their interfaces (APIs), descriptions of their interactions

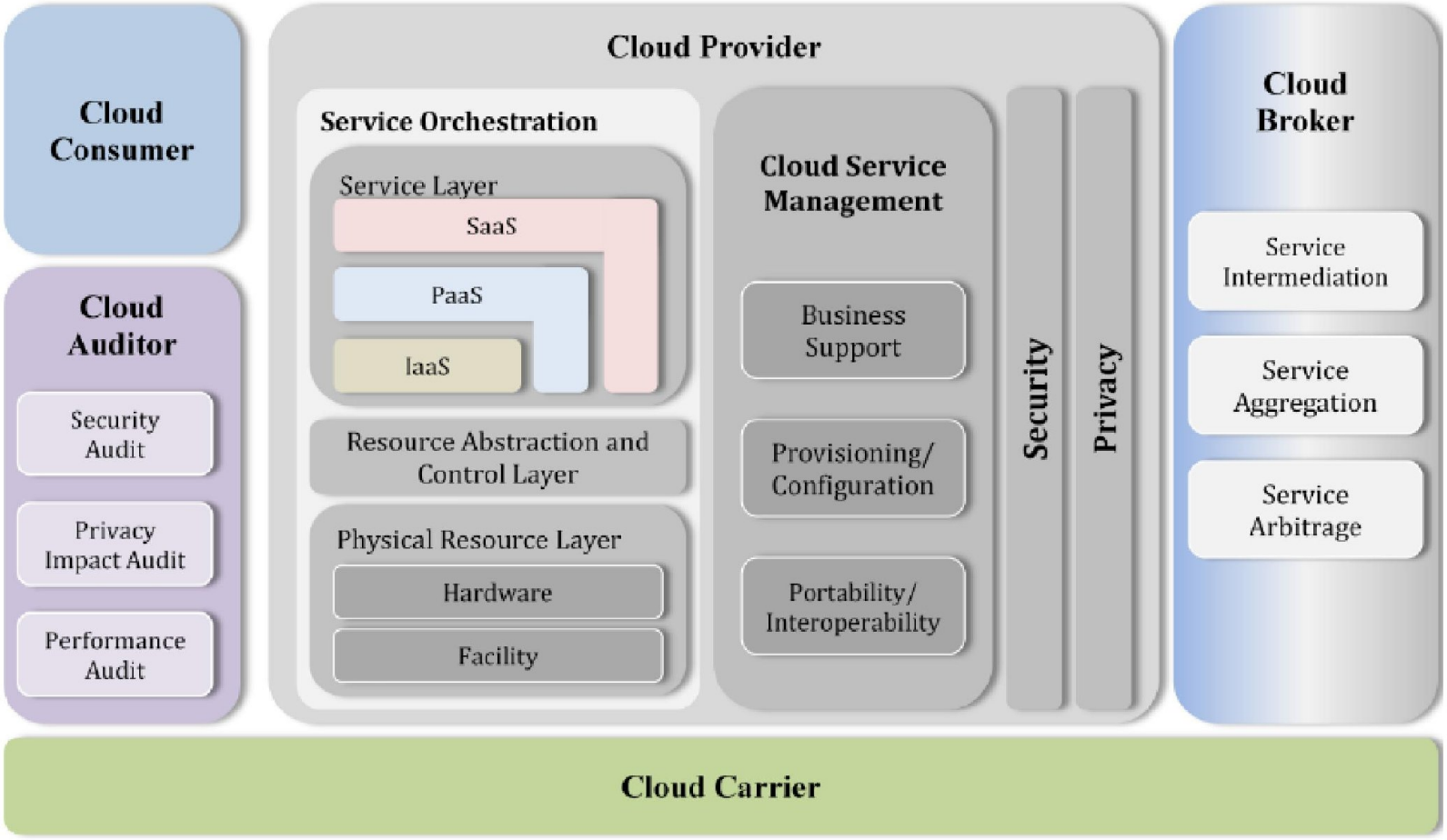
Cont.

- Actors and roles: core individuals or users with key responsibilities in system function
- Architectural components for managing and providing cloud services for
  - Deployment
  - Orchestration
  - Management
  - Security
  - Privacy

Actors and Roles

- Individuals or organizations with key roles
  - Consumer: acquires and uses services
  - Provider: the purveyor of services
  - Broker: intermediate between consumer – provider, they hide complexity of services or create new services
  - Auditor: independent performance, security monitoring and assessment of cloud services
  - Carrier: provides connectivity and transport of data and services between providers and consumers

Conceptual Reference Model



Cloud Consumer

- Browses the service catalogue of the provider
- Requests services depending on activities, usage scenarios
- Sets up service contracts with the providers
- May be billed for the service
  - SaaS consumers may be billed based on number of users, time of use, net bandwidth, storage volume
  - IaaS, PaaS consumers may be billed according to processing, storage, network resources, number of VMs, http calls, number of IPs used, net bandwidth, storage volume
- Consumers need SLAs to specify their performance requirements to be fulfilled by the provider (however SLAs are offered by cloud producers and in most cases aren't negotiable)

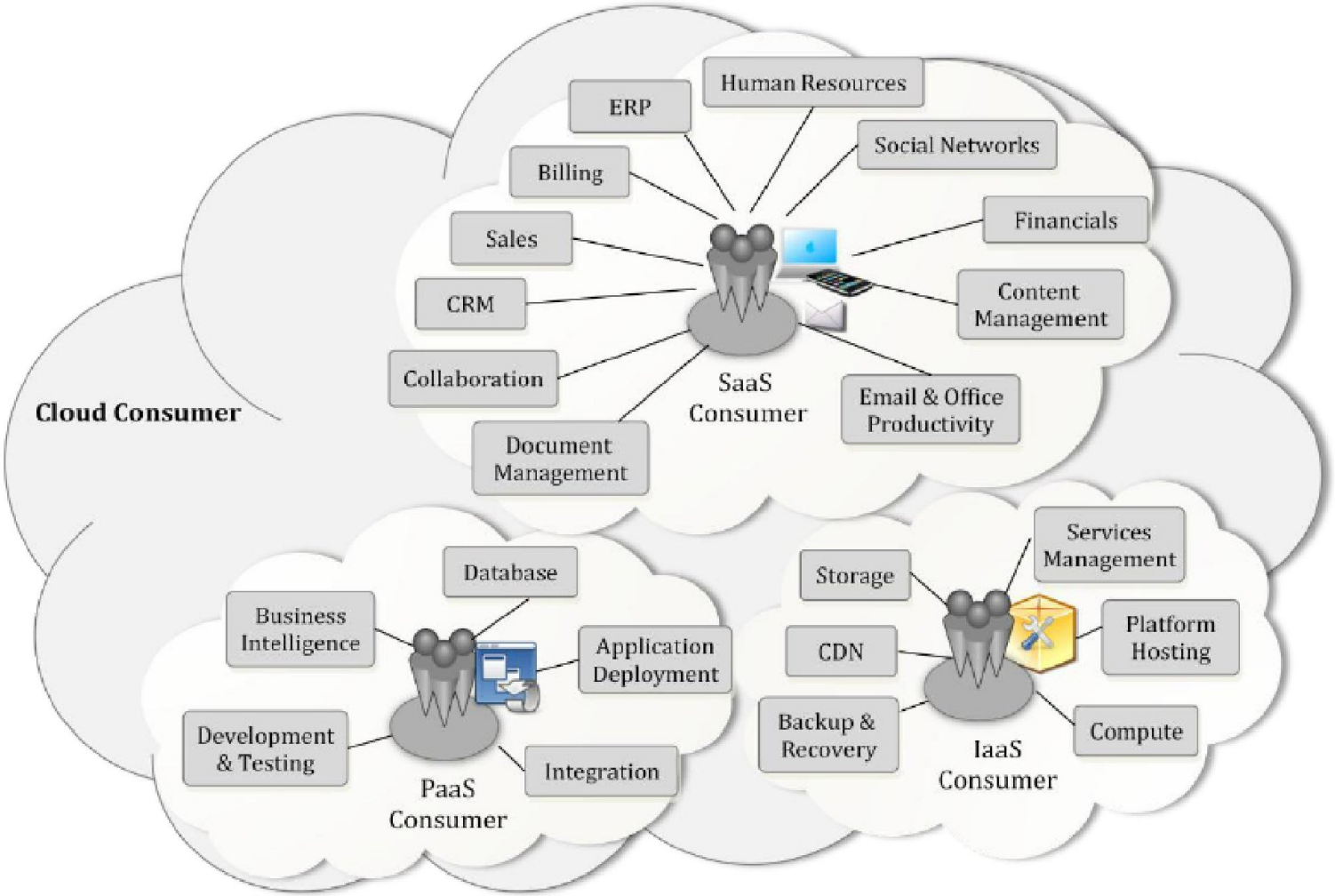
Service Level Agreements (SLAs)

- Contracts that are negotiated and agreed between provider and customers so to locate/reserve resources to satisfy consumers' requirements with efficiency and optimally resource and service usage
- To guarantee an agreed SLA, the auditor must be capable of measuring and monitoring relevant metrics (e.g., service availability, network metrics, storage metrics)
- Different SLA models must be considered for IaaS, PaaS and SaaS as each model sets different requirements
  - SLAs can be defined clearly for IaaS;
  - for PaaS and SaaS SLAs are still vague and difficult to be defined as these refer to higher levels of functionality but, can be agreed between providers / customers based on application requirements (business case) and business level plan

SLAs for IaaS

Parameter	Description
CPU capacity	CPU speed for Virtual Machines (VMs)
Memory size	Cash memory size for VM
Boot time	Time for VM to be ready for use
Storage	Storage size of data
Scale up	Max of VMs for one user
Scale down	Min number of VMs for one user
Scale up time	Time to increase number of VMs
Scale down time	Time to decrease number of VMs
Availability	Uptime of service in specific time

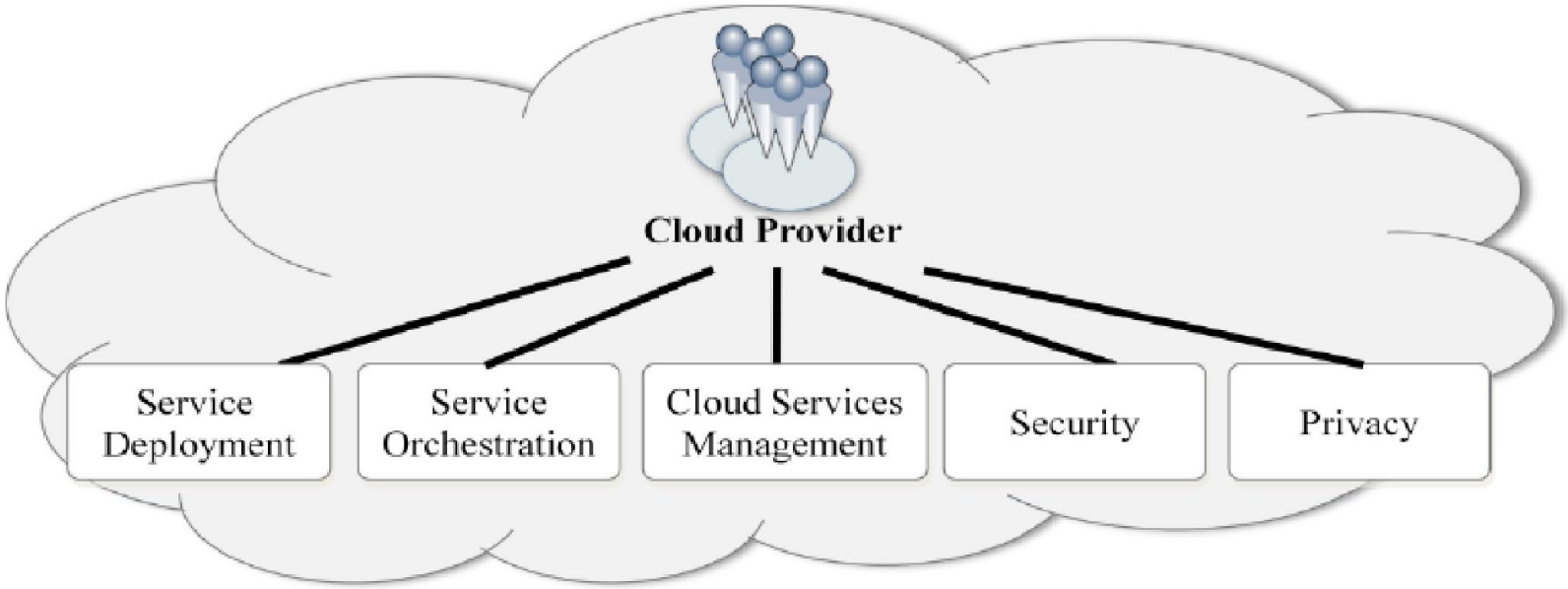
Examples of Cloud Services



Cloud Provider

- Acquires and manages the computing infrastructure
- Runs the cloud software, makes services available to interested parties
- Makes arrangements / contracts with consumers
- May also list SLAs i.e. Promises to consumers or limitations and obligations that consumers must accept
- Provider's pricing policy and SLAs are not negotiable in most cases

Responsibilities of Cloud Provider



Scope of Control (Provider)

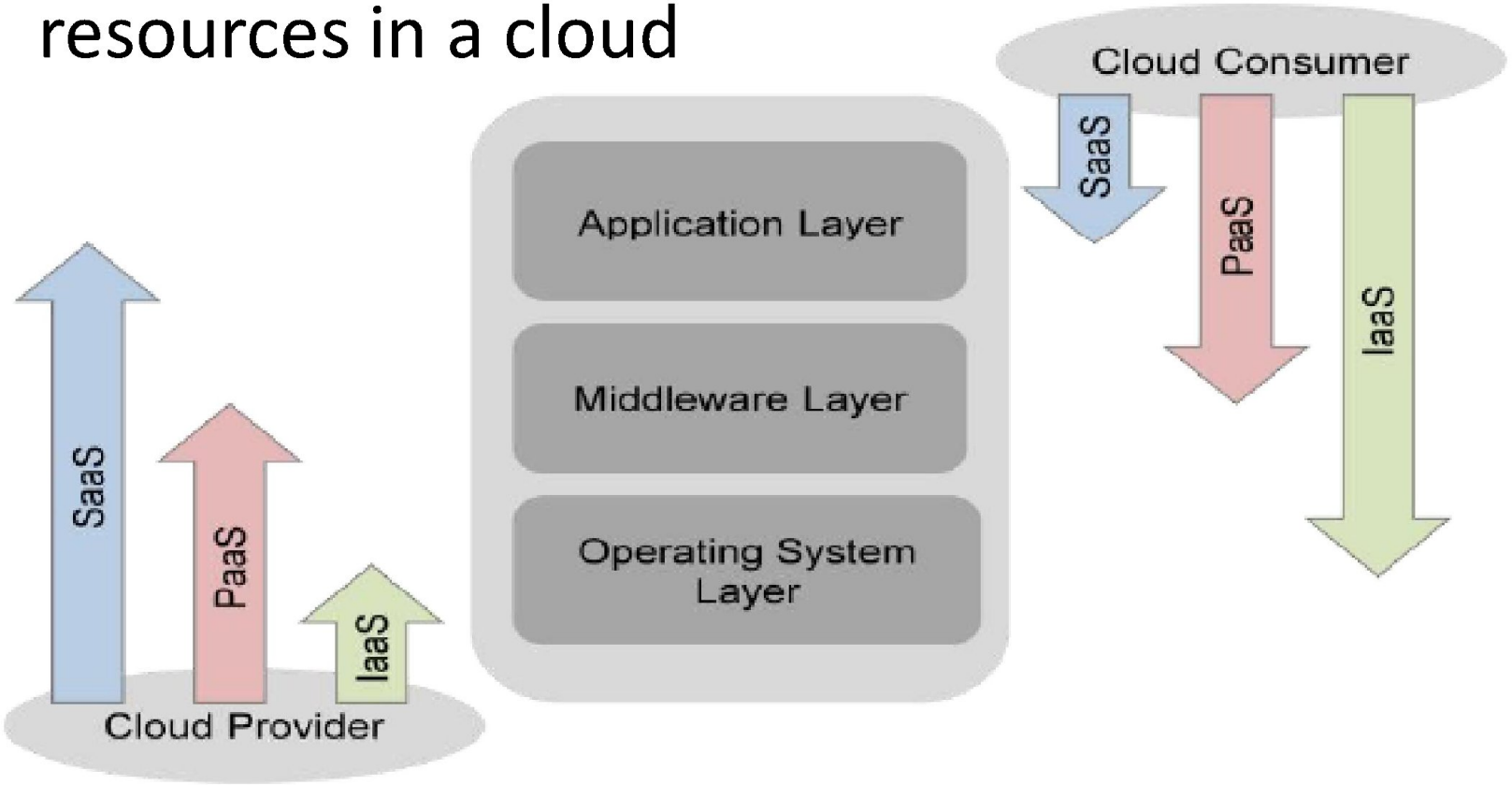
- Application layer: end-user apps and services used by SaaS consumers, installed/managed by PaaS consumers and SaaS providers
- Middleware layer (VM layer): provides building blocks for app development (libraries, dbms, Java VMs), used by PaaS consumers, installed/maintained/managed by PaaS providers, hidden from SaaS consumers
- OS layer: operating system VMs and drivers, hidden from SaaS /PaaS consumers, controlled by IaaS providers, used by IaaS consumers. An IaaS provider may allow multiple OS's as VMs

Scope of Control

- SaaS: Consumers have only limited administrative control of the applications and services
- PaaS: The provider manages infrastructure and provides tools of deployment of applications; the consumer has control over the application but limited / no access to the infrastructure (e.g. OS, servers, storage, drivers)
- IaaS: The provider acquires physical resources (servers, network, storage) and runs the software to make these available to IaaS, PaaS consumers through VMs; consumers have control over virtual software components (OS, network)

Scope of Control

- Provider and consumer share the control of resources in a cloud



Cloud Auditor

- Performs independent examination of cloud service controls and express opinion / issues evaluation
  - Ideally, have a contractual clause enabling 3rd parties to assess cloud operations
  - To determine the extend to which cloud operations are implemented/executed as planned and agreed
- Auditors objective is to verify conformance to standards (e.g. OCCI) or to security, privacy controls, performance, conformance to SLAs etc.
  - Issue security, privacy, performance audits

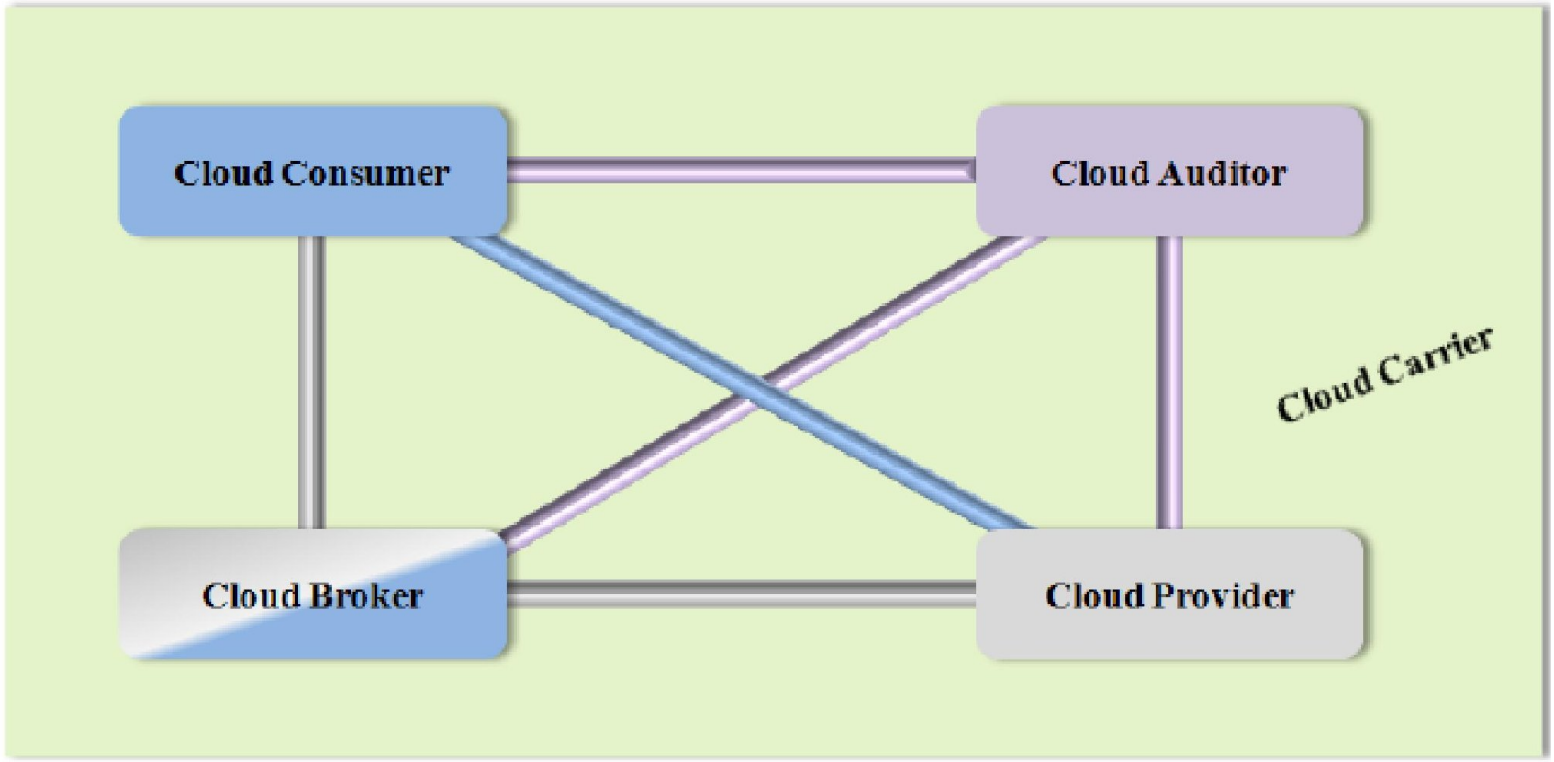
Cloud Broker

- Integration of cloud services by consumers can be too complex and can be requested from a cloud broker rather than from a provider directly
  - An entity/service operated by the provider or third party
- Provides services in three forms
  - Intermediation: presents the service to consumers (e.g. In catalogue), provides/enhances/improves a given service (e.g. by adding identity management, performance reporting, enhanced security)
  - Aggregation: combines and integrates multiple services into one
  - Arbitrage: the services being aggregated may change or come from different providers

Example Usage Scenario for Broker

- A consumer requests a service from a broker instead of contacting the provider directly
- The broker creates a new service by combining multiple services

Interactions between Actors



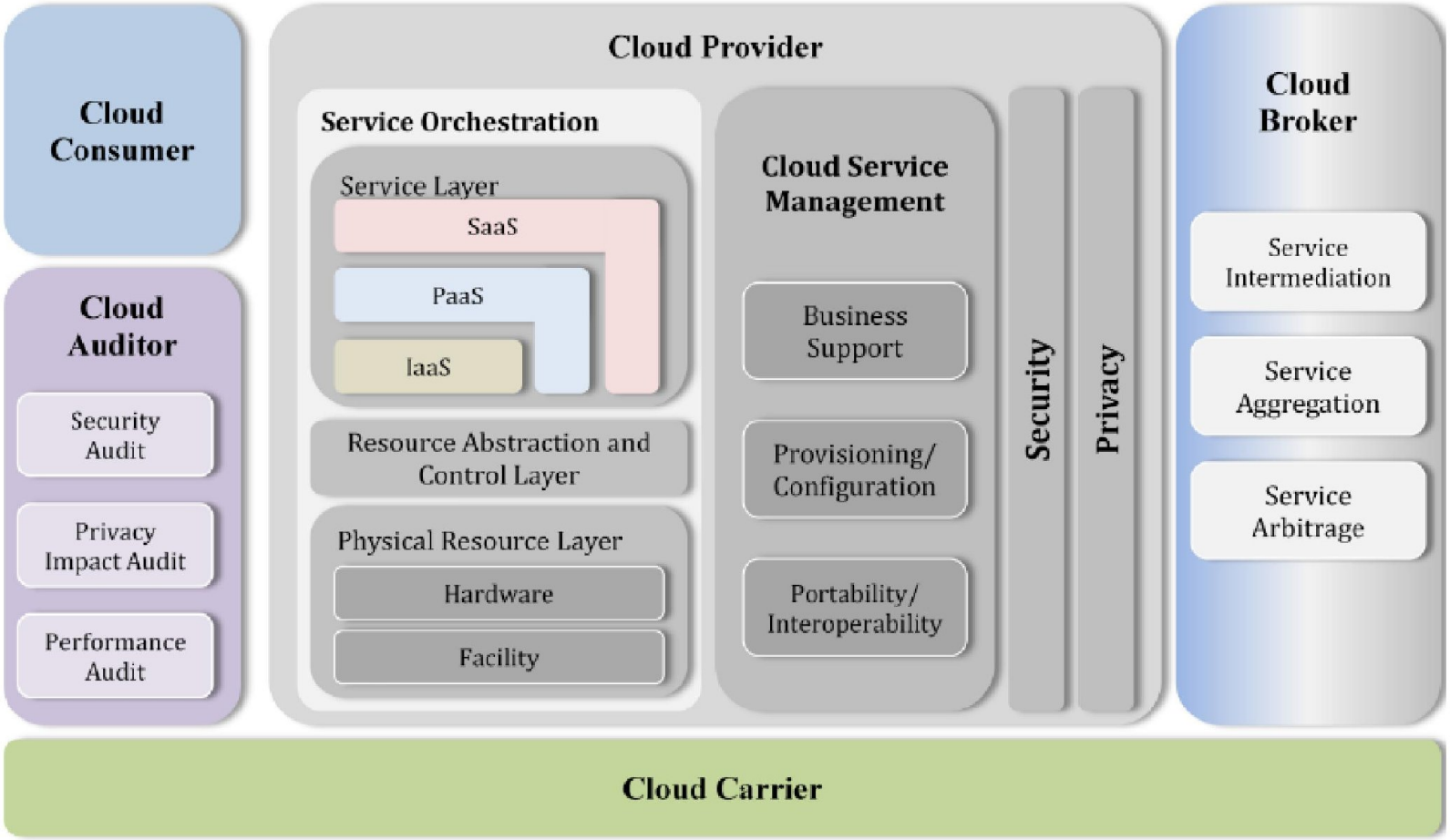
Cloud Carrier

- Acts as an intermediary that provides connectivity and transport of cloud services between cloud consumers and cloud providers
- Provides access to consumers through a public/private network or telecom provider
- A provider may set-up SLAs with cloud carriers in order to provide services with the level of SLAs offered to consumers (e.g. may require dedicated or secure connections)

Architectural Components

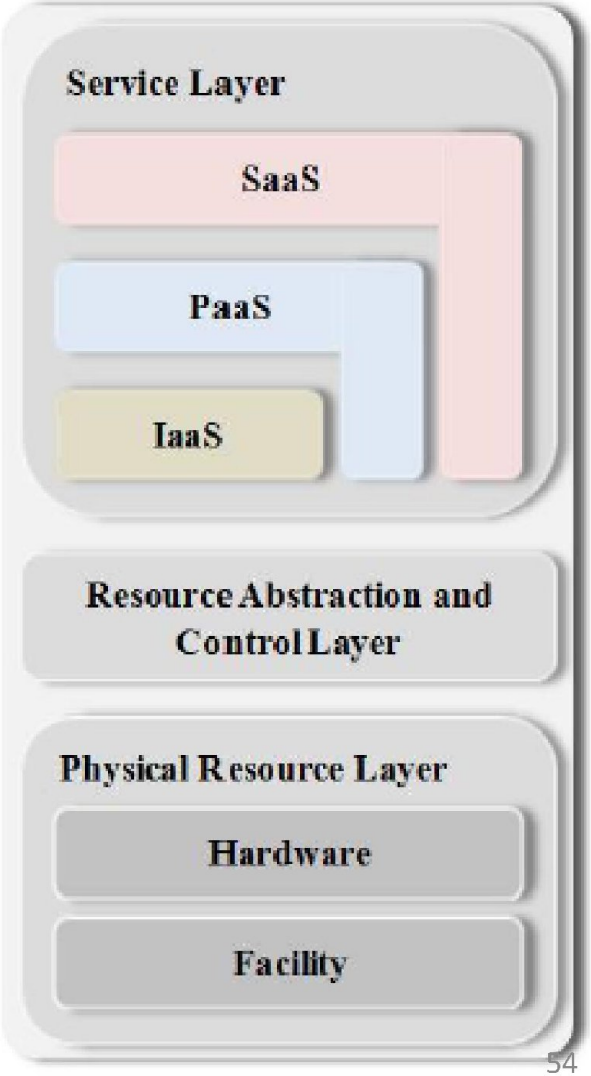
- Architectural Components for managing and providing cloud services, describe the important aspects of
  - Service deployment, orchestration, management, security and privacy
  - Portability and interoperability issues for data and services are also crucial factors as consumers need confidence and moving data and services across clouds
  - Security and privacy build trust and acceptance in clouds ability to provide a trustworthy and reliable system
  - Business support: implementation of specific business model

Conceptual Reference Model



Service Orchestration

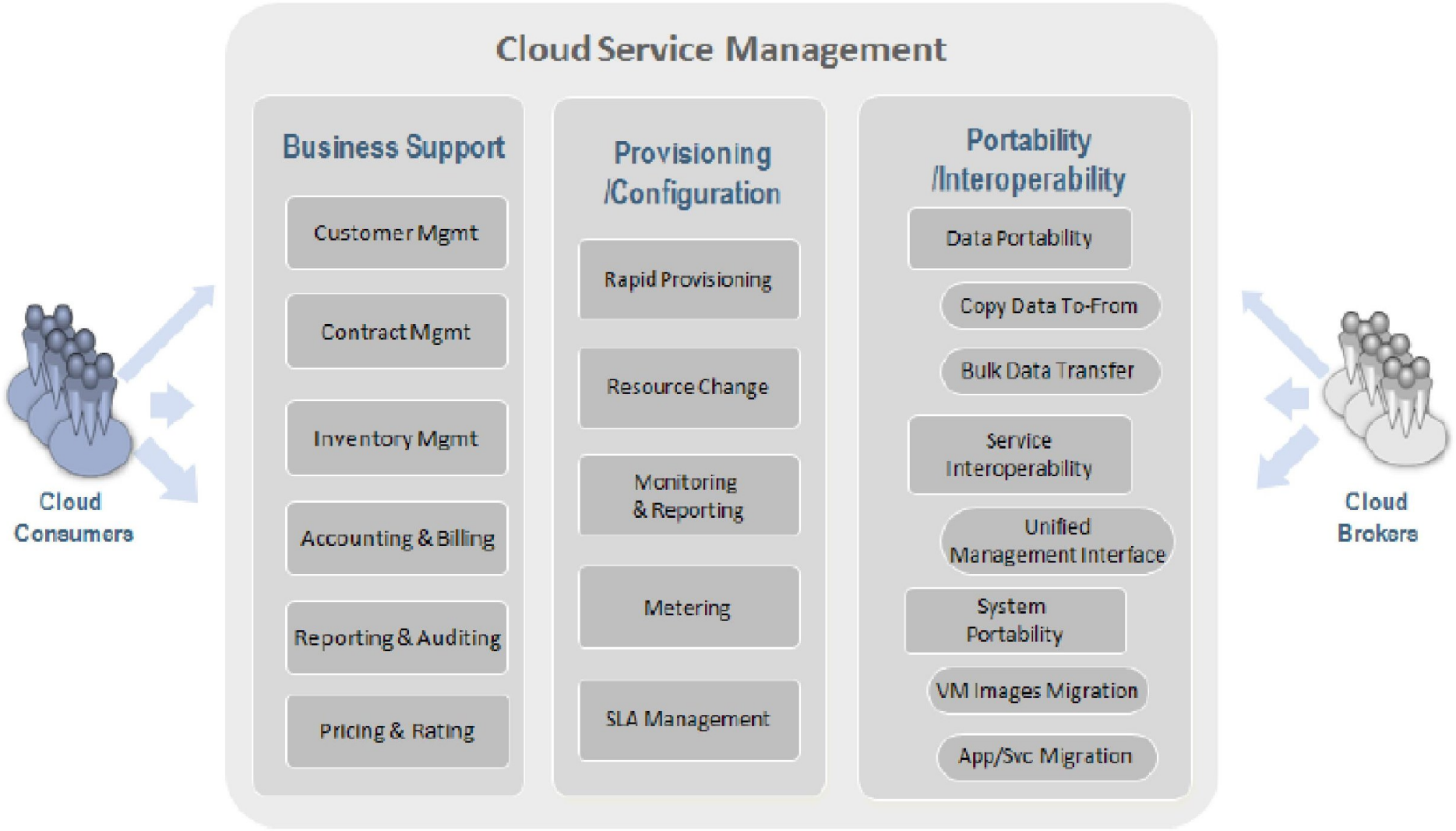
- Composition of service components to support cloud providers activities (in coordination with management of resources) in order to provide cloud services
- Service Layer: interfaces for accessing services (typically for IaaS, PaaS, SaaS)
- Resource Abstraction / Control Layer: interfaces for accessing virtualized resources e.g. hypervisors, VMs, virtual storage
- Physical Resource Layer: interfaces for accessing to physical resources (computers, disks, routers, firewalls, etc.)



Service Management

- Includes all of service-related functions that are necessary for the management and operation of services available to consumers
- Can be described from different perspectives
  - Business support
  - Provisioning and configuration
  - Portability and interoperability

Cloud Service Management



Management: Business Support

- Business related services
  - Customer management: manage customer accounts, open/close accounts, manage user profiles, manage provider-customer relationships
  - Contract management: setup/negotiate/terminated contract and SLAs
  - Pricing/Rating: evaluate cloud services, handle promotions and pricing rules by user profile
  - Accounting and Billing: collect billing information, send billing statements, manage payments
  - Reporting/auditing: monitor user operations, generate reports

Management: Provisioning/Configuration

- Responsibilities included
  - Rapid provisioning: automatically deploy cloud services based on user demands
  - Resource changing: adjust service configurations or, resource assignment for repairs/upgrades
  - Metering: Provide metering capability per service type
  - SLA management: define SLAs, monitor SLAs, enforce SLAs

Management: Portability/Interoperability

- Cloud adoption depends also how the cloud can address security, privacy, portability and interoperability concerns
- Portability: ability to move applications and data across clouds and cloud providers
  - Data portability: copy/move objects across clouds
  - System portability: move / migrate a stopped VMs or applications with their contents
  - Service Interoperability: use data and services across multiple cloud providers using common interface (RESTful APIs)
- Different requirements for different service models: IaaS, SaaS focus on data portability, IaaS, PaaS on compatibilities between different virtualization technologies, PaaS focus also on service interoperability

Privacy

- Ensure privacy of collected personal identifiable information that can be used to distinguish, trace user's identity based on
  - user habits (e.g. Buying patterns)
  - personal data: user id's, financial, health data, usage data
  - Also related to data security as application data encompass user related information
- Mainly a responsibility of cloud providers

Security

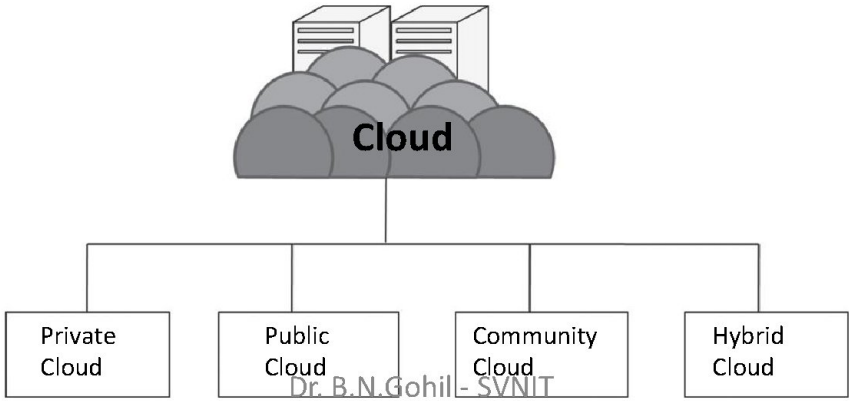
- Cloud systems need to address security requirements such as authentication, authorization, confidentiality, identity management, security monitoring, security policy management, incident response
- Responsibility shared between provider and consumer
- Consider impacts per service model:
  - SaaS: manage accessibility of cloud offerings using network connection and through Web browser (Web browser security is an issue)
  - IaaS: hypervisor security for VM isolation
  - PaaS: user authorization to use services
- Impacts per deployment model: private cloud is dedicated to one customer, public is not

Cloud Deployment Models

Importance of deployment models

- The deployment models are the different ways in which the cloud computing environment can be set up.
- Cloud computing is business oriented, and the popularity of the cloud is credited to its market-oriented nature.
- In the business perspective, making the correct decision regarding the deployment model is very important.
- A model should be selected based on the needs, requirements, budget, and security.
- A wrong decision in the deployment model may affect the organization heavily.
- There are many users of the cloud, and each user has different needs. One deployment model will not suite all the cloud users. Based on the cloud setup, the properties of the cloud change.
- There are four types of deployment models available in the cloud, namely, **private, public, community,** and **hybrid**.

- The classification of the cloud is based on several parameters such as the size of the cloud (number of resources), type of service provider, location, type of users, security, and other issues.
- The smallest in size is the private cloud



Private Cloud

- According to the National Institute of Standards and Technology (NIST), private cloud can be defined as the cloud infrastructure that is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units).
- It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.
- Private cloud can be deployed using open source tools such as Openstack, Eucalyptus.
- The private cloud is small in size as compared to other cloud models. Here, the cloud is deployed and maintained by the organizations itself.

Private cloud - Characteristics

- Secure: This is because usually the private cloud is deployed and managed by the organization itself, and hence there is least chance of data being leaked out of the cloud. There is no other risk from anybody else as all the users belong to the same organization.
- Central control: When it is managed by the organization itself, there is no need for the organization to rely on anybody.
- Weak SLAs: Formal SLAs may or may not exist in a private cloud. But if they exist they are weak as it is between the organization and the users of the same organization.

Private cloud - Suitability

- Suitability refers to the instances where this cloud model can be used. It also signifies the most suitable conditions and environment where this cloud model can be used, such as the following:

- The organizations or enterprises that require a separate cloud for their personal or official use.
- The organizations or enterprises that have a sufficient amount of funds as managing and maintaining a cloud is a costly affair.
- The organizations or enterprises that consider data security to be important.
- The organizations that want autonomy and complete control over the cloud.
- The organizations that have a less number of users.
- The organizations that have prebuilt infrastructure for deploying the cloud and are ready for timely maintenance of the cloud for efficient functioning.
- Special care needs to be taken and resources should be available for troubleshooting.

- The private cloud platform is not suitable for the following:
  - The organizations that have high user base
  - The organizations that have financial constraints
  - The organizations that do not have prebuilt infrastructure
  - The organizations that do not have sufficient manpower to maintain and manage the cloud

- According to NIST, the private cloud can be classified into several types based on their location and management:
- On-premise private cloud
- Outsourced private cloud

- On-Premise Private Cloud - A typical private cloud that is managed by a single organization. Here, the cloud is deployed in organizational premises and is connected to the organizational network.
- Issues
- SLA
- Network
- Performance
- Security and data privacy
- Location
- Cloud management
- Multi-tenancy
- Maintenance

- Outsourced Private Cloud – It is a cloud outsourced to a third party. A third party manages the whole cloud. Everything is same as usual private cloud except that here the cloud is outsourced.
- There are several advantages and disadvantages of outsourcing the cloud.
- Issues

- SLA
- Network
- Security and Privacy
- Laws and conflicts
- Location
- Performance
- Maintenance

Private cloud

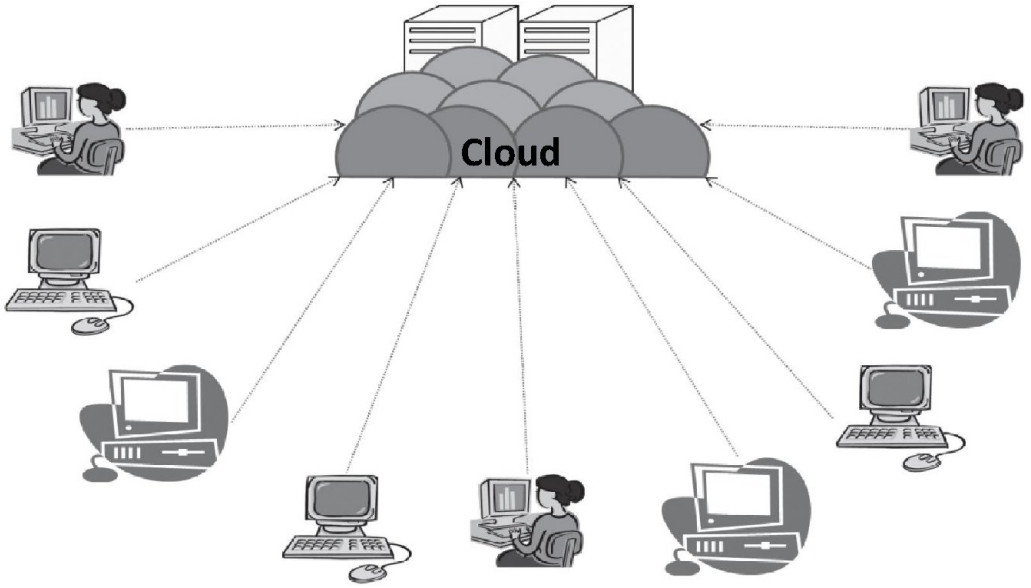
- Advantages
- The cloud is small in size and is easy to maintain.
- It provides a high level of security and privacy to the user.
- It is controlled by the organization.
- Disadvantages
- For the private cloud, budget is a constraint.
- The private clouds have loose SLAs.

Public cloud

- According to NIST, the public cloud is the cloud infrastructure that is provisioned for open use by the general public.
- It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
- Public cloud consists of users from all over the world.
- A user can simply purchase resources on an hourly basis and work with the resources.
- There is no need of any prebuilt infrastructure for using the public cloud. These resources are available in the cloud provider's premises.
- Usually, cloud providers accept all the requests, and hence, the resources in the service providers' end are considered *infinite in one aspect. Some of the well-known examples of the public cloud are Amazon AWS, Microsoft Azure, etc.*

Public cloud - characteristics

- Highly scalable
- Affordable
- Less secure
- Highly available
- Stringent SLAs



Public cloud - Suitability

- The requirement for resources is large, that is, there is large user base.
- The requirement for resources is varying.
- There is no physical infrastructure available.
- An organization has financial constraints.

The public cloud is not suitable for the following where:

- Security is very important.
- Organization expects autonomy.
- Third-party reliability is not preferred.

Public cloud - Issues

- SLA
- Network
- Performance
- Multi-tenancy
- Location
- Security and data privacy
- Laws and conflicts
- Cloud management
- Maintenance

Public cloud

- Advantages
  - There is no need of establishing infrastructure for setting up a cloud.
  - There is no need for maintaining the cloud.
  - They are comparatively less costly than other cloud models.
  - Strict SLAs are followed.
  - There is no limit for the number of users.
  - The public cloud is highly scalable.

- Disadvantages
  - Security is an issue.
  - Privacy and organizational autonomy are not possible.

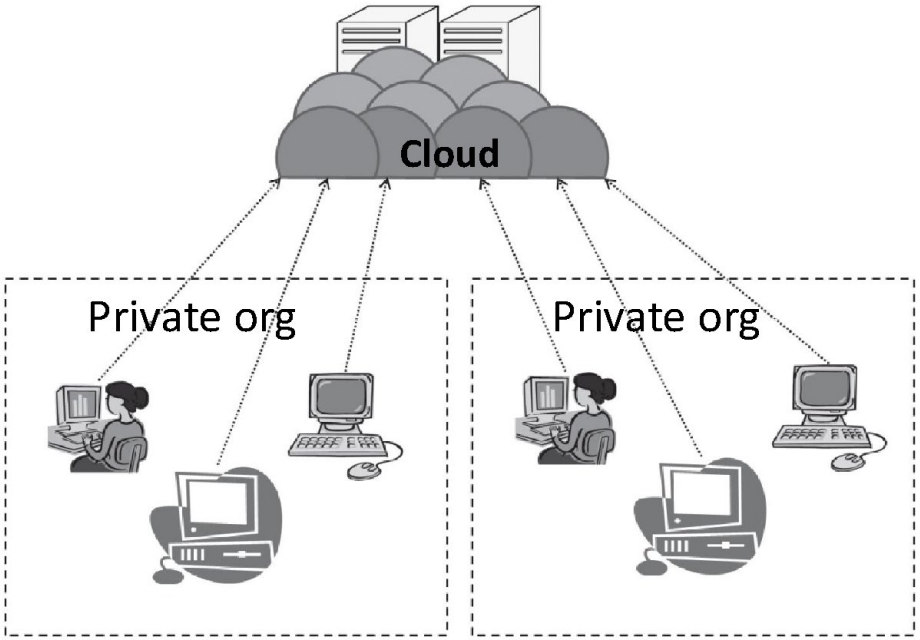
Community cloud

- According to NIST, the community cloud is the cloud infrastructure that is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations).
- It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.
- It is a further extension of the private cloud. Here, a private cloud is shared between several organizations.
- Either the organizations or a single organization may collectively maintain the cloud.

- The main advantage of the community cloud is that the organizations are able to share the resources among themselves based on specific concerns.
- Thus, here the organizations are able to extract the power of the cloud, which is much bigger than the private cloud, and at the same time, they are able to use it at a usually less cost.
- The community is formed based on any common cause, but eventually, all the members of the community are benefitted.
- This model is very suitable for organizations that cannot afford a private cloud and cannot rely on the public cloud either.

Community cloud - characteristics

- Collaborative and distributive maintenance
- Partially secure
- Cost effective



Community cloud - Suitability

- This kind of cloud is suitable for organizations that
  - Want to establish a private cloud but have financial constraint
  - Do not want to complete maintenance responsibility of the cloud
  - Want to establish the cloud in order to collaborate with other clouds
  - Want to have a collaborative cloud with more security features than the public cloud

- This cloud is not suitable for organizations that
  - Prefer autonomy and control over the cloud
  - Does not want to collaborate with other organizations

- There are two types of community cloud deployments:

1. On-premise community cloud
2. Outsourced community cloud

On- premise community cloud - issues

- SLA
- Network
- Performance
- Multi-tenancy
- Location
- Security and data privacy
- Laws and conflicts
- Cloud management
- Maintenance

Outsourced community cloud - issues

- SLA
- Network
- Performance
- Security and data privacy
- Laws and conflicts
- Cloud management
- Maintenance

Community cloud

- Advantages
- It allows establishing a low-cost private cloud.
- It allows collaborative work on the cloud.
- It allows sharing of responsibilities among the organization.
- It has better security than the public cloud.
- Disadvantages
- Autonomy of an organization is lost.
- Security features are not as good as the private cloud.
- It is not suitable if there is no collaboration.

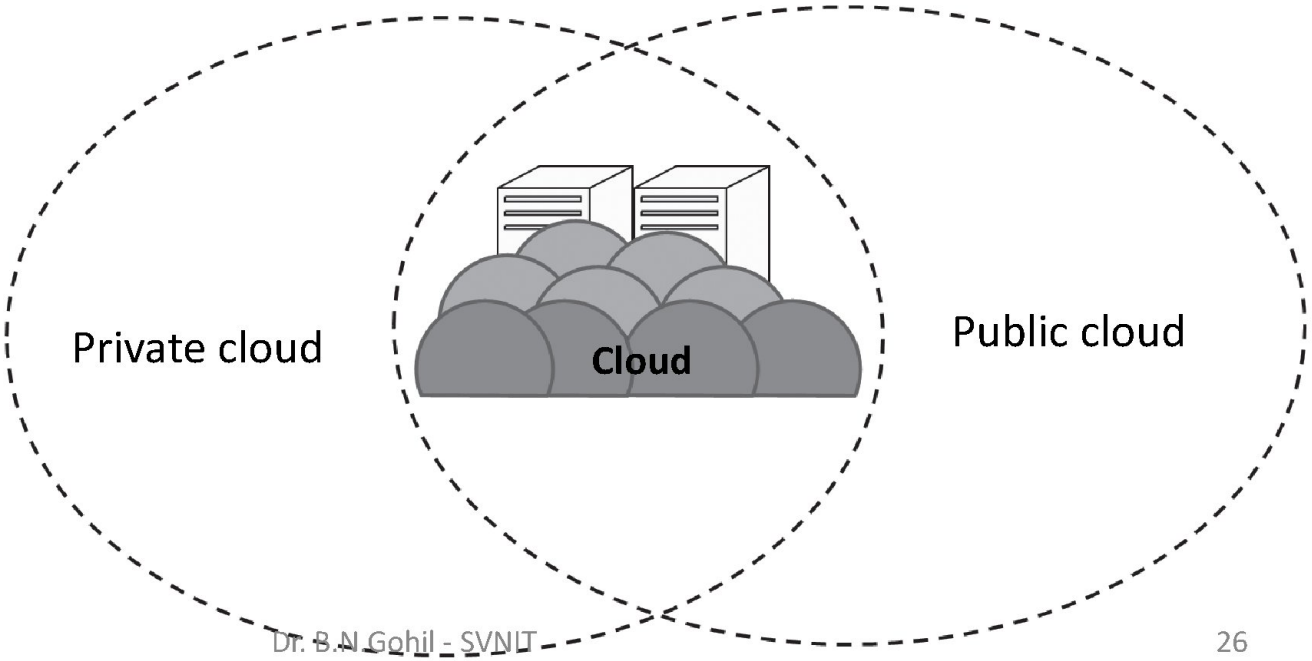
Hybrid Cloud

- According to NIST, the hybrid cloud can be defined as the cloud infrastructure that is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability.
- The hybrid cloud usually is a combination of both public and private clouds. This is aimed at combining the advantages of private and public clouds.
- The usual method of using the hybrid cloud is to have a private cloud initially, and then for additional resources, the public cloud is used.

- There are several advantages of the hybrid cloud.
- The hybrid cloud can be regarded as a private cloud extended to the public cloud. This aims at utilizing the power of the public cloud by retaining the properties of the private cloud.
- One of the popular examples for the hybrid cloud is Eucalyptus.
- Eucalyptus was initially designed for the private cloud and is basically a private cloud, but now it also supports hybrid cloud.

Hybrid cloud - characteristics

- Scalable
- Partially secure
- Stringent SLAs
- Complex cloud management



Hybrid cloud - Suitability

- The hybrid cloud environment is suitable for

- Organizations that want the private cloud environment with the scalability of the public cloud
- Organizations that require more security than the public cloud

- The hybrid cloud is not suitable for

- Organizations that consider security as a prime objective
- Organizations that will not be able to handle hybrid cloud management

Hybrid cloud - issues

- SLA
- Network
- Performance
- Multi-tenancy
- Location
- Security and data privacy
- Laws and conflicts
- Cloud management
- Maintenance

Hybrid cloud

- Advantages
- It gives the power of both the private and public clouds.
- It is highly scalable.
- It provides better security than the public cloud.
- Disadvantages
- The security features are not as good as the public cloud.
- Managing a hybrid cloud is complex.
- It has stringent SLAs.

Technological Drivers of CC

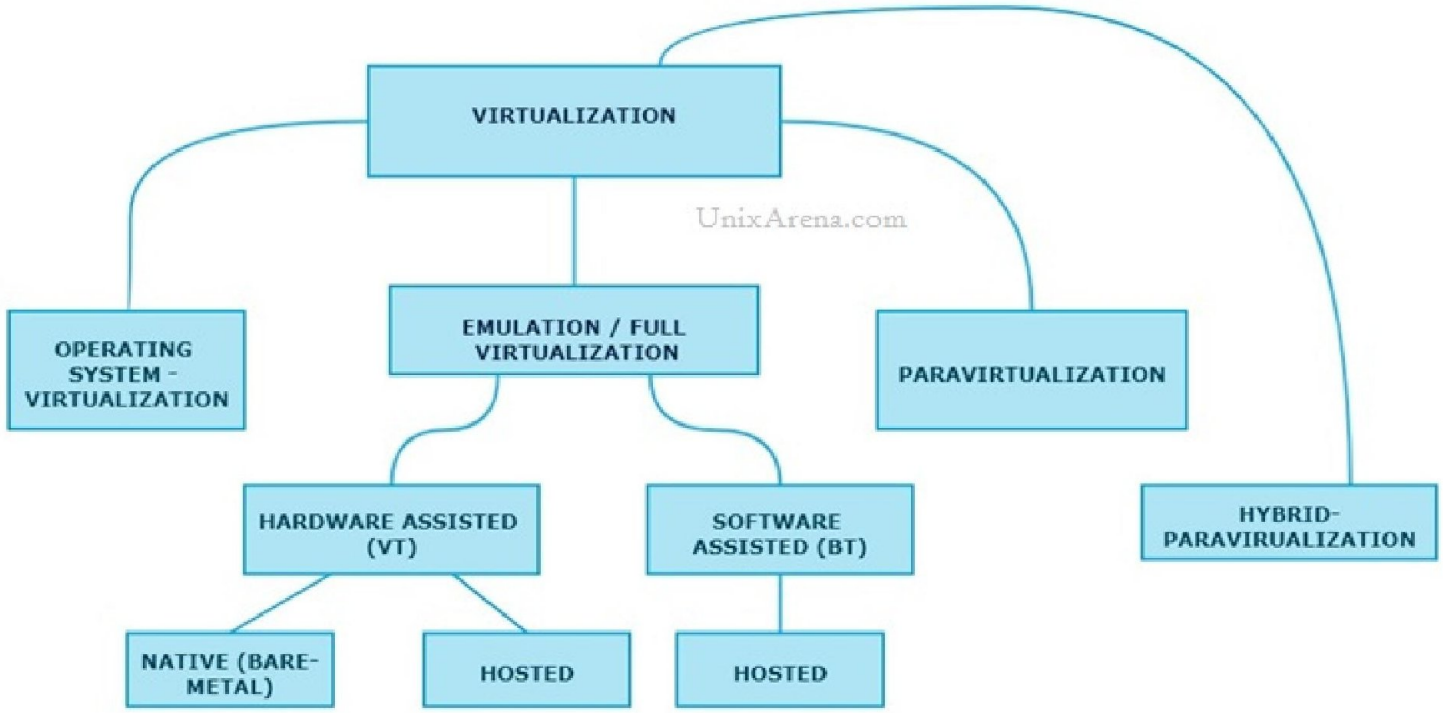
Virtualization

- Virtualization is nothing but abstracting operating system, application, storage or network away from the true underlying hardware or software.
- It creates the illusion of physical hardware to achieve the goal of operating system isolation.
- In last decade, data centers were occupied by a large number of physical servers, network switches, storage devices. It consumed a lot of power and manpower to maintain the data centers.

Cont..

- It is the underlying core technology of cloud computing.
- It helps in creating a multitenant model for the cloud environment by optimizing the resource usage through sharing.
- Benefits of virtualization include the lower costs and extended life of the technology, which has made it a popular option with small- to medium-sized businesses.
- Using virtualization, the physical infrastructure owned by the service provider is **shared among many users**, increasing the **resource utilization**. Virtualization provides efficient resource utilization and increased return on investment (ROI). Ultimately, it results in low capital expenditures (CapEx) and operational expenditures (OpEx).

Types of Virtualization



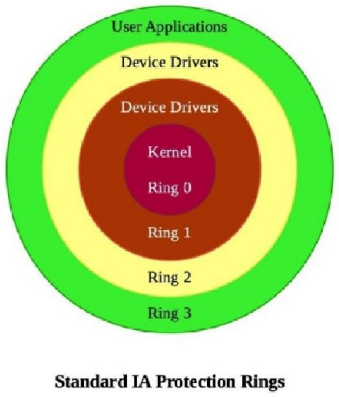
- **Full Virtualization:**
- Virtual machine simulates hardware to allow an unmodified guest OS to be run in isolation.
- Full virtualization uses a special kind of software called a hypervisor.
- The hypervisor interacts directly with the physical server's hardware resources, such as the CPU and storage space, and acts as a platform for the virtual server's OSs. It helps to keep each virtual server completely independent and unaware of the other virtual servers running on the physical machine.
- Each guest server or the virtual machine (VM) is able to run its own OS. That means one virtual server could be running on Linux and the other one could be running on Windows.

- There is two type of Full virtualizations in the enterprise market. On both full virtualization types, guest operating system's source information will not be modified.
  - Software assisted full virtualization
  - Hardware-assisted full virtualization

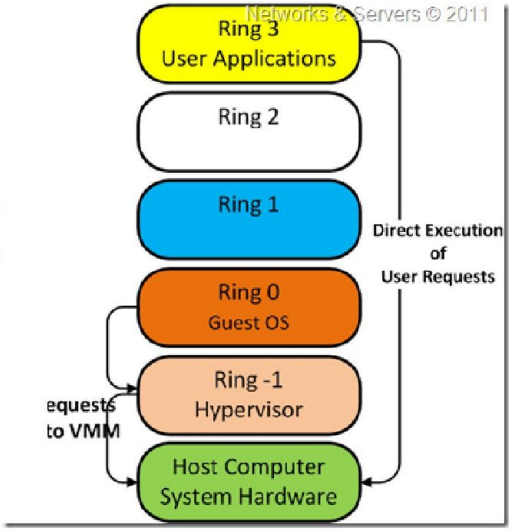
- Examples include VMWare ESX and VirtualBox.
- Advantages - isolation among the various VMs, isolation between the VMs and the hypervisor, concurrent execution of multiple OSs, and no change required in the guest OS.
- Disadvantage - the overall system performance may be affected due to binary translation.

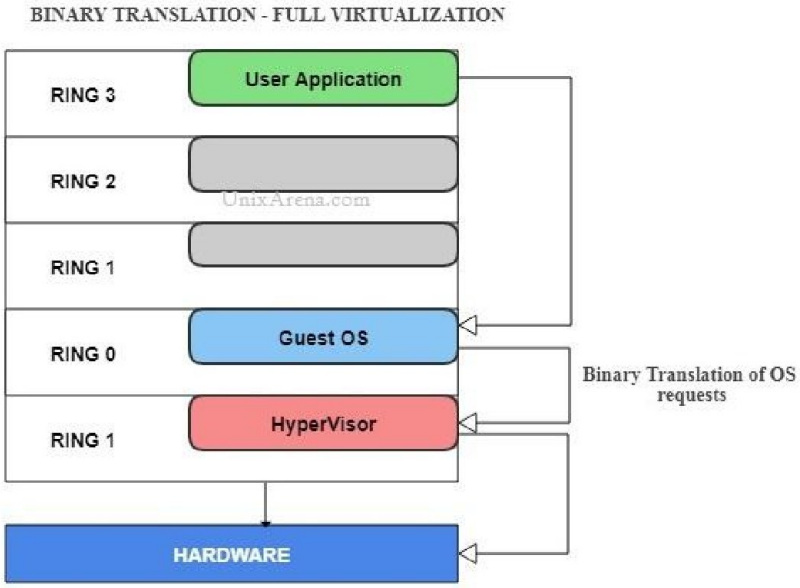
Software Assisted FV

- It completely relies on binary translation to trap and virtualize the execution of sensitive, non-virtualizable instructions sets.
- It emulates the hardware using the software instruction sets.
- Due to binary translation, it often criticized for performance issue.
- Example: VMware workstation, Virtual PC, VirtualBox, VMware Server



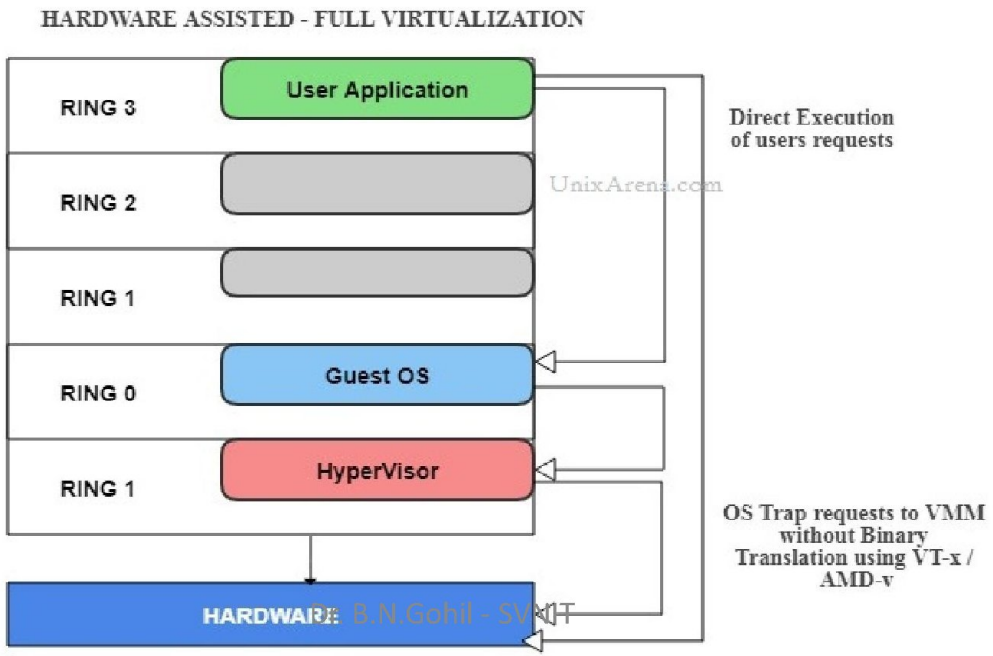
Standard IA Protection Rings





Hardware Assisted FV

- Hardware-assisted full virtualization eliminates the binary translation and it directly interrupts with hardware using the virtualization technology which has been integrated on X86 processors since 2005 (Intel VT-x and AMD-V).
- Guest OS's instructions might allow a virtual context execute privileged instructions directly on the processor, even though it is virtualized.
- Example(bare metal – type 1): Vmware ESXi/ESX, KVM, Hyper-V, Xen
- Example(hosted – type 2): Vmware, Vrtual Box

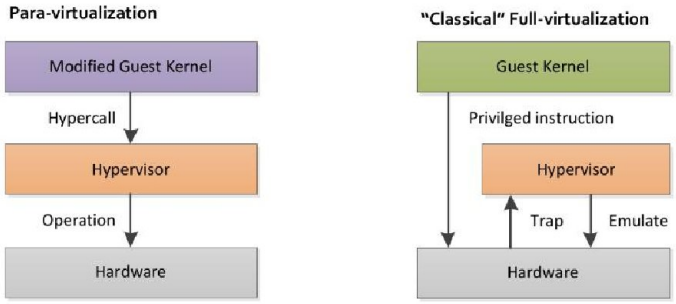


Para-virtualization

- It doesn't need to simulate the hardware for the virtual machines.
- The hypervisor is installed on a physical server (host) and a guest OS is installed into the environment.
- Virtual guests aware that it has been virtualized, unlike the full virtualization (where the guest doesn't know that it has been virtualized) to take advantage of the functions.
- In this virtualization method, guest source codes will be modified with sensitive information to communicate with the host.
- Guest Operating systems require extensions to make API calls to the hypervisor.
- In full virtualization, guests will issue a hardware calls but in paravirtualization, guests will directly communicate with the host (hypervisor) using the drivers.

Cont..

- VMs do not simulate the underlying hardware, but with the modified guest OS, API is used.
- Example – Xen, VMWare ESX server, IBM LPAR, Oracle VM
- Here, the guest OS is aware of the fact that it is running in a virtualized environment.
- Hypercalls are used for the direct communication between the guest OS and the hypervisor.
- Advantage - it improves the overall system performance by eliminating the overhead of binary translation.
- Disadvantage - modification of the guest OS is required.

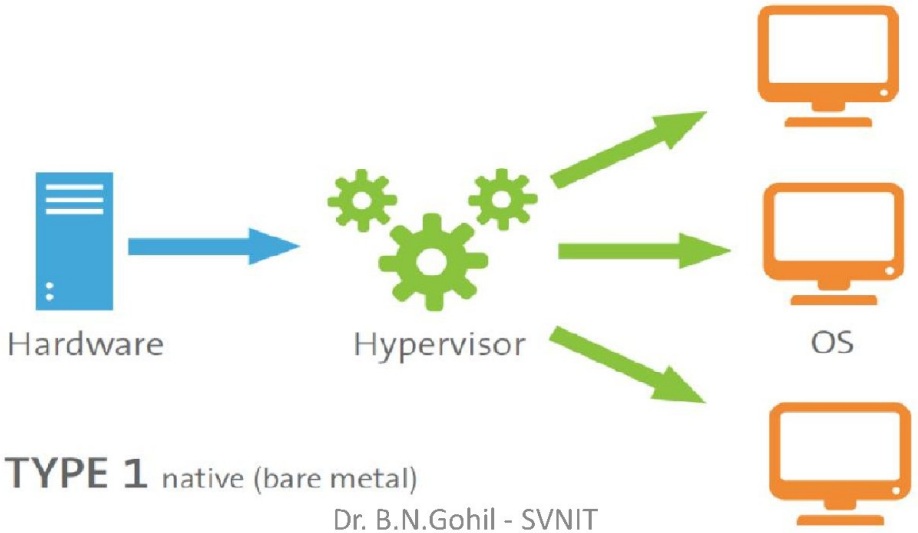


Hypervisor/VMM

- Hypervisors are software tools used to create the VMs, and they produce the virtualization of various hardware resources such as CPU, storage, and networking devices.
- They are also called virtual machine monitor (VMM) or virtualization managers.
- Example - VMware, Xen, Hyper-V, KVM, etc.
- Hypervisors help to run multiple OSs concurrently on a physical system **sharing its hardware**. Thus, **a hypervisor allows multiple OSs to share a single hardware host**.
- The hypervisor also makes sure that the guest OSs (called VMs) do not interrupt each other. It manages multiple OSs or multiple instances of the same OS on a single physical computer system.

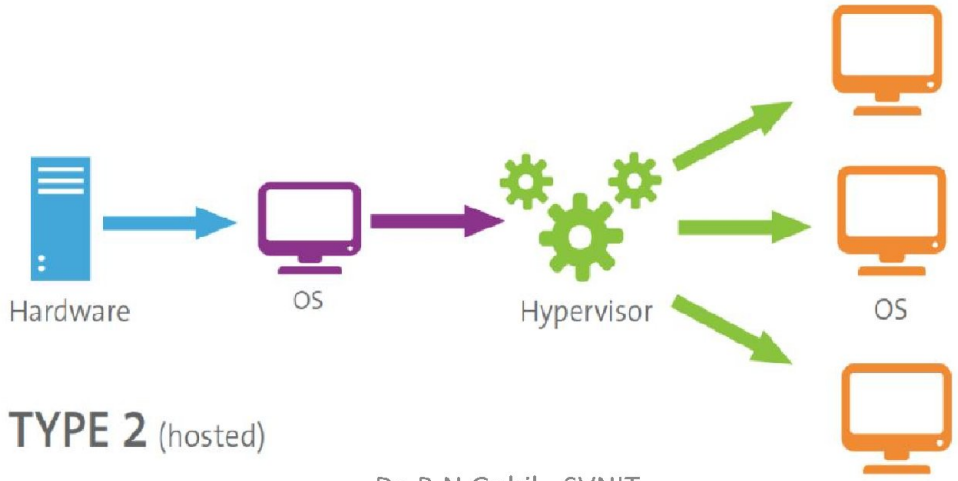
Cont..

- Type 1 hypervisor:
- This type of hypervisor runs directly on the host computer's hardware in order to control the hardware resources and also to manage the guest OSs.
- This is also known as native or bare-metal hypervisors.
- Examples - VMware ESXi, Citrix XenServer, and Microsoft Hyper-V hypervisor.

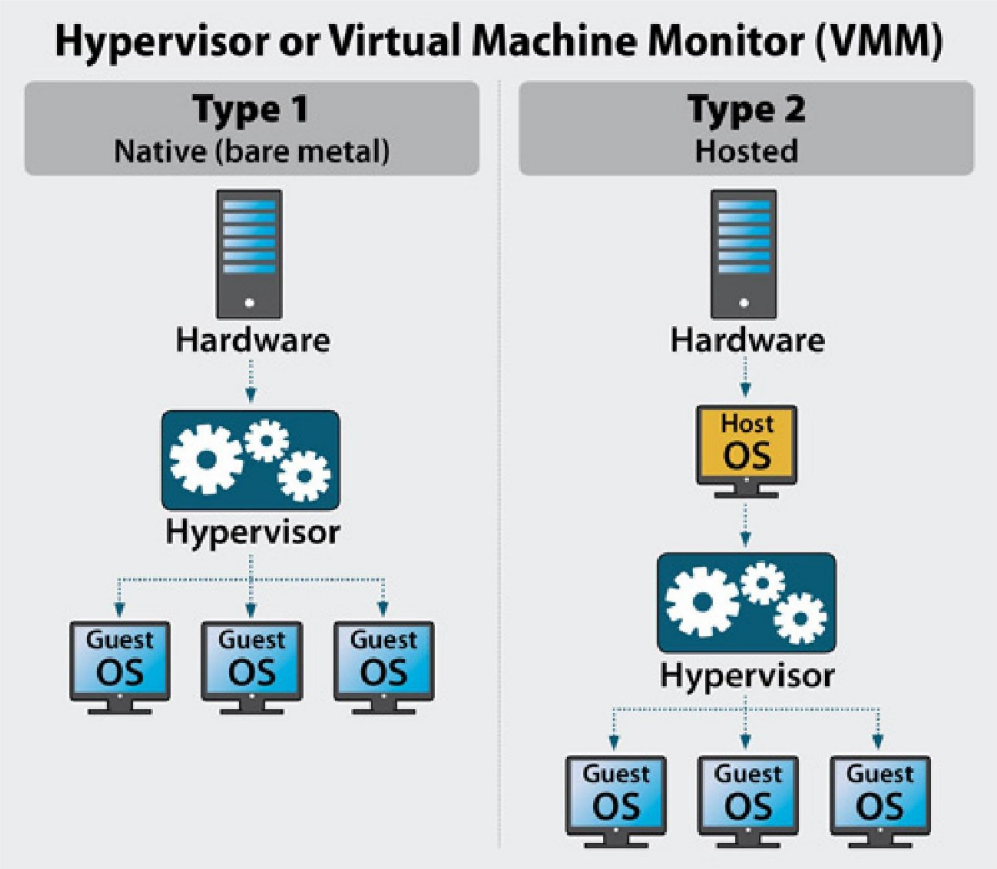


Cont..

- Type 2 hypervisor:
- This type of hypervisor runs within a formal OS environment as a distinct second layer while the guest OS runs as a third layer above the hardware.
- This is also known as the hosted hypervisors.
- Examples - VMware Workstation and VirtualBox



Type 1 Vs Type 2



Types of Virtualization

- Depending on the resources virtualized, the process of virtualization can be classified into the following types.

- OS virtualization/Containerization - The kernel of an operating system allows more than one isolated user-space instance to exist.
- Hardware/Server virtualization – creates VMs in single server.
- Memory virtualization -decouples memory from the server to provide a shared, distributed or networked function.
- Storage virtualization - multiple network storage resources are present as a single storage device for easier and more efficient management of these resources
- Network virtualization - process of combining hardware and software network resources and network functionality into a single, software-based administrative entity.
- Application virtualization - encapsulates computer programs from the underlying operating system on which they are executed.

Multicore Technology

- Two or more CPUs are working together on the same chip.
- In this type of architecture, a single physical processor contains the core logic of two or more processors. These processors are packaged into a single integrated circuit (IC).

Memory and Storage Technologies

The storage technology or solutions used in the cloud environment should meet the following requirements.

- Scalability
- High availability
- Constant performance
- High bandwidth
- Load balancing

Networking Technologies

Network requirements for cloud:

- Consolidate workloads and provide Infrastructure as a Service (IaaS)to various tenants
- Provide VM connectivity to physical and virtual networks
- Ensure connectivity and manage network bandwidth
- Speed application and server performance

Web 2.0

- It is the popular term given to the advanced Internet technology and applications that include blogs, wikis, really simple syndication

  (RSS), and social bookmarking.
- Characteristics:

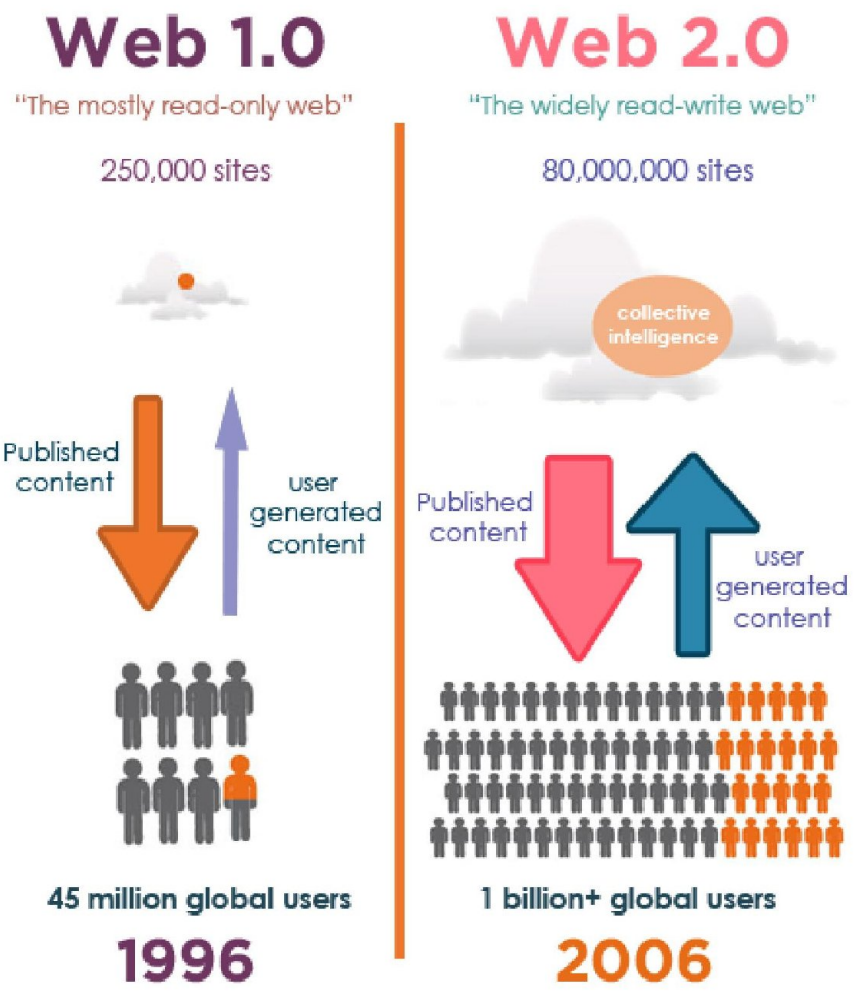
- instead of merely reading the contents from a web page, a user is allowed to write or contribute to the content available to everyone in an effective and user-friendly manner

Cont..

- Web 2.0 is also called network as a platform computing as it provides software, computing, and storage facilities to the user all through the browser.
- The major applications of Web 2.0 include social networking sites, self-publishing platforms, tagging, and social bookmarking.

Cont..

- The key features:
- Folksonomy – digital tagging
- Rich user experience
- User as a contributor
- User participation
- Dispersion



Web 3.0

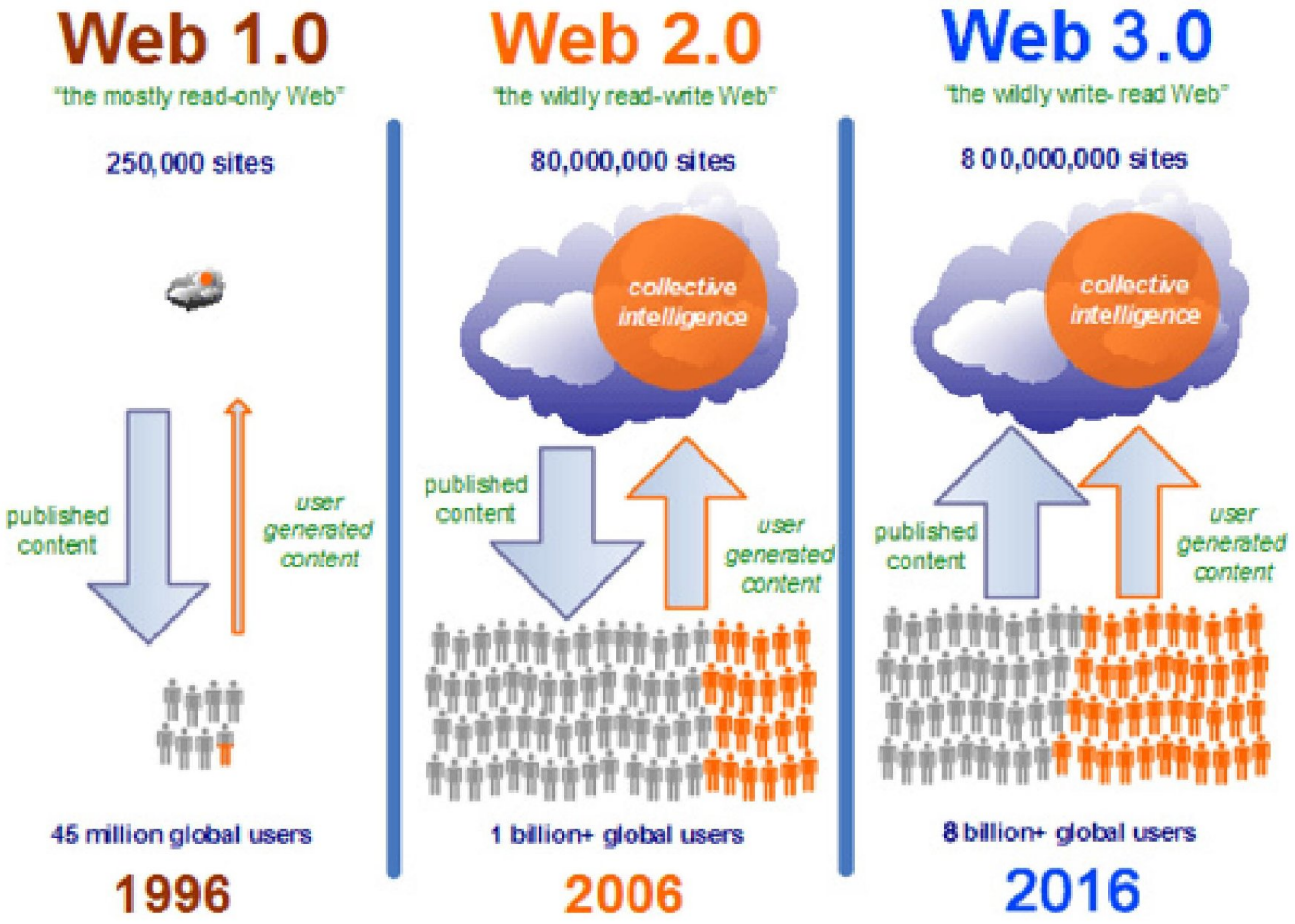
The two major components forming the basis of Web 3.0 are :

1. Semantic web - The semantic web provides the web user a common framework that could be used to share and reuse the data across various applications, enterprises, and community boundaries. The semantic web is a vision of IT that allows the data and information to be readily interpreted by machines, so that the machines are able to take contextual decisions on their own by finding, combining, and acting upon relevant information on the web.
2. Web services - A web service is a software system that supports computer-to-computer interaction over the Internet. Web services are usually represented as APIs.

Characteristics of Web 3.0:

- Ubiquitous connectivity
- Network computing
- Open technologies
- Open identity
- The intelligent web

Comparison


Web 1.0
"the mostly read-only Web"
250,000 sites
published content
user generated content
45 million global users
1996
Web 2.0
"the wildly read-write Web"
80,000,000 sites
collective intelligence
published content
user generated content
1 billion+ global users
2006
Web 3.0
"the wildly write- read Web"
8 00,000,000 sites
collective intelligence
published content
user generated content
8 billion+ global users
2016

Process models for cloud

- Waterfall
- V model
- Incremental
- RAD
- Agile
- Iterative
- Spiral

Agile SDLC for Cloud

- Existing software process models and framework activities are not adequate unless interaction with cloud providers is included.
- Requirements gathering phase so far included customers, users, and software engineers.
- Now, it has to include the cloud providers as well, as they will be providing the computing infrastructure and its maintenance.
- As only the cloud providers will know the size, architectural details, virtualization strategy, and resource utilization of the infrastructure, they should also be included in the planning and design phases of software development.
- Coding and testing can be done on the cloud platform, which is a huge benefit as everybody will have easy access to the software being built.
- This will reduce the cost and time for testing and validation.

Cont..

- In the cloud environment, software developers can use the web services and open-source software freely available from the cloud instead of procuring them.
- Software developers build software from readily available components rather than writing it all and building a monolithic application.
- Refactoring of existing application is required to best utilize the cloud infrastructure architecture in a cost-effective way.
- In the latest hardware technology, the computers are multicore and networked and the software engineers should train themselves in parallel and distributed computing to complement these advances of hardware and network technology.
- Cloud providers will insist that software should be as modular as possible for occasional migration from one server to another for LB as required by the cloud provider.

Cont..

- Most widely used agile frameworks:
- Agile Scrum Methodology
- Lean Software Development
- Kanban
- Extreme Programming (XP)
- Crystal
- Dynamic Systems Development Method (DSDM)
- Feature Driven Development (FDD)

Features of Cloud SDLC

- SDLC for cloud computing is different from the traditional SDLC in the following ways:

1. Inclination toward agile methodologies
2. Customizable SDLC framework for different stages: Cloud computing SDLC must have the capabilities to be customized according to the requirements of the project. In other words, the elasticity and robustness of cloud computing environment can be best utilized if the SDLCs for cloud are customizable.
3. Installation and configuration guidelines: SDLC for cloud must provide implementation approach and guidelines for installation and configuration of the cloud depending on its size. The guidelines must ensure that installation and configuration of infrastructure and application environment are completed appropriately for different stages of SDLC including operations and maintenance. These guidelines are the key to differentiating SDLC for cloud from traditional SDLC.

Advantages of Agile model

1. Faster time to market
2. Quick ROI
3. Shorter release cycles
4. Better quality
5. Better adaptability and responsiveness to business changing requirements
6. Early detection of failure/failing projects

CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services

Article · April 2009

Source: arXiv

CITATIONS
378

READS
1,423

4 authors, including:

R. Ranjan
Newcastle University
380 PUBLICATIONS **14,463** CITATIONS

SEE PROFILE

Rajkumar Buyya
University of Melbourne
970 PUBLICATIONS **74,899** CITATIONS

SEE PROFILE

CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services

Rodrigo N. Calheiros[1,2], Rajiv Ranjan[1], César A. F. De Rose[2], and Rajkumar Buyya[1]

[1]**Gri**d Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia

[2]Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, Brazil
{rodrigoc, rranjan, raj}@csse.unimelb.edu.au, cesar.derose@pucrs.br

Abstract

Cloud computing focuses on delivery of reliable, secure, fault-tolerant, sustainable, and scalable infrastructures for hosting Internet-based application services. These applications have different composition, configuration, and deployment requirements. Quantifying the performance of scheduling and allocation policy on a Cloud infrastructure (hardware, software, services) for different application and service models under varying load, energy performance (power consumption, heat dissipation), and system size is an extremely challenging problem to tackle. To simplify this process, in this paper we propose CloudSim: a new generalized and extensible simulation framework that enables seamless modelling, simulation, and experimentation of emerging Cloud computing infrastructures and management services. The simulation framework has the following novel features: (i) support for modelling and instantiation of large scale Cloud computing infrastructure, including data centers on a single physical computing node and java virtual machine; (ii) a self-contained platform for modelling data centers, service brokers, scheduling, and allocations policies; (iii) availability of virtualization engine, which aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node; and (iv) flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

1. Introduction

Cloud computing delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. These services in industry are respectively referred to as Infrastructure as a Service (Iaas), Platform as a Service (PaaS), and Software as a Service (SaaS). In a Feb 2009 Berkeley Report [11], Prof. Patterson et. al. stated "Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service".

Clouds [10] aim to power the next generation data centers by architecting them as a network of virtual services (hardware, database, user-interface, application logic) so that users are able to access and deploy applications from anywhere in the world on demand at competitive costs depending on users QoS (Quality of Service) requirements [1]. Developers with innovative ideas for new Internet services are no longer required to make large capital outlays in the hardware and software infrastructures to deploy their services or human expense to operate it [11]. It offers significant benefit to IT companies by freeing them from the low level task of setting up basic hardware (servers) and software infrastructures and thus enabling more focus on innovation and creation of business values.

Some of the traditional and emerging Cloud-based applications include social networking, web hosting, content delivery, and real time instrumented data processing. Each of these application types has different composition, configuration, and deployment requirements. Quantifying the performance of scheduling and allocation policy on Cloud infrastructures (hardware, software, services) for different application and service models under varying load, energy performance (power consumption, heat dissipation), and system size is an extremely challenging problem to tackle. The use of real test beds such as Amazon EC2, limits the experiments to the scale of the testbed, and makes the reproduction of results an extremely difficult undertaking, as the conditions prevailing in the Internet-based environments are beyond the control of the tester.

An alternative is the utilization of simulations tools that open the possibility of evaluating the hypothesis prior to software development in an environment where one can reproduce tests. Specifically in the case of Cloud

computing, where access to the infrastructure incurs payments in real currency, simulation-based approaches offer significant benefits, as it allows Cloud customers to test their services in repeatable and controllable environment free of cost, and to tune the performance bottlenecks before deploying on real Clouds. At the provider side, simulation environments allow evaluation of different kinds of resource leasing scenarios under varying load and pricing distributions. Such studies could aid the providers in optimizing the resource access cost with focus on improving profits. In the absence of such simulation platforms, Cloud customers and providers have to rely either on theoretical and imprecise evaluations, or on try-and-error approaches that lead to inefficient service performance and revenue generation.

Considering that none of the current distributed system simulators [4][7][9] offer the environment that can be directly used by the Cloud computing community, in this paper, we propose CloudSim: a new, generalized, and extensible simulation framework that enables seamless modeling, simulation, and experimentation of emerging Cloud computing infrastructures and application services. By using CloudSim, researchers and industry-based developers can focus on specific system design issues that they want to investigate, without getting concerned about the low level details related to Cloud-based infrastructures and services.

CloudSim offers the following novel features: (i) support for modeling and simulation of large scale Cloud computing infrastructure, including data centers on a single physical computing node; and (ii) a self-contained platform for modeling data centers, service brokers, scheduling, and allocations policies. Among the unique features of CloudSim, there are: (i) availability of virtualization engine, which aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node; and (ii) flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services. These compelling features of CloudSim would speed up the development of new algorithms, methods, and protocols in Cloud computing, hence contributing towards quicker evolution of the paradigm.

2. Related Works

Cloud computing

Cloud computing can be defined as "a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers" [1]. Some examples of emerging Cloud computing infrastructures are Microsoft Azure [2], Amazon EC2, Google App Engine, and Aneka [3].

The computing power in a Cloud computing environments is supplied by a collection of data centers, which are typically installed with hundreds to thousands of servers [9]. The layered architecture of a typical Cloud-based data center is shown in Figure 1. At the lowest layers there exist massive physical resources (storage servers and application servers) that power the data centers. These servers are transparently managed by the higher level virtualization [8] services and toolkits that allow sharing of their capacity among virtual instances of servers. These virtual instances are isolated from each other, which aid in achieving fault tolerant behavior and isolated security context.

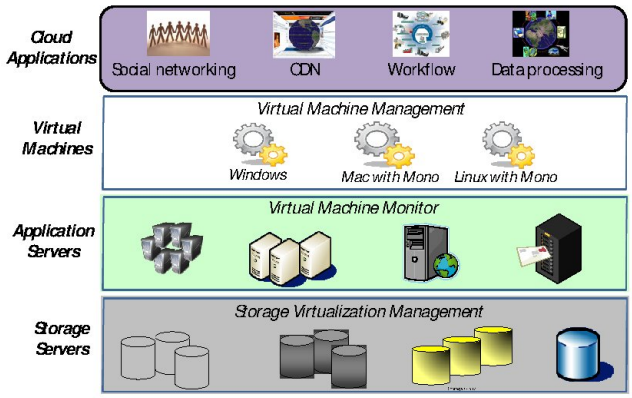


Figure 1. Typical data center.

Emerging Cloud applications such as Social networking, gaming portals, business applications, content delivery, and scientific workflows operate at the highest layer of the architecture. Actual usage patterns of many real-world applications vary with time, most of the time in unpredictable ways. These applications have different Quality of Service (QoS) requirements depending on time criticality and users' interaction patterns (online/offline).

Simulation

In the past decade, Grids [5] have evolved as the infrastructure for delivering high-performance service for compute and data-intensive scientific applications. To support research and development of new Grid components, policies, and middleware; several Grid simulators, such as GridSim [9], SimGrid [7], and GangSim [4] have been proposed. SimGrid is a generic framework for simulation of distributed applications in Grid platforms. GangSim is a Grid simulation toolkit that provides support for modeling of Grid-based virtual organisations and resources. On the other hand, GridSim is an event-driven simulation toolkit for heterogeneous Grid resources. It supports modeling of grid entities, users, machines, and network, including network traffic.

Although the aforementioned toolkits are capable of modeling and simulating the Grid application behaviors (execution, scheduling, allocation, and monitoring) in a distributed environment consisting of multiple Grid organisations, none of these are able to support the

infrastructure and application-level requirements arising from Cloud computing paradigm. In particular, there is very little or no support in existing Grid simulation toolkits for modeling of on-demand virtualization enabled resource and application management. Further, Clouds promise to deliver services on subscription-basis in a pay-as-you-go model to Cloud customers. Hence, Cloud infrastructure modeling and simulation toolkits must provide support for economic entities such as Cloud brokers and Cloud exchange for enabling real-time trading of services between customers and providers. Among the currently available simulators discussed in this paper, only GridSim offers support for economic-driven resource management and application scheduling simulation.

Another aspect related to Clouds that should be considered is that research and development in Cloud computing systems, applications and services are in the infancy stage. There are a number of important issues that need detailed investigation along the Cloud software stack. Topics of interest to Cloud developers include economic strategies for provisioning of virtualized resources to incoming user's requests, scheduling of applications, resources discovery, inter-cloud negotiations, and federation of clouds and so on. To support and accelerate the research related to Cloud computing systems, applications and services it is important that the necessary software tools are designed and developed to aid researchers and developers.

3. CloudSim Architecture

Figure 2 shows the layered implementation of the CloudSim software framework and architectural components. At the lowest layer is the SimJava discrete event simulation engine [6] that implements the core functionalities required for higher-level simulation frameworks such as queuing and processing of events, creation of system components (services, host, data center, broker, virtual machines), communication between components, and management of the simulation clock. Next follows the libraries implementing the GridSim toolkit [9] that support high level software components for modeling multiple Grid infrastructures, including networks and associated traffic profiles, and fundamental Grid components such as the resources, data sets, workload traces, and information services.

The CloudSim is implemented at the next level by programmatically extending the core functionalities exposed by the GridSim layer. CloudSim provides novel support for modeling and simulation of virtualized Cloud-based data center environments such as dedicated management interfaces for VMs, memory, storage, and bandwidth. CloudSim layer manages the instantiation and execution of core entities (VMs, hosts, data centers, application) during the simulation period. This layer is capable of concurrently instantiating and transparently managing a large scale Cloud infrastructure consisting of thousands of system components. The fundamental issues such as provisioning of hosts to VMs based on user requests, managing application execution, and dynamic monitoring are handled by this layer. A Cloud provider, who wants to study the efficacy of different policies in allocating its hosts, would need to implement his strategies at this layer by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer on how a host is allocated to different competing VMs in the Cloud. A Cloud host can be concurrently shared among a number of VMs that execute applications based on user-defined QoS specifications.

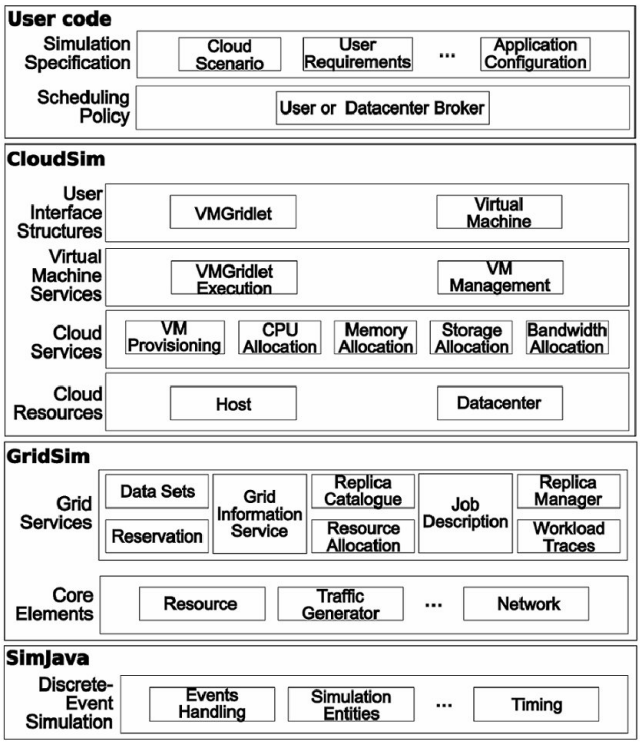


Figure 2. Layered CloudSim architecture.

The top-most layer in the simulation stack is the User Code that exposes configuration related functionalities for hosts (number of machines, their specification and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. A Cloud application developer can generate a mix of user request distributions, application configurations, and Cloud availability scenarios at this layer and perform robust tests based on the custom Cloud configurations already supported within the CloudSim.

As Cloud computing is a rapidly evolving research area, there is a severe lack of defined standards, tools and methods that can efficiently tackle the infrastructure and application level complexities. Hence in the near future there would be a number of research efforts both in academia and industry towards defining core algorithms, policies, application benchmarking based on execution contexts. By extending the basic functionalities already exposed by CloudSim, researchers would be able to perform tests based on specific scenarios and

configurations, hence allowing the development of best practices in all the critical aspects related to Cloud Computing.

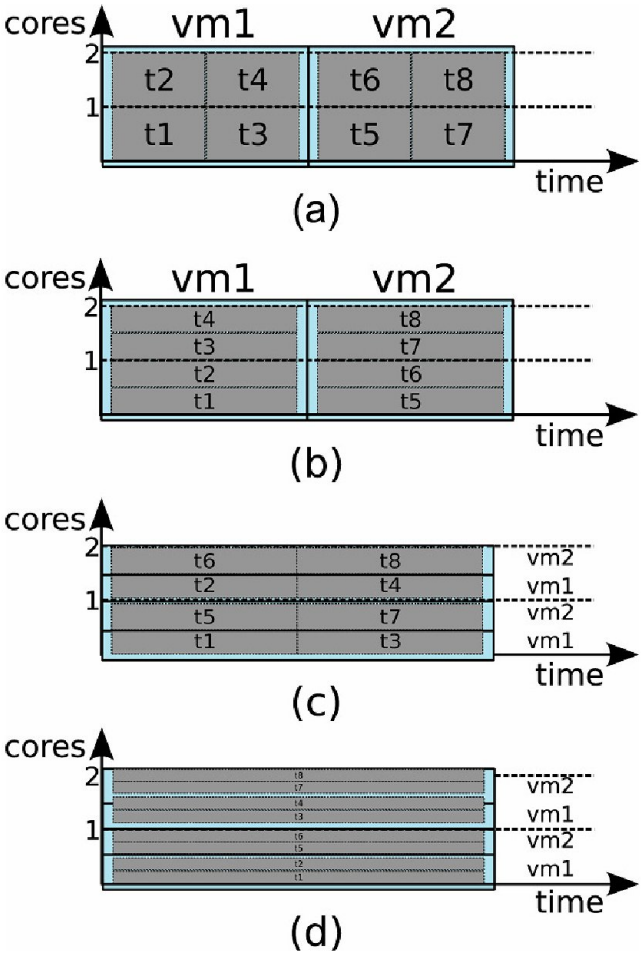


Figure 3. Effects of different scheduling policies in the task execution: (a) Space-shared for VMs and tasks, (b) Space-shared for VMs and time-shared for tasks, (c) Space-shared for VMs, time-shared for tasks, and (d) Space-shared for VMs and tasks.

One of the design decisions that we had to make as the CloudSim was being developed was whether to extensively reuse existing simulation libraries and frameworks or not. We decided to take advantage of already implemented, tested, and validated libraries such as GridSim and SimJava to handle low-level requirements of the system. For example, by using SimJava, we avoided reimplementation of event handling and message passing among components; this saved us a lot of time in software engineering and testing. Similarly, the use of the GridSim framework allowed us to reuse its implementation of networking, information services, files, users, and resources. Since SimJava and GridSim have been extensively utilized in conducting cutting edge research in Grid resource management by several researchers, bugs that may compromise the validity of the simulation have been already detected and fixed. By reusing these long validated frameworks, we were able to focus on critical aspects of the system that are relevant to Cloud computing, while taking advantage of the reliability of components that are not directly related to Clouds.

3.1. Modeling the Cloud

The core hardware infrastructure services related to the Clouds are modeled in the simulator by a Datacenter component for handling service requests. These requests are application elements sandboxed within VMs, which need to be allocated a share of processing power on Datacenter's host components. By VM processing, we mean set of operations related to VM life cycle: provisioning of a host to a VM, VM creation, VM destruction, and VM migration.

A Datacenter is composed by a set of hosts, which is responsible for managing VMs during their life cycles. Host is a component that represents a physical computing node in a Cloud: it is assigned a pre-configured processing (expressed in million of instructions per second – MIPS, per CPU core), memory, storage, and a scheduling policy for allocating processing cores to virtual machines. The Host component implements interfaces that support modeling and simulation of both single-core and multi-core nodes.

Allocation of application-specific VMs to Hosts in a Cloud-based data center is the responsibility of the Virtual Machine Provisioner component (refer to Figure 2). This component exposes a number of custom methods for researchers, which aids in implementation of new VM provisioning policies based on optimization goals (user centric, system centric). The default policy implemented by the VM Provisioner is a straightforward policy that allocates a VM to the Host in First-Come-First-Serve (FCFS) basis. The system parameters such as the required number of processing cores, memory and storage as requested by the Cloud user form the basis for such mappings. Other complicated policies can be written by the researchers based on the infrastructure and application demands.

For each Host component, the allocation of processing cores to VMs is done based on a host allocation. The policy takes into account how many processing cores will be delegated to each VM, and how much of the processing core's capacity will effectively be attributed for a given VM. So, it is possible to assign specific CPU cores to specific VMs (a space-shared policy) or to dynamically distribute the capacity of a core among VMs (time-shared policy), and to assign cores to VMs on demand, or to specify other policies.

Each Host component instantiates a VM scheduler component that implements the space-shared or time-shared policies for allocating cores to VMs. Cloud system developers and researchers can extend the VM scheduler component for experimenting with more custom allocation

policies. Next, the finer level details related to the time-shared and space-shared policies are described.

3.2. Modeling the VM allocation

One of the key aspects that make a Cloud computing infrastructure different from a Grid computing is the massive deployment of virtualization technologies and tools. Hence, as compared to Grids, we have in Clouds an extra layer (the virtualization) that acts as an execution and hosting environment for Cloud-based application services.

Hence, traditional application mapping models that assign individual application elements to computing nodes do not accurately represent the computational abstraction which is commonly associated with the Clouds. For example, consider a physical data center host that has single processing core, and there is a requirement of concurrently instantiating two VMs on that core. Even though in practice there is isolation between behaviors (application execution context) of both VMs, the amount of resources available to each VM is constrained by the total processing power of the host. This critical factor must be considered during the allocation process, to avoid creation of a VM that demands more processing power than the one available in the host, and must be considered during application execution, as task units in each virtual machine shares time slices of the same processing core.

To allow simulation of different policies under different levels of performance isolation, CloudSim supports VM scheduling at two levels: First, at the host level and second, at the VM level. At the first level, it is possible to specify how much of the overall processing power of each core in a host will be assigned to each VM. At the next level, the VMs assign specific amount of the available processing power to the individual task units that are hosted within its execution engine.

At each level, CloudSim implements the time-shared and space-shared resource allocation policies. To better illustrate the difference between these policies and their effect on the application performance, in Figure 3 we show a simple scheduling scenario. In the figure, a host with two CPU cores receives request for hosting two VMs, and each one requiring two cores and running four tasks units: t1, t2, t3 and t4 to be run in VM1, while t5, t6, t7, and t8 to be run in VM2.

Figure 3(a) presents a space-shared policy for both VMs and task units: as each VM requires two cores, only one VM can run at a given instance of time. Therefore, VM2 can only be assigned the core once VM1 finishes the execution of task units. The same happens for tasks hosted within the VM: as each task unit demands only one core, two of them run simultaneously, and the other two are queued until the completion of the earlier task units.

In Figure 3(b), space-shared policy is used for allocating VMs, but a time-shared policy is used for allocating individual task units within VM. So during a VM lifetime, all the tasks assigned to it dynamically context switch until their completion. This allocation policy enables the task units to be scheduled at an earlier time, but significantly affecting the completion time of task units that head the queue.

In Figure 3(c), a time-shared scheduling is used for VMs, and a space-shared one is used for task units. In this case, each VM receives a time slice of each processing core, and then slices are distributed to task units on space-shared basis. As the core is shared, the amount of processing power available to the VM is comparatively lesser than the aforementioned scenarios. As task unit assignment is space-shared, hence only one task can be allocated to each core, while others are queued in for future consideration.

Finally, in Figure 3(d) a time-shared allocation is applied for both VMs and task units. Hence, the processing power is concurrently shared by the VMs and the shares of each VM are concurrently divided among the task units assigned to each VM. In this case, there are no queues either for virtual machines or for task units.

3.3. Modeling the Cloud market

Support for services that act as a market maker enabling capability sharing across Cloud service providers and customer through its match making services is critical to Cloud computing. Further, these services need mechanisms to determine service costs and pricing policies. Modeling of costs and pricing policies is an important aspect to be considered when designing a Cloud simulator. To allow the modeling of the Cloud market, four market-related properties are associated to a data center: cost per processing, cost per unit of memory, cost per unit of storage, and cost per unit of used bandwidth. Cost per memory and storage incur during virtual machine creation. Cost per bandwidth incurs during data transfer. Besides costs for use of memory, storage, and bandwidth, the other cost is associated to use of processing resources. Inherited from the GridSim model, this cost is associated with the execution of user task units. So, if VMs were created but no task units were executed on them, only the costs of memory and storage will incur. This behavior may, of course, be changed by users.

4. Design and Implementation of CloudSim

The Class design diagram for the simulator is depicted in Figure 4. In this section, we provide finer details related to the fundamental classes of CloudSim, which are building blocks of the simulator.

Datacenter. This class models the core infrastructure level services (hardware, software) offered by resource providers in a Cloud computing environment. It encapsulates a set of compute hosts (blade servers) that can be either homogeneous or heterogeneous as regards to their resource configurations (memory, cores, capacity, and storage). Furthermore, every Datacenter component instantiates a generalized resource provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices.

DatacenterBroker. This class models a broker, which is responsible for mediating between users and service

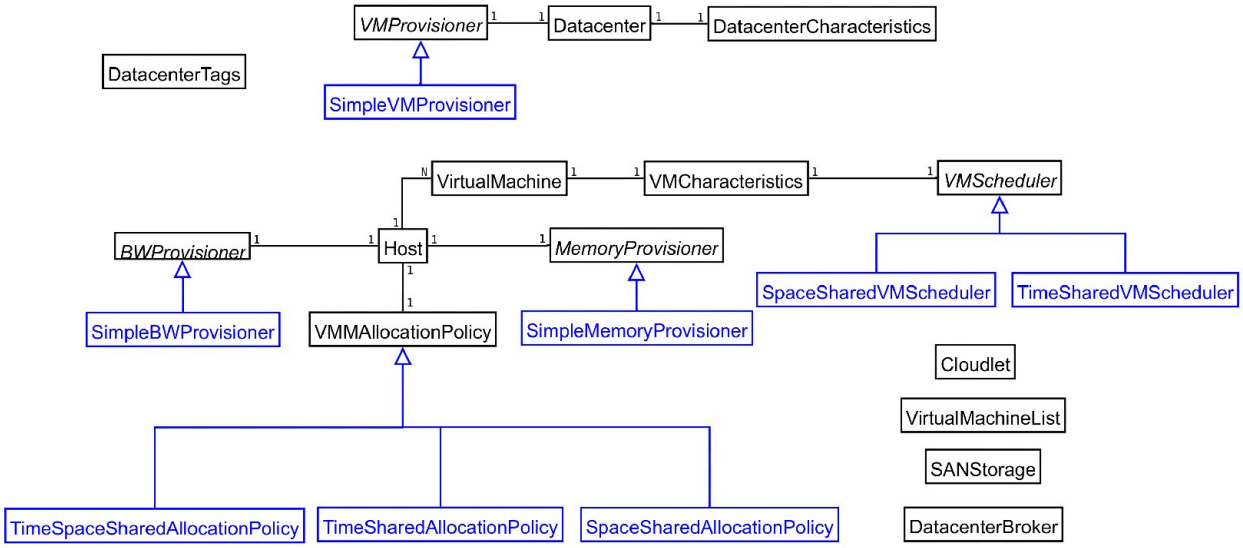


Figure 4. CloudSim class design diagram.

providers depending on users' QoS requirements and deploys service tasks across Clouds. The broker acting on behalf of users identifies suitable Cloud service providers through the Cloud Information Service (CIS) and negotiates with them for an allocation of resources that meets QoS needs of users. The researchers and system developers must extend this class for conducting experiments with their custom developed application placement policies.

SANStorage. This class models a storage area network that is commonly available to Cloud-based data centers for storing large chunks of data. SANStorage implements a simple interface that can be used to simulate storage and retrieval of any amount of data, at any time subject to the availability of network bandwidth. Accessing files in a SAN at run time incurs additional delays for task unit execution, due to time elapsed for transferring the required data files through the data center internal network.

VirtualMachine. This class models an instance of a VM, whose management during its life cycle is the responsibility of the Host component. As discussed earlier, a host can simultaneously instantiate multiple VMs and allocate cores based on predefined processor sharing policies (space-shared, time-shared). Every VM component has access to a component that stores the characteristics related to a VM, such as memory, processor, storage, and the VM's internal scheduling policy, which is extended from the abstract component called VMScheduling.

Cloudlet. This class models the Cloud-based application services (content delivery, social networking, business workflow), which are commonly deployed in the data centers. CloudSim represents the complexity of an application in terms of its computational requirements. Every application component has a pre-assigned instruction length (inherited from GridSim's Gridlet component) and amount of data transfer (both pre and post fetches) that needs to be undertaken for successfully hosting the application.

BWProvisioner. This is an abstract class that models the provisioning policy of bandwidth to VMs that are deployed on a Host component. The function of this component is to undertake the allocation of network bandwidths to set of competing VMs deployed across the data center. Cloud system developers and researchers can extend this class with their own policies (priority, QoS) to reflect the needs of their applications.

MemoryProvisioner. This is an abstract class that represents the provisioning policy for allocating memory to VMs. This component models policies for allocating physical memory spaces to the competing VMs. The execution and deployment of VM on a host is feasible only if the MemoryProvisioner component determines that the host has the amount of free memory, which is requested for the new VM deployment.

VMProvisioner. This abstract class represents the provisioning policy that a VM Monitor utilizes for allocating VMs to Hosts. The chief functionality of the VMProvisioner is to select available host in a data center, which meets the memory, storage, and availability requirement for a VM deployment. The default SimpleVMProvisioner implementation provided with the CloudSim package allocates VMs to the first available Host that meets the aforementioned requirements. Hosts are considered for mapping in a sequential order. However, more complicated policies can be easily implemented within this component for achieving optimized allocations, for example, selection of hosts based on their ability to meet QoS requirements such as response time, budget.

VMMAllocationPolicy. This is an abstract class implemented by a Host component that models the policies (space-shared, time-shared) required for allocating processing power to VMs. The functionalities of this class

can easily be overridden to accommodate application specific processor sharing policies.

4.1. Entities and threading

As the CloudSim programmatically builds upon the SimJava discrete event simulation engine, it preserves the SimJava's threading model for creation of simulation entities. A programming component is referred to as an entity if it directly extends the core Sim_Entity component of SimJava, which implements the Runnable interface. Every entity is capable of sending and receiving messages through the SimJava's shared event queue. The message propagation (sending and receiving) occurs through input and output ports that SimJava associates with each entity in the simulation system. Since, threads incur a lot of memory and processor context switching overhead; having a large number of threads/entities in a simulation environment can be performance bottleneck due to limited scalability. To counter this behavior, CloudSim minimizes the number of entities in the system by implementing only the core components (Users and Datacenters) as the inherited members of SimJava entities. This design decision is significant as it helps CloudSim in modeling a really large scale simulation environment on a computing machine (desktops, laptops) with moderate processing capacity. Other key CloudSim components such as VMs, provisioning policies, hosts are instantiated as standalone objects, which are lightweight and do not compete for processing power.

Hence, regardless of the number of hosts in a simulated data center, the runtime environment (java virtual machine) needs to manage only three threads (User, Datacenter, and Broker). As the processing of task units is handled by respective VMs, therefore their (task) progress must be updated and monitored after every simulation step. To handle this, an internal event is generated regarding the expected completion time of a task unit to inform the Datacenter entity about the future completion events. Thus, at each simulation step, each Datacenter invokes a method called updateVMsProcessing() for every host in the system, to update processing of tasks running within the VMs. The argument of this method is the current simulation time and the return type is the next expected completion time of a task running in one of the VMs on a particular host. The least time among all the finish times returned by the hosts is noted for the next internal event.

At the host level, invocation of updateVMsProcessing() triggers an updateGridletsProcessing() method, which directs every VM to update its tasks unit status (finish, suspended, executing) with the Datacenter entity. This method implements the similar logic as described previously for updateVMsProcessing() but at the VM level. Once this method is called, VMs return the next expected completion time of the task units currently managed by them. The least completion time among all the computed values is send to the Datacenter entity. As a result, completion times are kept in a queue that is queried by Datacenter after each event processing step. If there are completed tasks waiting in the queue, then they are removed from it and sent back to the user.

4.2. Communication among Entities

Figure 5 depicts the flow of communication among core CloudSim entities. In the beginning of the simulation, each Datacenter entity registers itself with the CIS (Cloud Information Service) Registry. CIS provides database level match-making services for mapping user requests to suitable Cloud providers. Brokers acting on behalf of users consult the CIS service about the list of Clouds who offer infrastructure services matching user's application requirements. In case the match occurs the broker deploys the application with the Cloud that was suggested by the CIS.

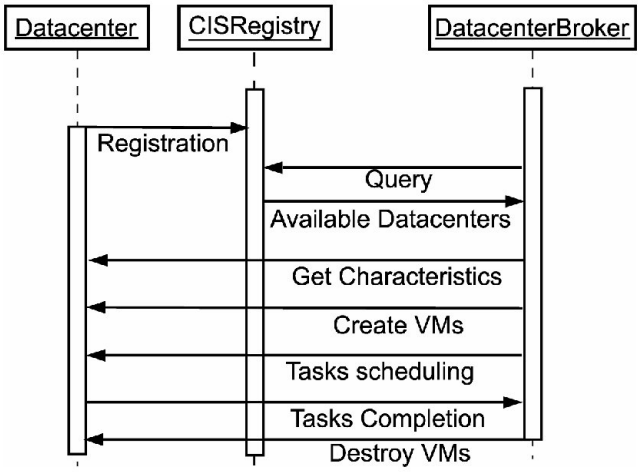


Figure 5. Simulation data flow.

The communication flow described so far relates to the basic flow in a simulated experiment. Some variations in this flow are possible depending on policies. For example, messages from Brokers to Datacenters may require a confirmation, from the part of the Datacenter, about the execution of the action, or the maximum number of VMs a user can create may be negotiated before VM creation.

5. Tests and Evaluation

In this section, we present tests and evaluation that we undertook in order to quantify the efficiency of CloudSim in modeling and simulating Cloud computing environment. The tests were conducted on a Celeron machine having configuration: 1.86GHz with 1MB of L2 cache and 1 GB of RAM running a standard Ubuntu Linux version 8.04 and JDK 1.6.

To evaluate the overhead in building a simulated Cloud computing environment that consists of a single data center, a broker and a user, we performed series of experiments. The number of hosts in the data center in each experiment was varied from 100 to 100000. As the goal of these tests were to evaluate the computing power requirement to instantiate the Cloud simulation infrastructure, no attention was given to the user workload.

For the memory test, we profile the total physical memory used by the hosting computer (Celeron machine) in order to fully instantiate and load the CloudSim environment. The total delay in instantiating the simulation environment is the time difference between the following events: (i) the time at which the runtime environment (java virtual machine) is directed to load the CloudSim program; and (ii) the instance at which CloudSim's entities and components are fully initialized and are ready to process events.

Figures 6 and 7 present, respectively, the amount of time and the amount of memory is required to instantiate the experiment when the number of hosts in a data center increases. The growth in memory consumption (see Fig. 7) is linear, with an experiment with 100000 machines demanding 75MB of RAM. It makes our simulation suitable to run even on simple desktop computers with moderated processing power because CloudSim memory requirements, even for larger simulated environments can easily be provided by such computers.

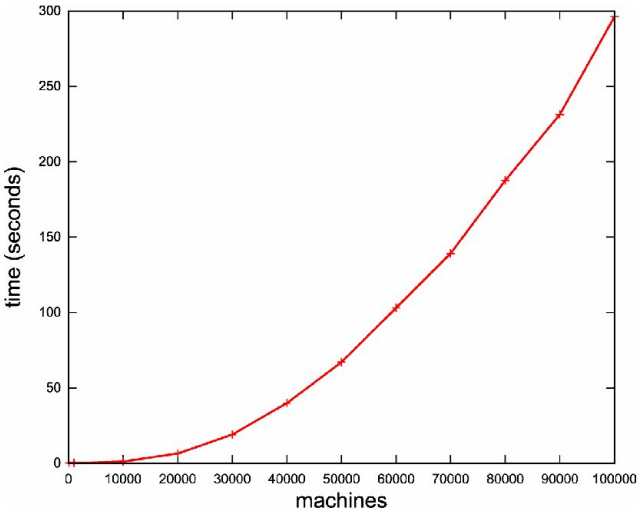


Figure 6. Time to simulation instantiation.

Regarding time overhead related to simulation instantiation, the growth in terms of time grows exponentially with the number of hosts/machines. Nevertheless, the time to instantiate 100000 machines is below 5 minutes, which is reasonable considering the scale of the experiment. Currently, we are investigating the cause of this behavior to avoid it in future versions of CloudSim.

The next test aimed at quantifying the performance of CloudSim's core components when subjected to user workloads such as VM creation, task unit execution. The simulation environment consisted of a data center with 10000 hosts, where each host was modeled to have a single CPU core (1000MIPS), 1GB of RAM memory and 2TB of storage. Scheduling policy for VMs was Space-shared, which meant only one VM was allowed to be hosted in a host at a given instance of time. We modeled the user (through the DatacenterBroker) to request creation of 50 VMs having following constraints: 512MB of physical memory, 1 CPU core and 1GB of storage. The application unit was modeled to consist of 500 task units, with each task unit requiring 1200000 million instructions (20 minutes in the simulated hosts) to be executed on a host. As networking was not a concern in these experiments, task units required only 300kB of data to be transferred to and from the data center.

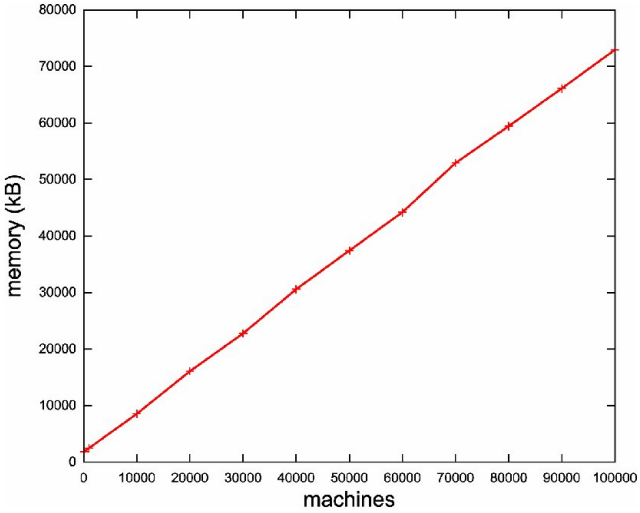


Figure 7. Memory usage in resources instantiation.

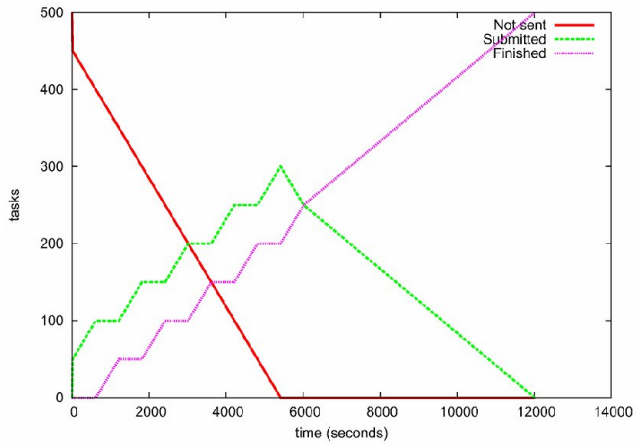


Figure 8. Tasks execution with space-shared scheduling of tasks.

After creation of VMs, task units were submitted in groups of 50 (one submitted to each VM) every 10 minutes. The VM were configured to use both space-shared and time-shared policies for allocating tasks units to the processing cores.

Figures 8 and 9 present task units progress status with increase in simulation steps (time) for the space-shared test and for the time-shared tests respectively. As expected, in the space-shared case every task took 20 minutes for completion as they had dedicated access to the processing core. Since, in this policy each task unit had its own dedicated core therefore number of incoming tasks or

queue size did not affect execution time of individual task units.

However, in the time-shared case execution time of each task varied with increase in number of submitted taks units. Using this policy, execution time is significantly affected as the processing core is concurrently context switched among the list of scheduled tasks. The first group of 50 tasks was able to complete earlier than the other ones because in this case the hosts were not over-loaded at the beginning of execution. To the end, as more tasks reached completion, comparatively more hosts became available for allocation. Due to this we observed improved response time for the tasks as shown in Figure 9.

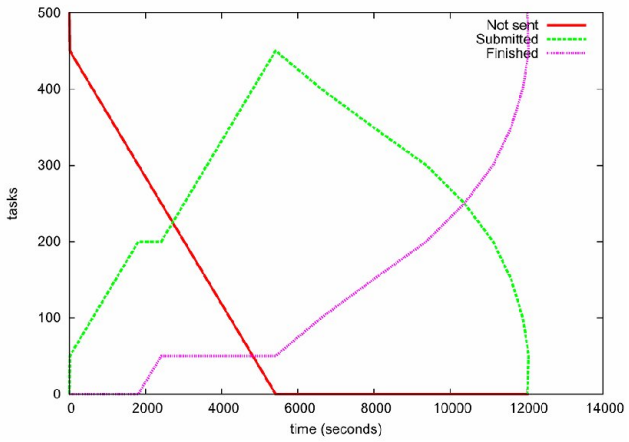


Figure 9. Task execution with time-shared scheduling of tasks.

6. Conclusion and Future Work

The recent efforts to design and develop Cloud technologies focus on defining novel methods, policies and mechanisms for efficiently managing Cloud infrastructures. To test these newly developed methods and policies, researchers need tools that allow them to evaluate the hypothesis prior to real deployment in an environment where one can reproduce tests. Especially in the case of Cloud computing, where access to the infrastructure incurs payments in real currency, simulation-based approaches offer significant benefits, as it allows Cloud developers to test performance of their provisioning and service delivery policies in repeatable and controllable environment free of cost, and to tune the performance bottlenecks before deploying on real Clouds.

To this end, we developed the CloudSim system, a framework for modeling and simulation of next-generation Clouds. As a completely customizable tool, it allows extension and definition of policies in all the components of the software stack, which makes it suitable as a research tool that can handle the complexities arising from simulated environments. As future work, we are planning to incorporate new pricing and provisioning policies to CloudSim, in order to offer a built-in support to simulate the currently available Clouds. We also intend to provide support for simulating federated network of clouds, with focus on designing and testing elastic Cloud applications.

References

[1] R. Buyya, C. S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities. *In Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*, 2008.

[2] D. Chappell. Introducing the Azure services platform. White paper, Oct. 2008.

[3] X. Chu et al. Aneka: Next-generation enterprise grid platform for e-science and e-business applications. In *Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing*, 2007.

[4] C. L. Dumitrescu and I. Foster. GangSim: a simulator for grid scheduling studies. *In Proceedings of the IEEE International Symposium on Cluster Computing and the Grid,* 2005.

[5] I. Foster and C. Kesselman (editors). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[6] F. Howell and R. Mcnab. SimJava: A discrete event simulation library for java. *In Proceedings of the first International Conference on Web-Based Modeling and Simulation,* 1998.

[7] A. Legrand, L. Marchal, and H. Casanova. Scheduling distributed applications: the SimGrid simulation framework. In *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2003.

[8] J. E. Smith and R. Nair. *Virtual Machines: Versatile platforms for systems and processes.* Morgan Kauffmann, 2005.

[9] R. Buyya and M. Murshed, GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing, *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[10] A. Weiss. Computing in the clouds. *NetWorker*, 11(4):16–25, Dec. 2007.

[11] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. *Above the Clouds: A Berkeley View of Cloud computing*. Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009.

Security issues in Cloud

Need for Security and Privacy in Cloud

- Cloud computing - grid + distributed computing, utilizing the Internet as a service delivery network.
- The public Cloud environment is extremely complex when compared to a traditional data center environment.
- Virtual environments are used in Cloud to achieve multi-tenancy.
- Vulnerabilities in virtual machines pose direct threat to the privacy and security of the Cloud services.

Cont..

- Shared and distributed resources in the Cloud systems make it hard to develop a security model for ensuring the data security and privacy.
- Due to transparency issues, no Cloud provider allows its customers to implement intrusion detection or security monitoring systems extending into the management services layer behind virtualized Cloud instances.

Security issues in cloud computing

- Vulnerabilities - the loopholes in the security architecture of Cloud, which can be exploited by an adversary via sophisticated techniques to gain access to the network and other infrastructure resources.
- A threat - a potential (or actual adverse) event that may be malicious or incidental (i.e. failure of a storage device), compromising Cloud resources.
- An attack - an action to harm Cloud resources.
- Exploitation of vulnerabilities would affect the availability and economic benefit of Cloud computing.

Vulnerabilities in Cloud computing

- Virtualization/multi-tenancy is the basis for Cloud computing architecture.
- Mainly three types of virtualization : OS level, application level, and Hypervisor level.
- In OS level virtualization, multiple guest OSs are running on host OS that has visibility and control on each guest OS. In this, an attacker can get control on the entire guest OSs by compromising the host OS.
- In application based virtualization, virtualization is enabled on the top layer of the host OS. In this, each VM has its guest OS and related applications. Application based virtualization also suffers from the same vulnerability as in OS based vulnerabilities.
- Hypervisor or virtual machine monitor (VMM) is just like code embedded to host OS. Such code may contain native errors. This code is available at boot time of the host OS to control multiple guest OSs. If the hypervisor is compromised, then the entire controlled guest OSs can be compromised.
- Vulnerabilities in virtualization or hypervisor allows an attacker to perform cross-VM side-channel attacks and DoS attacks. For instance, a malformed code in Microsoft's Hyper-V run by an authenticated user in one of the VM caused a DoS attack.

Cont..

- Vulnerabilities in Internet protocols may prove to be an implicit way of attacking the Cloud system that include common types of attacks like man-in-the-middle attack, IP spoofing, ARP spoofing, DNS poisoning, RIP attacks, and flooding.
- Vulnerabilities like SQL injection flaw, OS injection flaw, and Lightweight Directory Access Protocol (LDAP) injection flaw are used to disclose application components. Such vulnerabilities are the outcomes of defects in design and architecture of applications. These data may be the organization's applications or private data of other organization's applications residing on the same Cloud.

Cont..

- Cloud providers publish a set of software interfaces (or APIs) that customers can use to manage and interact with Cloud services. Service provisioning, management, orchestration, and monitoring are performed using these interfaces via clients (e.g. Web browser).
- Security and availability of Cloud services depend on the security of these APIs.
- Examples of browser based attacks (HTML based services) are SSL certificate spoofing, attacks on browser caches and phishing attacks on mail clients

Threats to cloud computing

- **Changes to business model -** Cloud computing changes the way in which IT services are delivered. As servers, storage and applications are provided by off-site external service providers, organizations need to evaluate the risks associated with the loss of control over the infrastructure. (A reliable end-to-end encryption and appropriate trust management scheme can simplify such a threat to some extent.)
- **Abusive use of Cloud computing -** Cloud computing provides several utilities including bandwidth and storage capacities. Some vendors also give a predefined trial period to use their services. However, they do not have sufficient control over the attackers, malicious users or spammers that can take advantages of the trials. These can often allow an intruder to plant a malicious attack and prove to be a platform for serious attacks. Areas of concern include password and key cracking, launching dynamic attack points, DDoS, Captcha solving farms, etc. Such threats affect the IaaS and PaaS service models. (For protection, initial registration should be through proper validation/verification and through stronger authentication. In addition to this, the user's network traffic should be monitored comprehensively.)

Cont..

- **Insecure interfaces and API -** Cloud providers often publish a set of APIs to allow their customers to design an interface for interacting with Cloud services. These interfaces often add a layer on top of the framework, which in turn would increase the complexity of Cloud. Such type of threat may affect the IaaS, PaaS, and SaaS service models. (This can be avoided by using a proper security model for Cloud provider's interface and ensuring strong authentication and access control mechanism with encrypted transmission.)

- **Malicious insiders -** Most of the organizations hide their policies regarding the level of access to employees and their recruitment procedure for employees. However, using a higher level of access, an employee can gain access to confidential data and servicesThis type of threat may be relevant to SaaS, PaaS, and IaaS. (To avoid this risk, more transparency is required in security and management process including compliance reporting and breach notification.)

Cont..

- **Shared technology issues/multi-tenancy nature -** In multi-tenant architecture, virtualization is used to offer shared on-demand services. This type of threat affects IaaS. (Implementation of SLA for patching, strong authentication, and access control to administrative tasks are some of the solutions to address this issue.)

- **Data loss and leakage -** This may include data compromise, deletion, or modification. Due to the dynamic and shared nature of the Cloud, such threat could prove to be a major issue leading to data theft. This threat can applicable to SaaS, PaaS, and IaaS. (Solutions include security of API, data integrity, secure storage for used keys, data backup, and retention policies)

Cont..

- **Service hijacking -** Service hijacking may redirect the client to an illegitimate website. User accounts and service instances could in turn make a new base for attackers. This threat can affect IaaS, PaaS, and SaaS. (Some of the mitigation strategies to address this threat include security policies, strong authentication, and activity monitoring.)
- **Risk profiling -** Cloud offerings make organizations less involved with ownership and maintenance of hardware and software. This offers significant advantages. However, this makes them unaware of internal security procedures, security compliance, hardening, patching, auditing, and logging process and expose the organization to greater risk. (To avoid this Cloud provider should disclose partial infrastructure details, logs, and data. In addition to this, there should also be a monitoring and alerting system.)

Cont..

- **Identity theft -** Identity theft is a form of fraud in which someone pretends to be someone else, to access resources or obtain credit and other benefits. The victim (of identity theft) can suffer adverse consequences and losses and held accountable for the perpetrator's actions. Relevant security risks include weak password recovery workflows, phishing attacks, key loggers, etc. This affects SaaS, PaaS, and IaaS. (The solution is to use strong authentication mechanisms.)

Attacks on Cloud computing

- **Zombie attack -** Through the Internet, an attacker tries to flood the victim by sending requests from innocent hosts in the network. These types of hosts are called *zombies*. The Cloud may be overloaded to serve a number of requests, and hence exhausted, which can cause DoS (Denial of Service) or DDoS (distributed denial of service) to the servers. Cloud in the presence of attacker's flooded requests cannot serve valid user's requests. (Better authentication and authorization and IDS/IPS can provide protection against such an attack.)
- **Service injection attack -** Cloud system is responsible for determining and eventually instantiating a free-to-use instance of the requested service. The address for accessing that new instance is to be communicated back to the requesting user. An adversary tries to inject a malicious service or new virtual machine into the Cloud system and can provide malicious service to users. (Service integrity checking module should be implemented. Strong isolation between VMs may disable the attacker from injecting malicious code in the neighbour's VM.)

Cont..

- **Attacks on virtualization -** There are mainly two types of attacks performed over virtualization: VM Escape and Rootkit in hypervisor.
- VM Escape*:* An attacker's program running in a VM breaks the isolation layer in order to run with the hypervisor's root privileges instead with the VM privileges. This allows an attacker to interact directly with the hypervisor. Therefore, VM Escape from the isolation is provided by the virtual layer. By VM Escape, an attacker gets access to the host OS and the other VMs running on the physical machine.

- Rootkit in Hypervisor*:* VM-based rootkits initiate a hypervisor compromising the existing host OS to a VM. The new guest OS assumes that it is running as the host OS with the corresponding control over the resources, however, in reality this host does not exist. (This allows an attacker to control over any VM running on the host machine and to manipulate the activities on the system.)

Cont..

- **Man-in-the Middle attack -** If secure socket layer (SSL) is not properly configured, then any attacker is able to access the data exchange between two parties. In Cloud, an attacker is able to access the data communication among data centers. (Proper SSL configuration and data communication tests between authorized parties can be useful to reduce the risk of Man-in-the-Middle attack.)

- **Metadata spoofing attack -** An adversary modifies or changes the service's Web Services Description Language (WSDL) file where descriptions about service instances are stored. If the adversary succeeds to interrupt service invocation code from WSDL file at delivering time, then this attack can be possible. (To overcome such an attack, information about services and applications should be kept in encrypted form. Strong authentication (and authorization) should be enforced for accessing such critical information.)

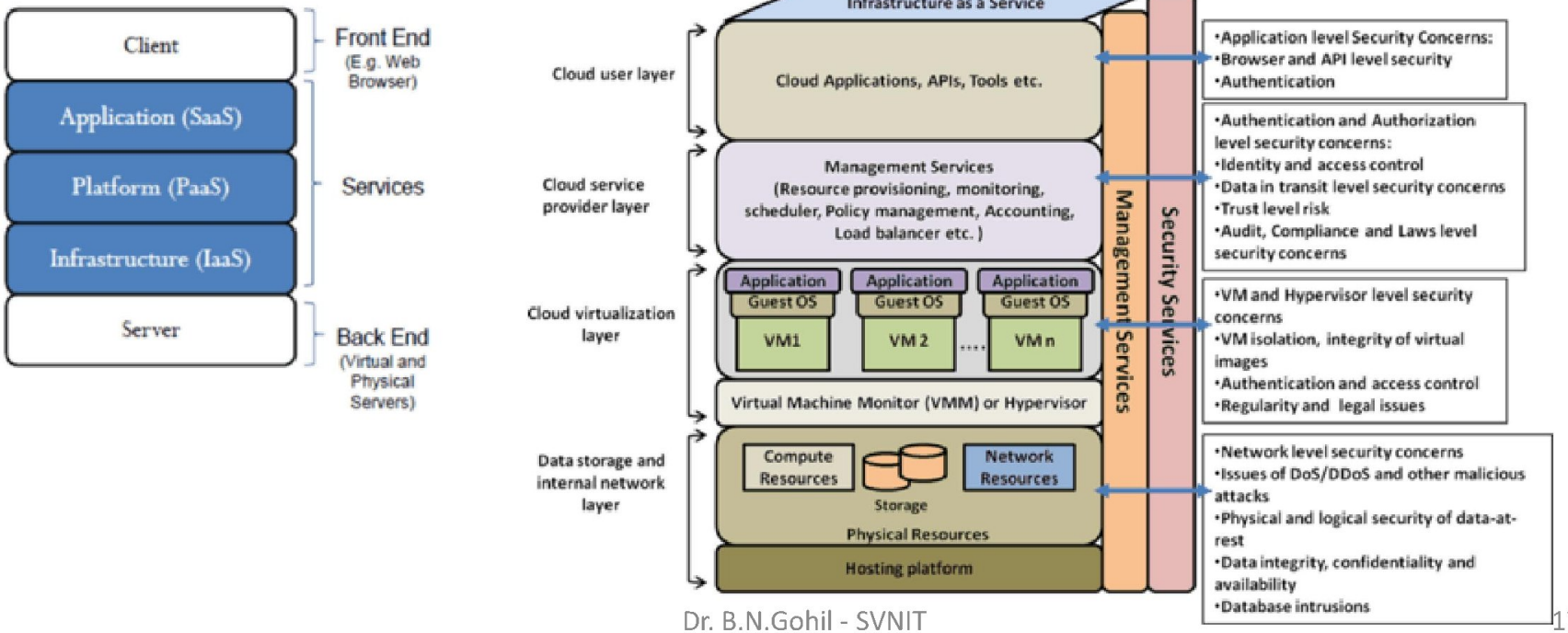
Cont..

- **Phishing attack -** Phishing attacks are well known for manipulating a web link and redirecting a user to a false link to get sensitive data.

- **Backdoor channel attack -** It is a passive attack, which allows hackers to gain remote access to the compromised system. Using backdoor channels, hackers can be able to control victim's resources and can make it a *zombie* for attempting a DDoS attack. (Better authentication and isolation between VMs can provide protection against such attacks.)

Security issues at different layers in Cloud

- Different layers with associate security concerns in cloud enabled system:



Application level security issues

- It refers to the usage of software and hardware resources for providing security to applications such that the attackers are not able to get control over applications and make desirable changes to their format.
- Since Web applications and SaaS are tightly coupled with providing Cloud services, the security and availability of general cloud services are dependent upon the security of Web browsers, APIs and vulnerability free applications.
- A Web browser is the platform independent client program that is mostly used to access the Cloud services (SaaS), web applications, web pages, or web 2.0. It uses SSL/TLS protocols for secure transmission and authentication of data.
- Therefore, attacks on browser based Cloud authentication directly affect the security of Cloud applications. Any attacker can get access to other user's XML tokens (authentication related credentials in the browser) and accesses the services of the victim. One of the solutions viz; XML signature and XML encryption can be used to enhance browser security.

Cont..

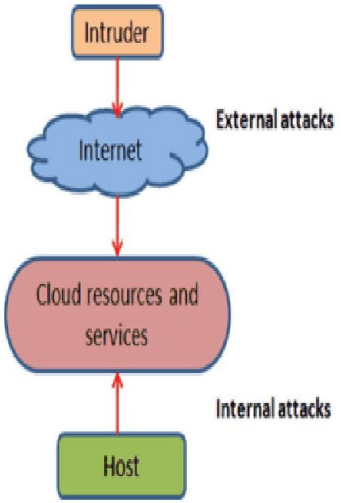
- **Service availability -** Temporary or permanent loss of services and DoS/DDoS attacks are the main threats affecting availability of Cloud services. For better QoS, services should be available as promised when they are requested.
- One such incident is the database cluster failure caused at Salesforce.com.
- In February 2011, Gmail went down for a few hours and due to service disruption, 0.29 % of Gmail users were affected and lost their previous emails and other data.
- On March 28, 2011, thousands of users registered at Intuit (which offers financial and tax preparation software and related services) experienced an outage for 2 to 5 days during the network configuration update and scheduled maintenance. As a result, customers were blocked to access offered services. To address such issues, proper configuration of an IDS/IPS can be investigated.

Cont..

- **Integrity of workload state -** The integrity for the state of a workload should be preserved to ensure expected results. Applications involving workflows are required to store temporary results of computation at different levels. There is no standard mechanism used to secure such sensitive files. If these sensitive files are disclosed to an attacker, he/she may be able to threaten the expected behavior of the application.

Network level security issues

- The network is the backbone of Cloud, and hence vulnerabilities in network directly affect the security of Cloud.
- Security issues at network level should be considered in terms of both external and internal networks.
- An adversary outside the Cloud network often performs DoS or DDoS attacks to affect the availability of Cloud services and resources.
- DoS/DDoS attacks reduce the bandwidth and increases the congestion causing poor service to the users.
- Due to the distributed nature of the Cloud, it is hard to prevent DoS/DDoS and Economic Denial of Sustainability (EDoS can be called as HTTP and XML based DDoS) attacks.



Cont..

- Some common attacks at the network layer are DNS poisoning attack, Sniffer attack, Port scanning, Cross site scripting, ARP spoofing, IP spoofing, and phishing attack, which are executed to gain access of Cloud resources.
- Internal network attacker (authorized users or users within the cloud network) can easily get access to other user's resources without being detected. An insider has higher privileges and knowledge (related to network, security mechanism, and resources to attack) than the external attacker. Therefore, it is easy for an insider to penetrate an attack than external attackers.
- Major security issues at network level include vulnerabilities in Internet protocols, authorization, and authentication, intrusions, backdoor attack, session hijacking, and clear data transmission.
- To address some of the issues at the network level, major Cloud providers (like Amazon, Window Azure, Rack Space, Eucalyptus, etc.) are running their applications behind firewall. It only provides security at boundary of network and cannot detect the internal attacks. Network based intrusion detection system (NIDS) can be integrated to address some of the security issues.
- NIDS should be configured for detecting external intrusions as well as internal intrusions. It should also be capable of detecting intrusions from encrypted traffic.

Data storage level security issues

- Security issues during data-in-transit, data-at-rest, data lineage, data eminence, data provenance, data recovery, data location, data breaches, and investigative support.
- In data-in-transit, adversary in network affects the confidentiality and integrity of data. The biggest risks for data-in-transit include poor encryption technology and network protocols.
- Data at rest (stored in Cloud storage) needs physical, logical, and personnel access control policies.

Cont..

- Tracing the data path is known as data lineage and it is important for auditing purposes in the cloud. Due to the shared environment, maintaining the integrity of data is the most challenging task in the Cloud.
- Data-Remanence refers to the data left out in case of data transfer or data removal. It causes minimal security threats viz; disclosure of sensitive information, data sold to others, etc.
- Data recovery is one of the most challenging problems. Data can be lost due to accidental damage or natural disaster to storage. It poses a risk to data availability for users.
- Tracing location of data is difficult in the Cloud since user's data are dynamically migrated from one region (or country) to another region (or country). It increases risk of data privacy and security since data owner loses the control over his/her data.

Virtualization level security issues

- In the virtualized (multi-tenant) environment, multiple OSs run concurrently on a host computer using hypervisor.
- As the number of guest operating systems (OSs) running on a hypervisor increase, the security concerns with that newer guest OSs also increase. Because it is not possible to keep track of all guest OSs, and hence maintaining the security of those OSs is difficult.
- It may happen that a guest system tries to run a malicious code on the host system and bring the system down or take full control of the system and block access to other guest OSs.

Cont..

- Isolation between two VMs is not completely adequate by current virtual machine monitors (VMMs).
- By compromising the lower layer hypervisor vulnerabilities, an attacker can gain control over installed VMs, for example, Bluepill, SubVirt, and DKSM are some well-known attacks on the virtual layer.

Cont..

- Virtualization based malware and rootkit: New generation of rootkits that benefit from the processor technology that allows an attacker to insert an additional hypervisor between the hardware and the software.
- The hypervisor takes control of the system and converts the original operating system into a virtual guest on the fly.
- In contrast to software-based virtualization, this kind of hijacking does not need a restart, and that makes it all the more difficult to detect the intrusion.

Cont..

- Sharing of VM images in Cloud introduces security risks.
- The owner of an image is concerned about confidentiality (e.g. unauthorized accesses to the image). The user of an image is concerned about safety (e.g. a malicious image that is capable of corrupting or stealing the user's own private data).

Authentication and access control level security issues

- In Cloud, client's information is transmitted over the Internet, which poses data ownership issues. As this information is processed outside the enterprise, it brings an inherent level of risk.
- This issue is addressed by providing support for security assertion markup language (SAML) federation protocol (which contains authentication credentials in the form of SAML assertions) with their own authentication protocol.
- SAML is issued to exchange information, such as assertions related to a subject or authentication information between the cooperating domains. The request and response messages of it are mapped over Simple Object Access Protocol (SOAP) relying on XML.
- Using a Signature Wrapping Attack, it is possible to modify an eavesdropped message despite of it being digitally signed. Thus, an attacker may be able to execute arbitrary machine commands on behalf of a legitimate user.
- To address such issues, data should be transmitted via secured channel and fine-grained authentication and authorization techniques can be used for preventing data from unauthorized access.

Trust level security issues

- This is one of the serious problems in the Cloud.
- Since users have lack of control over resources, they have to rely on trust mechanisms and contracts in conjunction with mechanisms that provide a compensation.
- Trust is a very fuzzy concept and very difficult to calculate in a heterogeneous environment that is assessed by a human or social trust.
- Contractors may be sub-contracting without user's knowledge.
- Limiting visibility of network and system monitoring to user poses major trust issues.
- Contract requirements may not be propagated down the sub-contract chain.
- Employees (authorized users) or malicious insiders of organizations often perform attacks that affect the confidentiality and privacy of other users' data as well as resources.

Cont..

- Lack of public relations poses a trust issue.
- Data processing outside the organizations poses an inherent level of risk. There is no direct control on some service components outside the organization.
- Limiting visibility of network and system monitoring to user may also pose a trust issue.
- This issue can be addressed by providing an adequate means of visibility of the monitoring system.
- Cross-site scripting, access control weaknesses, insecure storage, and insecure configuration are some of the threat examples.
- Advanced cryptographic techniques and signature techniques can be used to address trust issues when outsourcing data.

Security issues related to auditing, regulatory compliance and laws

- Audit and compliance to internal processes and external processes must be met with classified requirement and customer agreements, laws, and regulations. Therefore, such policies should be monitored.
- The multi-tenancy nature of Cloud increases the difficulty of monitoring and logging process of VMs.
- Due to the dynamic nature of the Cloud, it is difficult to audit and manage compliance by coordination with external auditing and regulatory bodies.
- There are different types of compliance.

Cont..

- Privacy compliance*:* Only owners of the data are responsible for the security and privacy of their outsourced data even if the data is held by a service provider. This is due to the various laws and regulations in different countries. It poses a risk of data security, confidentiality, and availability. This is an open problem for providing transparency and controlled environment to owners about their data.

- Geographic compliance*:* If the tenant or cloud customer operates in the United States, Canada, or the European Union, they are subject to numerous regulatory requirements. These include control objectives for information and related technology. These laws might relate to where the data is stored or transferred, as well as how well this data is protected from a confidentiality aspect.

- Industry compliance*:* Industry compliance considerations are typically seen as an area where many Cloud migrations flounder. Typical regulatory requirements can include: Payment Card Industry Data Security Standard (PCI-DSS), Health Insurance Portability and Accountability Act (HIPAA), Family Educational Rights and Privacy Act (FERPA), Federal Information Processing Standard (FIPS) 140-2, and Trusted Internet Connections (TIC) compliance.

Security standards

- It defines the processes, procedures, and practices necessary for implementing a security program.
- These standards also apply to cloud related IT activities and include specific steps that should be taken to ensure a secure environment is maintained that provides privacy and security of confidential information in a cloud environment.

Security Assertion Markup Language (SAML)

- SAML is an XML-based standard for communicating authentication, authorization, and attribute information among online partners.
- It allows businesses to securely send assertions between partner organizations regarding the identity and entitlements of a principal.
- The Organization for the Advancement of Structured Information Standards (OASIS) Security Services Technical Committee is in charge of defining, enhancing, and maintaining the SAML specifications.
- SAML is built on a number of existing standards, namely, SOAP, HTTP, and XML.
- SAML relies on HTTP as its communications protocol and specifies the use of SOAP.
- SAML assertions and protocols are specified using XML schema.
- Both SAML 1.1 and SAML 2.0 use digital signatures(based on the XMLSignature standard) for authentication and message integrity.

Open Authentication (OAuth)

- OAuth is an open protocol, initiated by Blaine Cook and Chris Messina, to allow secure API authorization in a simple, standardized method for various types of web applications.
- OAuth is a method for publishing and interacting with protected data.
- For developers, OAuth provides users access to their data while protecting account credentials.
- OAuth allows users to grant access to their information, which is shared by the service provider and consumers without sharing all of their identity.
- The Core designation is used to stress that this is the baseline, and other extensions and protocols can build on it.
- OAuth by itself provides no privacy at all and depends on other protocols such as SSL to accomplish that.
- With Oauth, sites use tokens coupled with shared secrets to access resources.
- Secrets, just like passwords, must be protected.

OpenID

- OpenID is an open, decentralized standard for user authentication and access control that allows users to log onto many services using the same digital identity.
- It is a single-sign-on (SSO) method of access control.
- It replaces the common log-in process (i.e., a log-in name and a password) by allowing users to log in once and gain access to resources across participating systems.
- An OpenID is in the form of a unique URL and is authenticated by the entity hosting the OpenID URL.
- The OpenID protocol does not rely on a central authority to authenticate a user's identity.
- Neither the OpenID protocol nor any web sites requiring identification can mandate that a specific type of authentication be used; nonstandard forms of authentication such as smart cards, biometrics, or ordinary passwords are allowed.
- A typical scenario for using OpenID might be something like this: A user visits a web site that displays an OpenID log-in form somewhere on the page. Unlike a typical log-in form, which has fields for user name and password, the OpenID log-in form has only one field for the OpenID identifier (which is an OpenID URL). This form is connected to an implementation of an OpenID client library. A user will have previously registered an OpenID identifier with an OpenID identity provider. The user types this OpenID identifier into the OpenID log-in form.

SOA and Cloud Computing

Technological drivers for Cloud

- The success of cloud computing is due to service-oriented architecture (SOA), virtualization, multicore technology, memory and storage technologies, networking technologies, Web 2.0, and Web 3.0.
- Also, the advancements in programming models, software development models, pervasive computing, operating systems (OSs), and application environment have contributed to the successful deployment of various clouds.

SOA – Service Oriented Architecture

- SOA is a flexible set of design principles and standards used for systems development and integration.
- A properly implemented SOA-based system provides a loosely coupled set of services that can be used by the service consumers for meeting their service requirements within various business domains.
- Cloud computing is a service delivery model in which shared services and resources are consumed by the users across the Internet just like a public utility on an on-demand basis.
- SOA is used by enterprise applications, and cloud computing is used for availing the various Internet-based services.

- Different companies or service providers may offer various services such as financial services, health-care services, manufacturing services, and HR services.
- Various users can acquire and leverage the offered services through the Internet.
- The programs running on cloud could be implemented using SOA-related technologies.
- A cloud user can combine the services offered by a cloud service provider (CSP) with other in-house and public cloud services to create SOA-based composite applications.

Service

- Definition: The performance of work (a function) by one for another.

- The notion of **Service** also includes:

  - The capability to perform work for another.
  - The specification of the work offered for another.
  - The offer to perform work for another.

- *In SOA, services are the mechanisms by which needs and capabilities are brought together.*

Example

- An electric utility has the capacity to generate and distribute electricity (the underlying capability). The wiring from the electric company's distribution grid (the service) provides the means to supply electricity to support typical usage for a residential consumer's house (service functionality), and a consumer accesses electricity generated (the output of invoking the service) via a wall outlet (service interface). In order to use the electricity, a consumer needs to understand what type of plug to use, what is the voltage of the supply, and possible limits to the load; the utility presumes that the customer will only connect devices that are compatible with the voltage provided and load supported; and the consumer in turn assumes that compatible consumer devices can be connected without damage or harm (service technical assumptions).

Cont.

- A residential or business user will need to open an account with the utility in order to use the supply (service constraint) and the utility will meter usage and expects the consumer to pay for use at the rate prescribed (service policy). When the consumer and utility agree on constraints and polices (service contract), the consumer can receive electricity using the service and the consumer can have payment sent (e.g. a cash payment or electronic funds transfer) to the utility (reachability).

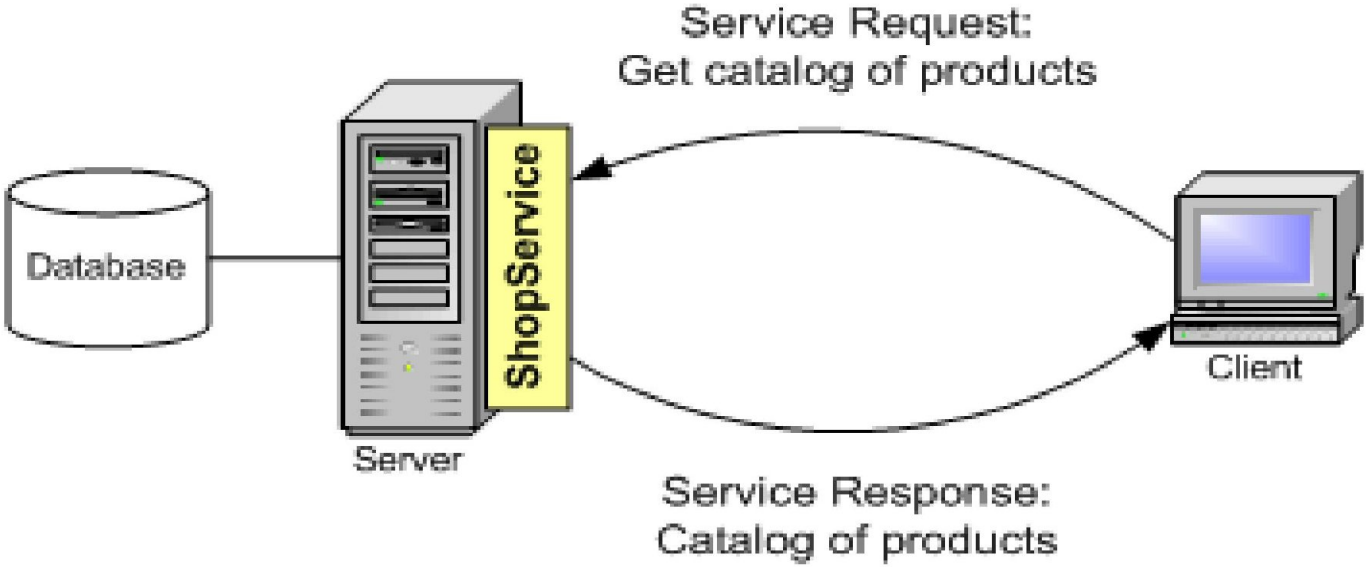
What is Web Services?

- "Software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts" – W3C Web Services Architecture Requirements, Oct. 2002
- "Programmable application logic accessible using Standard Internet Protocols…" – Microsoft
- "An interface that describes a collection of operations that are network accessible through standardized XML messaging …" – IBM
- "Software components that can be spontaneously discovered, combined, and recombined to provide a solution to the user's problem/request … " - SUN

User's View

- Software components designed to provide specific operations ("services") accessible using standard Internet technology
- Not tied to any one operating system or programming language
- Should be *self-describing:*
  - publish a public interface to the services
- Should be *discoverable*:
  - Mechanism for publishing what you have created
  - Can be found via a simple ***'find'*** mechanism
- Usually through SOAP (**Simple Object Access Protocol**) messages carrying XML documents, and a HTTP transport protocol or using REST (**REpresentational STate**)

Client Server Model



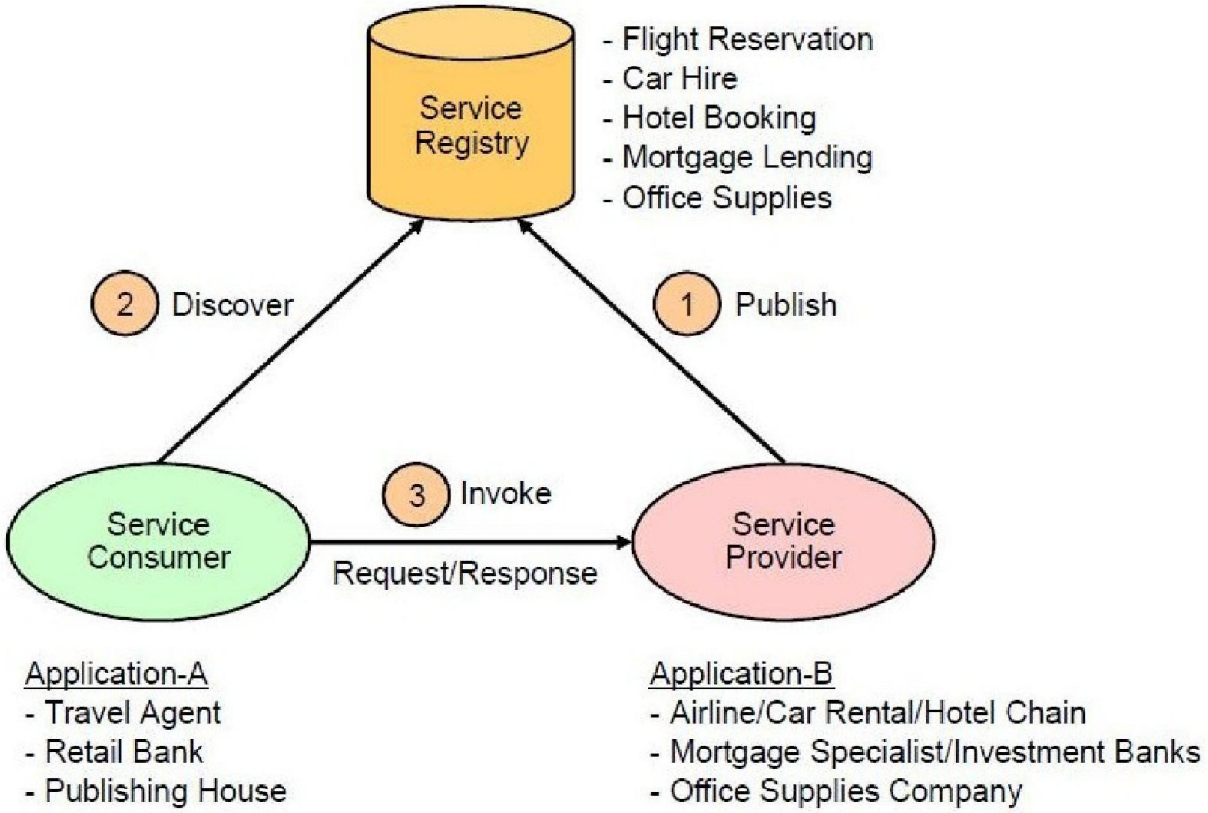
- Client needs to:
  - identify location of the required service
  - know how to communicate with the service to get what it required.

- Uses service registry - a third party

SOA model

- Services "published" in a Service registry.
- Service requestor asks Service registry to locate the service.
- Service requestor "binds" with service provider to invoke service.

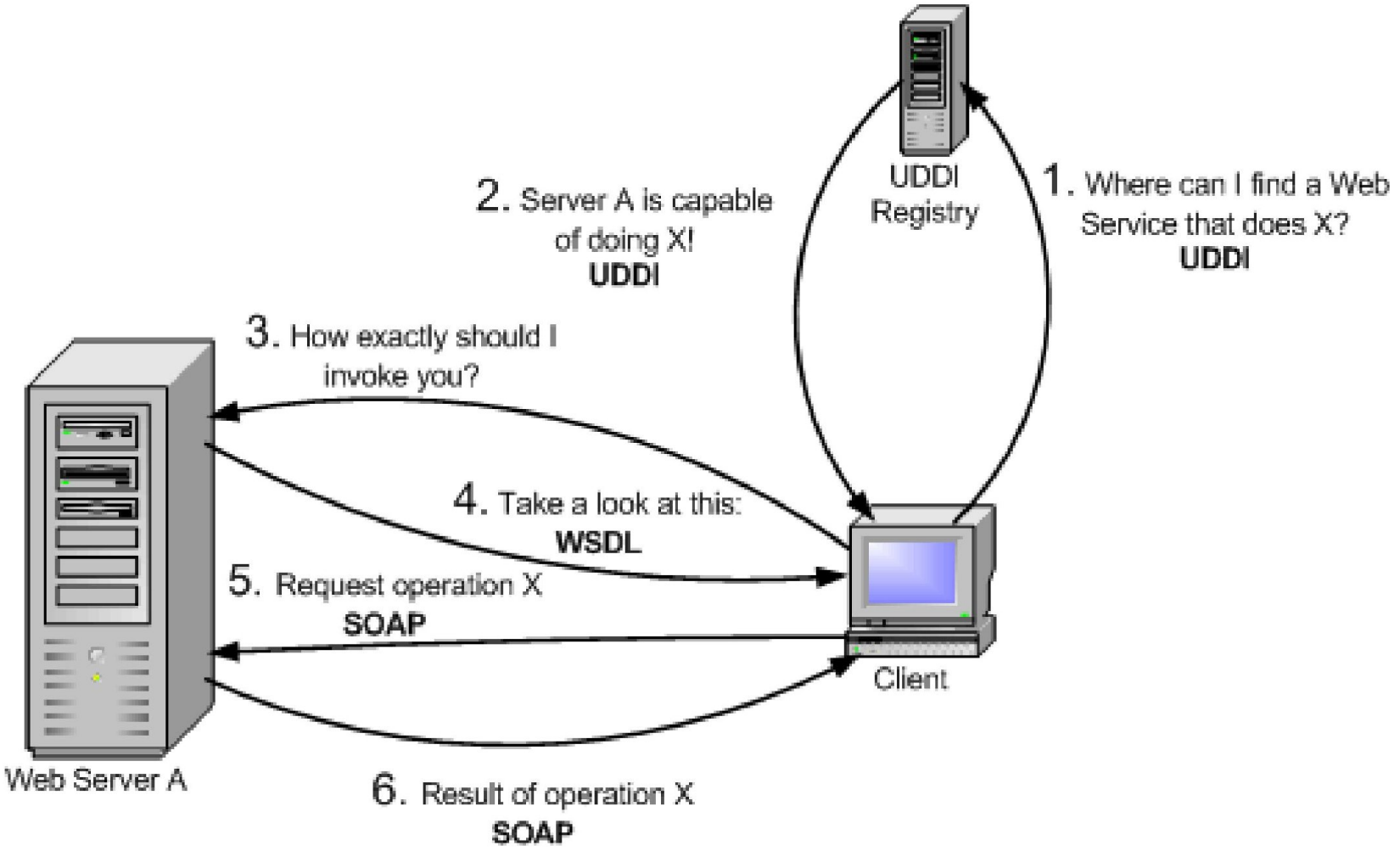
SOA Architectural model



Web Services - Stack

- Service providers publish the details of their services in the service registry using an Extensible Markup Language (XML) called Web Services Description Language (WSDL).
- Service requestors find the suitable services from the service registry using specifications such as Universal Description, Discovery, and Integration (UDDI).
- Service providers and service requestors communicate with each other using protocols such as Simple Object Access Protocol (SOAP).
- SOAP allows a program or service running on one platform to communicate with another program or service running on a different platform, using the Hypertext Transfer Protocol (HTTP) and its XML as the mechanisms for information exchange.

Web Services



SOA: Benefits

- SOA enables mutual data exchange between programs of different vendors without the need for additional programming or changes to the services.
- The services should be independent, and they should have standard interfaces that can be called to perform their tasks in a standard way.
- A service need not have prior knowledge of the calling application, and the application does not need to have knowledge about how the tasks are performed by a service.

- Reuse of services: results in lower development/ maintenance costs and in quicker time to market.
- 2. Agility: through the wide use of standards such as web services, the ability to change the business processes quickly when needed to support the change in the business activities.
- 3. Monitoring: It helps to monitor the performance of various services to make the required changes.
- 4. Extended reach: In the collaboration between enterprises or in the case of shared processes, it is the ability to get the service of various other processes for completing a particular task.

Similarities – SOA and Cloud

- both rely on the service concept to achieve the objectives
- SOA and cloud computing use service delegation in that the required task is delegated either to service provider (in the case of cloud computing) or to other application or business components in the enterprise (in the case of SOA).
- both cloud computing and SOA promote loose coupling among the components or services, which ensures the minimum dependencies among different parts of the system.

Differences: SOA and Cloud

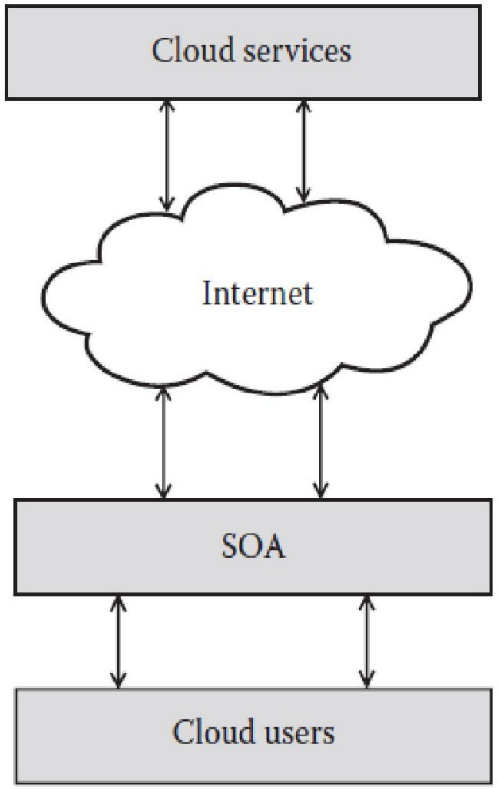
- The services in SOA mainly focus on business. Each service in SOA may represent one aspect of the business process.
- The services could be combined together to provide the required complete business application or business solution. Hence, in this sense, the services are horizontal.
- Various services in cloud computing are usually layered such as infrastructure, platform, or software, and the lower layer services support the upper services to deliver applications. Hence, the services in this case are vertical

- SOA is used for defining the application architecture. The various components or services of the application are divided based on their roles in the SOA applications. That means the solution for a business problem could be achieved by combining the various abstract services performing the required functions. The services in the SOA can be reused by other applications.
- Cloud computing is a mechanism for delivering IT services. The various services can be divided or grouped based on their roles such as infrastructure, platform, or software. The services in this case could also be reused by other applications.

How SOA meets Cloud

- The web services standards (WS*) used in SOA are also used in the cloud computing domain for solving various issues, such as asynchronous messaging, metadata exchange, and event handling.
- SOA is an architecture, and cloud computing is an instance of architecture or an architectural option, not an architecture by itself.
- When used with cloud computing, SOA helps to deliver IT resources as a service over the Internet, and to mix and match the resources to meet the business requirements.
- SOA using cloud computing architecture provides the agility in such a way that it could easily be changed to incorporate the business needs since it uses services that are configured through a configuration or process layer.

Convergence



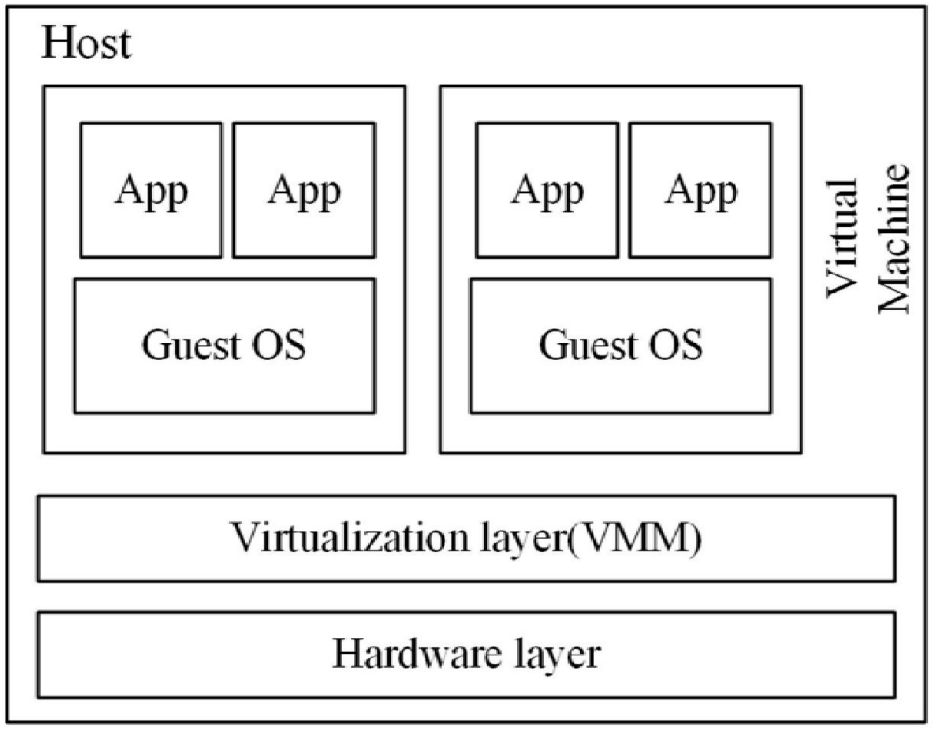
Cloud Computing Simulation tools

Components of Cloud Computing

Host: Server that **accepts connections from clients** who request a service function.

A cloud host is a server that provides services to customers via **multiple connected servers** that comprise a cloud.

A virtual machine is an **emulated computer system** created using software.



Simulation Tools

What is Simulation?

Without creating an actual structure we see the results by modeling a dummy structure having same parameters.

Why Simulation is Required?

Cloud computing requires a large amount of infrastructure and huge amount of money is required for it.

To know the feasibility of cloud for a particular application we simulate it.

Cloud Computing Simulation Softwares/Tools

Open Source Cloud Computing Simulators

- GreenCloud
  - Simulation environment for **energy-aware** cloud computing data centres.
  - Modelling of the **energy consumed by the data centre's** IT equipment
- iCanCloud
  - Simulation platform supports the simulation of **large storage networks**
  - Predicting the trade-offs between cost and performance of a given set of applications executed in specific hardware.

- EMUSIM
  - Integrated Emulation and Simulation(EMUSIM).
  - **Combines emulation and Simulation** (CloudSim) to enable more accurate models
- GroudSim
  - Designed for **scientific applications** on grid and cloud environments.
- DCSim (Data Centre Simulation)
  - Framework for performing **high-end experiments on data centre** management techniques
- CloudAnalyst
  - **GUI based simulator** derived on the bases of CloudSim
  - Evaluation of social network tools according to the **geographical distribution of users and data centres**.

CloudSim Tool

CloudSim is a simulation toolkit that supports the **modeling and simulation of the core functionality of cloud**.

Developed in the CLOUDS Laboratory, at the Computer Science and Software Engineering Department of the University of Melbourne.

Provides environment to

- Test application services in a **repeatable** and **controllable** environment.
- Tune the **system bottlenecks** before deploying apps in an actual cloud.
- Experiment with **different workload mix** and resource performance scenarios.

Features

- Large scale Cloud computing data centers
- Virtualized server hosts, with customizable policies for provisioning host resources to virtual machines
- Application containers
- Energy-aware computational resources
- Data center network topologies and message-passing applications
- Dynamic insertion of simulation elements, stop and resume of simulation
- User-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines

Limitations

- Not Suitable for real-time applications, security algorithms, platform implementations, etc
- No Graphical User Interface (GUI) is available

Frequently used for

- Load Balancing of resources and tasks
- Task scheduling and its migrations
- Optimizing the Virtual machine allocation and placement policies
- Energy-aware Consolidations or Migrations of virtual machines
- Optimizing schemes for Network latencies for various cloud scenarios

Versions of CloudSim

- CloudSim 1.0 beta
- CloudSim 2.X
  - CloudSim 2.0
  - CloudSim 2.1
  - CloudSim 2.1.1
- CloudSim 3.X
  - CloudSim 3.0.1
  - CloudSim 3.0.2
  - CloudSim 3.0.3
- CloudSim 4.0
- CloudSim 5.0 (Latest release)

Versions of CloudSim

CloudSim 1.0 to CloudSim 2.0 : New Simulation Core, Improvement In Schedulers

CloudSim 2.X to CloudSim 3.0 : New Vm Scheduler, New Datacenter Network Model, New Vm Allocation And Selection Policies Etc.

CloudSim 3.X to CloudSim 4.0 : Added support for Container virtualization, Bugfixes

CloudSim 4.0 to CloudSim 5.0 : VM extensions with performance monitoring, Work with other simulation models such as Software-defined Networks (SDN) / Service Function Chaining (SFC).

Other Related Tools

Cloudsim 3.x.x became base for many other tools

- CloudSimEx
  - Set of **extensions** for the CloudSim simulator
    - Web session modeling, Better logging utilities, Utilities for generating CSV files for statistical analysis, Automatic id generation, Utilities for running multiple experiments in parallel, MapReduce simulation.
- EdgeCloudSim
  - Simulation environment **specific to Edge Computing** scenarios
    - Experiments that considers both computational and networking resources
- WorkflowSim
  - Support of workflow preparation and execution
    - With an implementation of a stack of workflow parser, workflow engine and job scheduler.

- CloudReports
  - Graphic tool that **simulates distributed computing environments** based on the Cloud Computing paradigm
- CloudAnalyst
  - Evaluation of social networks tools according to **geographic distribution of users** and data centers.
- iFogSim
  - Enables modelling and **simulation of Fog computing** environments.
  - Evaluation of resource management and scheduling policies across edge and cloud resources under different scenarios

Classes of CloudSim

Datacenter

Models the **core infrastructure-level services**, that will consist set of Hosts which is responsible for managing VMs during their life cycle.

Host

Physical computing node in a Cloud with processing capabilities, memory, storage and scheduling policy for allocating processing cores to Virtual Machines

VM

Models a Virtual Machine. Host can simultaneously instantiate multiple VMs and allocate cores based on processor sharing policies(Space shared, Time shared).