Jaipur Engineering College & Research Centre, Jaipur

Notes

Cloud Computing

[6CS4-06]

Prepared By:
Suniti Chouhan
Abhishek Jain

VISION AND MISSION OF INSTITUTE

VISION

To become renowned centre of outcome based learning and work towards academic, professional, cultural and social enrichments of the lives of individual and communities"

MISSION

M1. Focus on evaluation of learning outcomes and motivate students to inculcate research aptitude by project based learning.

M2. Identify areas of focus and provide platform to gain knowledge and solutions based on informed perception of Indian, regional and global needs.

M3. Offer opportunities for interaction between academia and industry.

M4. Develop human potential to its fullest extent so that intellectually capable and imaginatively gifted leaders can emerge in a range of professions.

VISION AND MISSION OF DEPARTMENT

VISION

To become renowned Centre of excellence in computer science and engineering and make competent engineers & professionals with high ethical values prepared for lifelong learning.

MISSION

M1: To impart outcome based education for emerging technologies in the field of computer science and engineering.

M2: To provide opportunities for interaction between academia and industry.

M3: To provide platform for lifelong learning by accepting the change in technologies

M4: To develop aptitude of fulfilling social responsibilities.

COURSE OUTCOMES

CO1: Implement the cloud computing architecture i.e, the model, types of clouds, various service models and programming concepts.

CO2: Acquire knowledge about the recent trends in area of cloud computing like Hadoop, programming of Google app engine and virtualization technology and resource management.

CO3: Identify the various threats related to cloud and as well as disaster recovery related to same.

CO4: Analyze the cloud platforms in IT industry and various case studies on the industries providing cloud services.

Program Outcomes (PO)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Educational Objectives (PEO)

1. To provide students with the fundamentals of Engineering Sciences with more emphasis in Computer Science & Engineering by way of analyzing and exploiting Engineering challenge

2. To train students with good scientific and engineering knowledge so as to comprehend, analyze, design, and create novel products and solutions for the real life problems.

3. To inculcate professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, entrepreneurial thinking and an ability to relate engineering issues with social issues.

4. To provide students with an academic environment aware of excellence, leadership, written ethical codes and guidelines, and the self-motivated life-long learning needed for a successful professional career.

5. To prepare students to excel in Industry and Higher education by Educating Students along with High moral values and Knowledge.

MAPPING CO-PO

Cos/POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	3	3	3	2	2	2	1	2	2	3
CO2	3	3	3	3	3	2	2	2	1	2	2	3
CO3	3	3	3	3	2	2	2	2	2	2	2	3
CO4	3	3	3	3	2	2	2	2	2	2	2	3

PSO

PSO1: Ability to interpret and analyze network specific and cyber security issues, automation in real word environment.

PSO2: Ability to Design and Develop Mobile and Web-based applications under realistic constraints.

SYLLABUS

UNIT 1: Introduction: Objective, scope and outcome of the course.

UNIT 2: Introduction: Objective, scope and outcome of the course. Introduction Cloud Computing: Nutshell of cloud computing, Enabling Technology, Historical development, Vision, feature Characteristics and components of Cloud Computing. Challenges, Risks and Approaches of Migration into Cloud. Ethical Issue in Cloud Computing, Evaluating the Cloud's Business Impact and economics, Future of the cloud. Networking Support for Cloud Computing. Ubiquitous Cloud and the Internet of Things.

UNIT 3: Cloud Computing Architecture: Cloud Reference Model, Layer and Types of Clouds, Services models, Data centre Design and interconnection Network, Architectural design of Compute and Storage Clouds. Cloud Programming and Software: Fractures of cloud programming, Parallel and distributed programming paradigms-Map Reduce, Hadoop, High level Language for Cloud. Programming of Google App engine.

UNIT 4: Virtualization Technology: Definition, Understanding and Benefits of Virtualization. Implementation Level of Virtualization, Virtualization Structure/Tools and Mechanisms, Hypervisor VMware, KVM, Xen. Virtualization: of CPU, Memory, I/O Devices, Virtual Cluster and Resources Management, Virtualization of Server, Desktop, Network, and Virtualization of data-centre.

UNIT 5: Securing the Cloud: Cloud Information security fundamentals, Cloud security services, Design principles, Policy Implementation, Cloud Computing Security Challenges, Cloud Computing Security Architecture . Legal issues in cloud Computing. Data Security in Cloud: Business Continuity and Disaster Recovery , Risk Mitigation , Understanding and Identification of Threats in Cloud, SLA-Service Level Agreements, Trust Management

UNIT 6: Cloud Platforms in Industry: Amazon web services , Google AppEngine, Microsoft Azure Design, Aneka: Cloud Application Platform -Integration of Private and Public Clouds Cloud applications: Protein structure prediction, Data Analysis, Satellite Image Processing, CRM

Unit 2: Cloud Computing Architecture

The term cloud computing is a wide umbrella encompassing many different things; lately it has become a buzzword that is easily misused to revamp existing technologies and ideas for the public. What makes cloud computing so interesting to IT stakeholders and research practitioners? How does it introduce innovation into the field of distributed computing? This chapter addresses all these questions and characterizes the phenomenon. It provides a reference model that serves as a basis for discussion of cloud computing technologies.

Cloud services like virtual hardware, development platform or application software depend on a distributed infrastructure owned by the provider or rented from a third party. The characterization of a cloud is quite general as it can be implemented using a datacenter, a collection of clusters or a heterogeneous distributed system composed of desktop machines, workstations and servers. Generally, clouds are built by using one or more datacenters. Hardware resources are virtualized to provide isolation of workloads and to best exploit the infrastructure. The cloud computing paradigm developed by the union of various existing models, technologies, and concepts that changed the way we deliver and use IT services. The definition of this process is as follows:

Cloud computing is a utility-oriented and Internet-centric way of delivering IT services on demand and these services cover the entire computing stack from the hardware infrastructure packaged as a set of virtual machines to software services such as development platforms and distributed applications.

In this chapter we will discuss about the Cloud Reference Model and Layers, Types and service models of Clouds. We will also discuss about the Data Center Design and interconnection networks of cloud as well also about the architectural design of compute and storage clouds.

Cloud Reference Model

Cloud reference model includes the aspects like infrastructure, development platforms, application and services.

• Architecture:

o It is possible to organize all the existing realizations of cloud computing into a layered view covering the entire load from hardware appliances to software systems.

o Cloud resources are coupled together to offer “computing horsepower” which is required for providing services.

o Cloud infrastructure can be heterogeneous in nature because a variety of resources such as clusters and also networked PCs can be used to build it.

o The physical infrastructure is managed by the core middleware because the objectives of it is to provide an appropriate runtime environment for applications and to best utilize resources.

o Hardware virtualization technologies are used to guarantee runtime environment customization, application isolation, sandboxing and quality of service.

o Infrastructure management is the key function of core middleware which supports capabilities such as negotiation of the quality of service, admission control, execution management and monitoring, accounting, and billing.

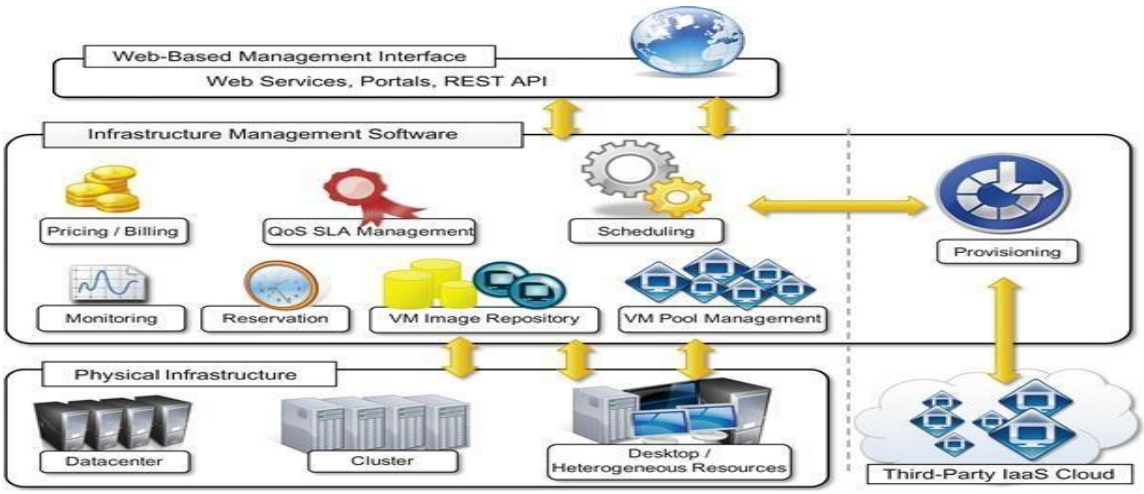


FIGURE 2.1 The cloud computing architecture.

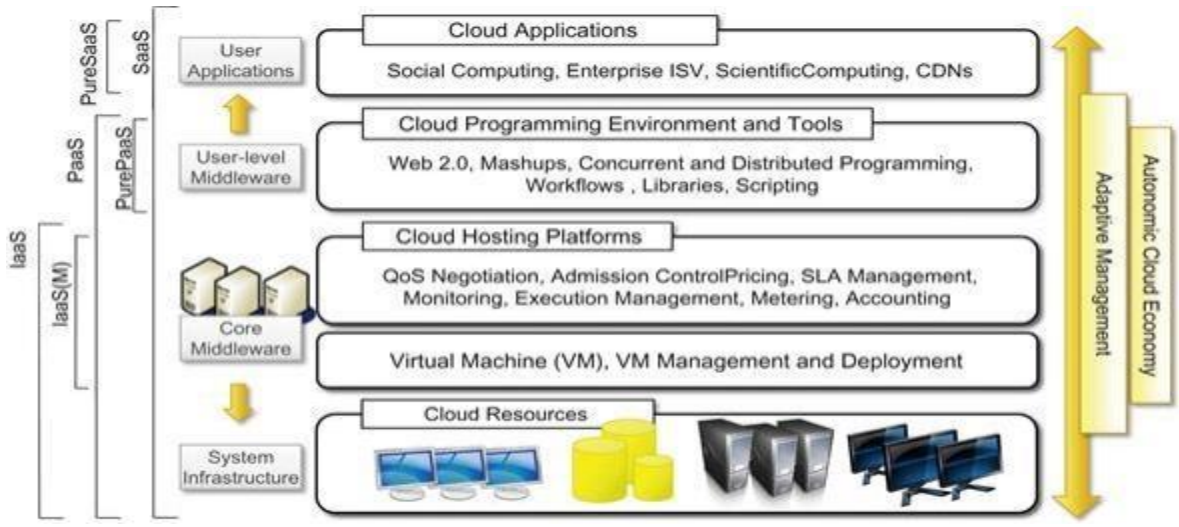


FIGURE 2.2 Infrastructure-as-a-Service reference implementation.

• **Infrastructure-as-a-Service (IaaS):** The combination of cloud hosting platforms and resources is generally classified as an Infrastructure-as-a-Service (IaaS) solution.

o Provide both the management layer and the physical infrastructure but others provide only the management layer and the management layer is integrated with IaaS to provide physical infrastructure.

o Suitable for designing the system infrastructure but provide limited services to build applications. Such services are provided by cloud programming environments and tools (Web-based interfaces, command-line tools, and frameworks) which form a new layer for offering users a development platform for applications.

Platform-as-a-Service (PaaS): users develop their applications specifically for the cloud by using the API at the user-level middleware. This method is also known as Platform-as-a- Service (PaaS) because the service offered to the user is a development platform rather than an infrastructure.

oPaaS solutions generally include the infrastructure which is bundled as part of the service provided to users.

o In the case of Pure PaaS only the user-level middleware is offered and it has to be complemented with a virtual or physical infrastructure.

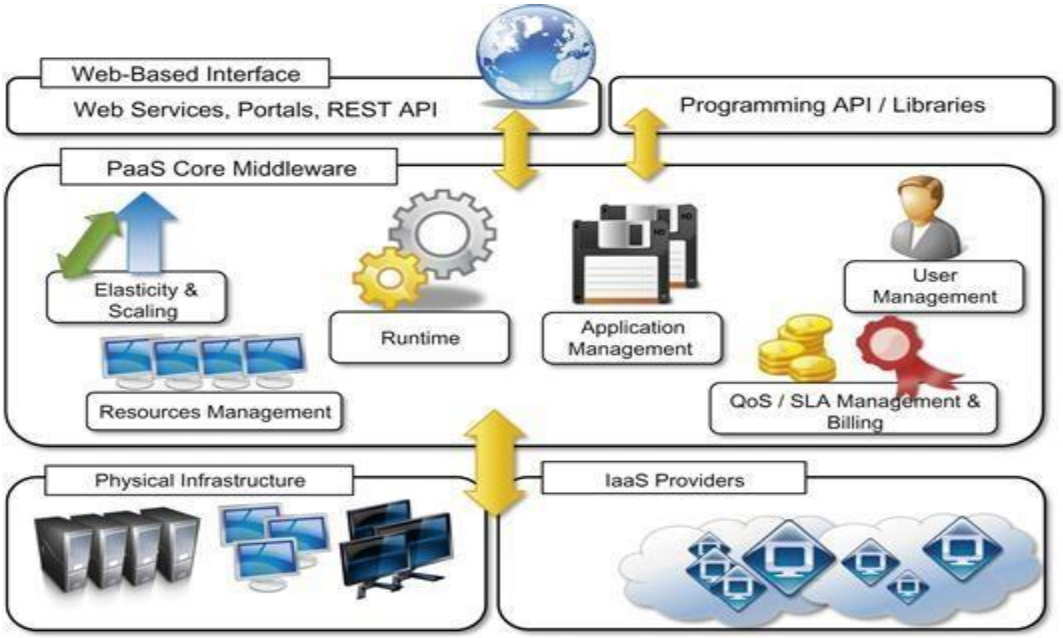


FIGURE 2.3 The Platform-as-a-Service reference model.

Software-as-a-Service (SaaS): It is the top layer of the reference model contains services delivered at the application level. These are Web-based applications that rely on the cloud to provide service to end users.

Applications: The horsepower of the cloud provided by IaaS and PaaS solutions allows independent software vendors to deliver their application services over the Internet.

o The reference model also introduces the concept of everything as a Service (XaaS). Cloud services from different providers can be combined to provide a completely integrated solution covering all the computing stack of a system.

oIaaS providers offer virtual machines where PaaS solutions are deployed. When there is no need for a PaaS layer it is possible to directly customize the virtual infrastructure with the software stack needed to run applications.

A distributed system composed of Web servers, database servers and load balancers on top of which pre-packaged software is installed to run Web applications. This possibility has made cloud computing an interesting option for reducing initial capital investment in IT.

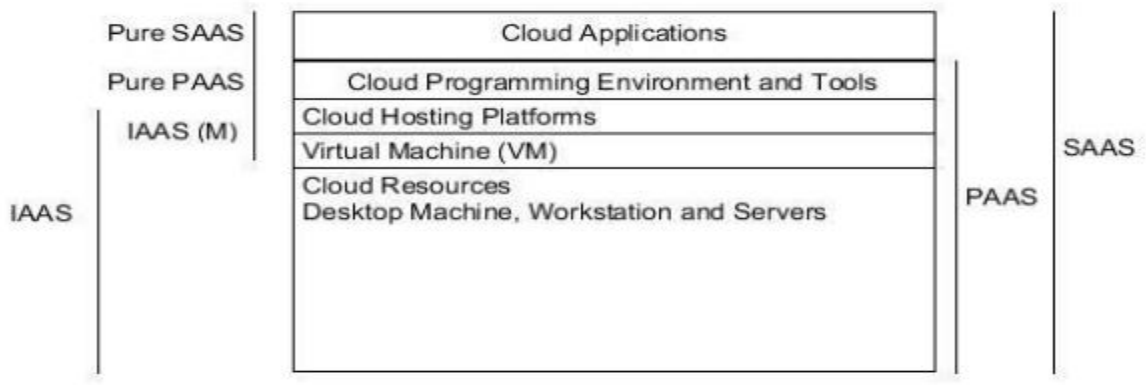


Fig 2.1: Cloud Reference Model

Layers and Types of Clouds:

Cloud Layers are divided into three classes as follows:

Infrastructure as a Service

Platform as a Service

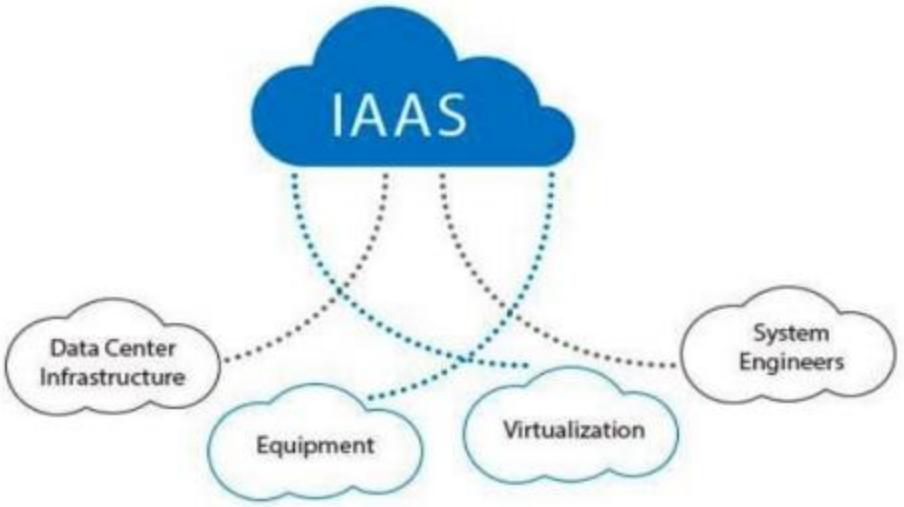
Software as a Service

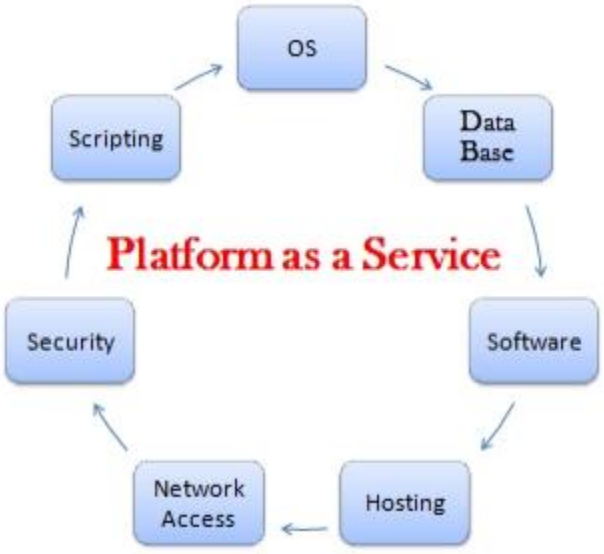
Infrastructure as a Service

o Offering virtualized resources (computation, storage, and communication) on demand is known as Infrastructure as a Service (IaaS).

o A cloud infrastructure enables on-demand provisioning of servers running several choices of operating systems and a customized software stack. Infrastructure services are considered to be the bottom layer of cloud computing systems.

o Users are given privileges to perform numerous activities to the server such as: starting and stopping it, customizing it by installing software packages, attaching virtual disks to it and configuring access permissions and firewalls rules.



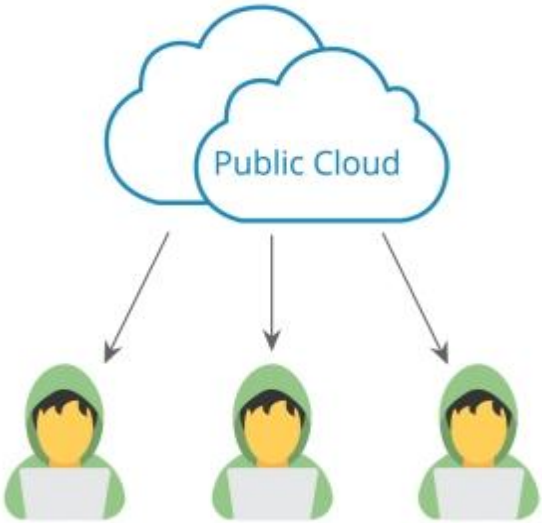


• Platform as a Service

o It is an approach to offer a higher level of abstraction to make a cloud easily programmable and it is known as Platform as a Service (PaaS).

o A cloud platform offers an environment on which developers create and deploy applications and do not necessarily need to know how many processors or how much memory that applications will be using.

o In addition, multiple programming models and specialized services (e.g., data access, authentication, and payments) are offered as building blocks to new applications.



• Software as a Service

o Applications reside on the top of the cloud stack.

o Services provided by this layer can be accessed by end users through Web portals. Therefore, consumers are increasingly shifting from locally installed computer programs to on-line software services that offer the same functionally.

o Traditional desktop applications such as word processing and spreadsheet can now be accessed as a service in the Web. This model of delivering applications, known as Software as a Service (SaaS), eases the burden of software maintenance for customers and simplifies development and testing for providers.

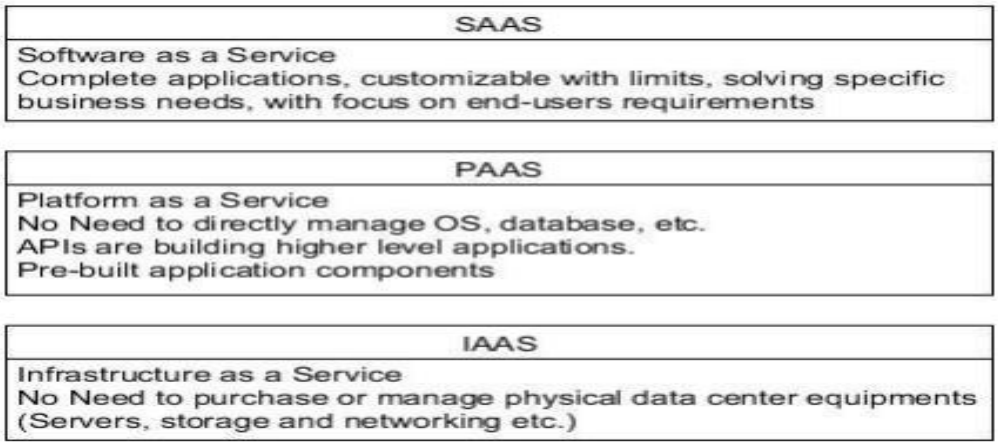


Fig 2.2: Layers of Cloud

Types of Cloud

Cloud types are estimated by the deployment models. According to the NIST cloud can be classified as public, private, community or hybrid based on model of deployment.

• Public cloud: The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

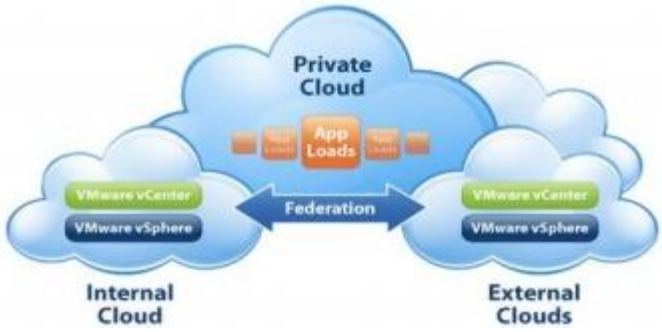
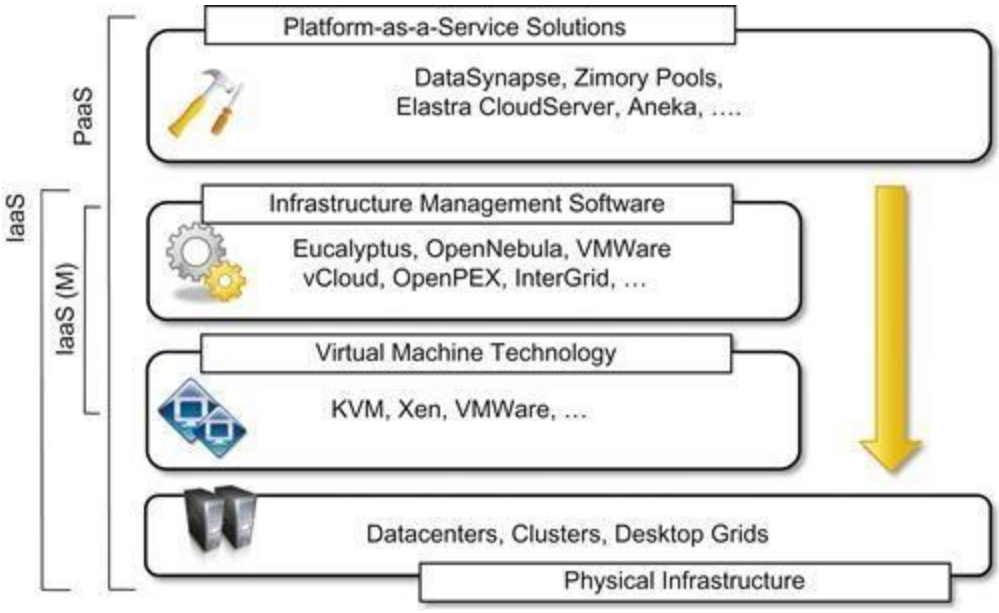


FIGURE 4.4 Private clouds hardware and software stack.

Private cloud: The cloud infrastructure is operated exclusively for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.



Hybrid cloud: The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability.

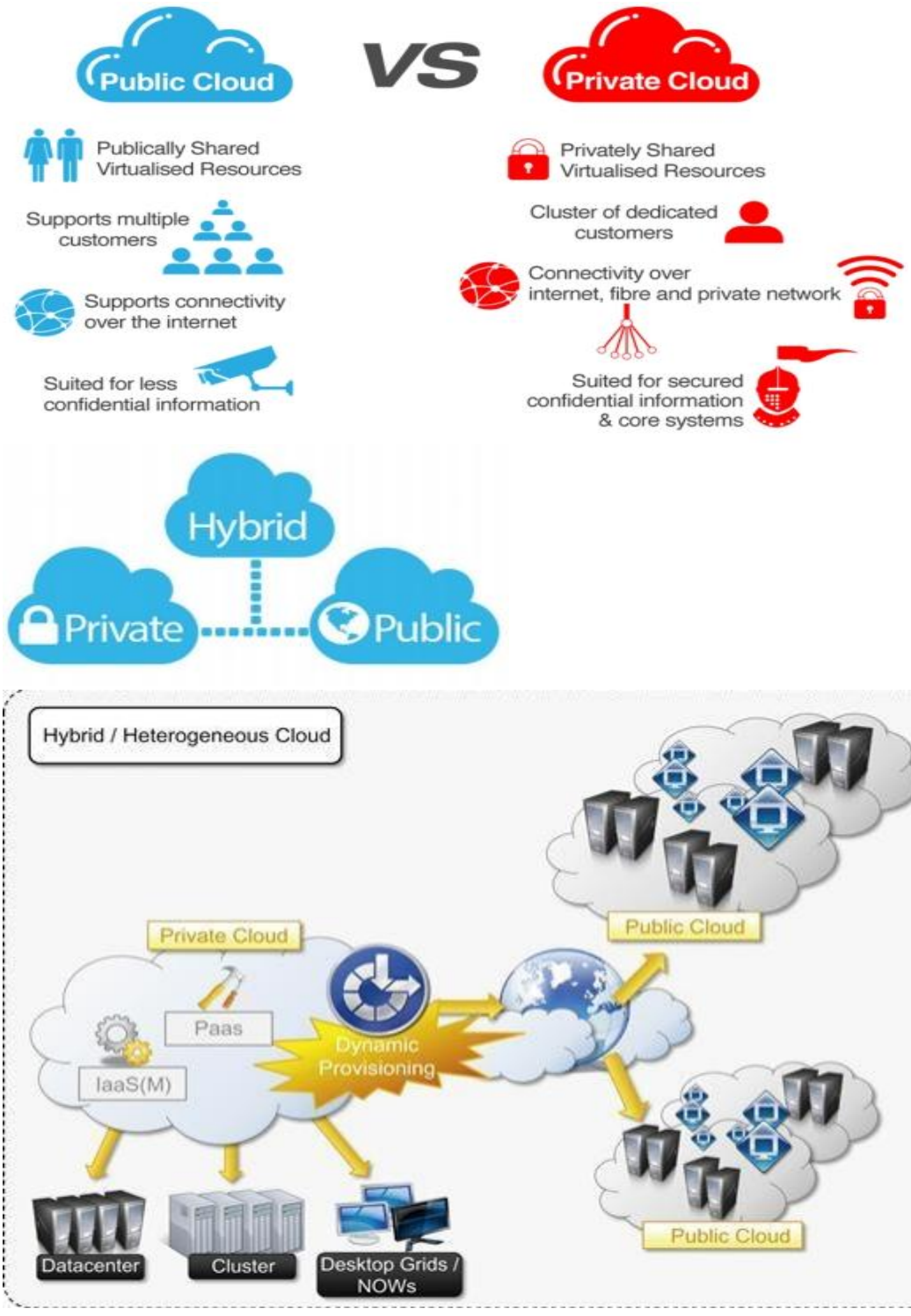


FIGURE 4.5 Hybrid/heterogeneous cloud overview.

Community cloud: The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns like 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.

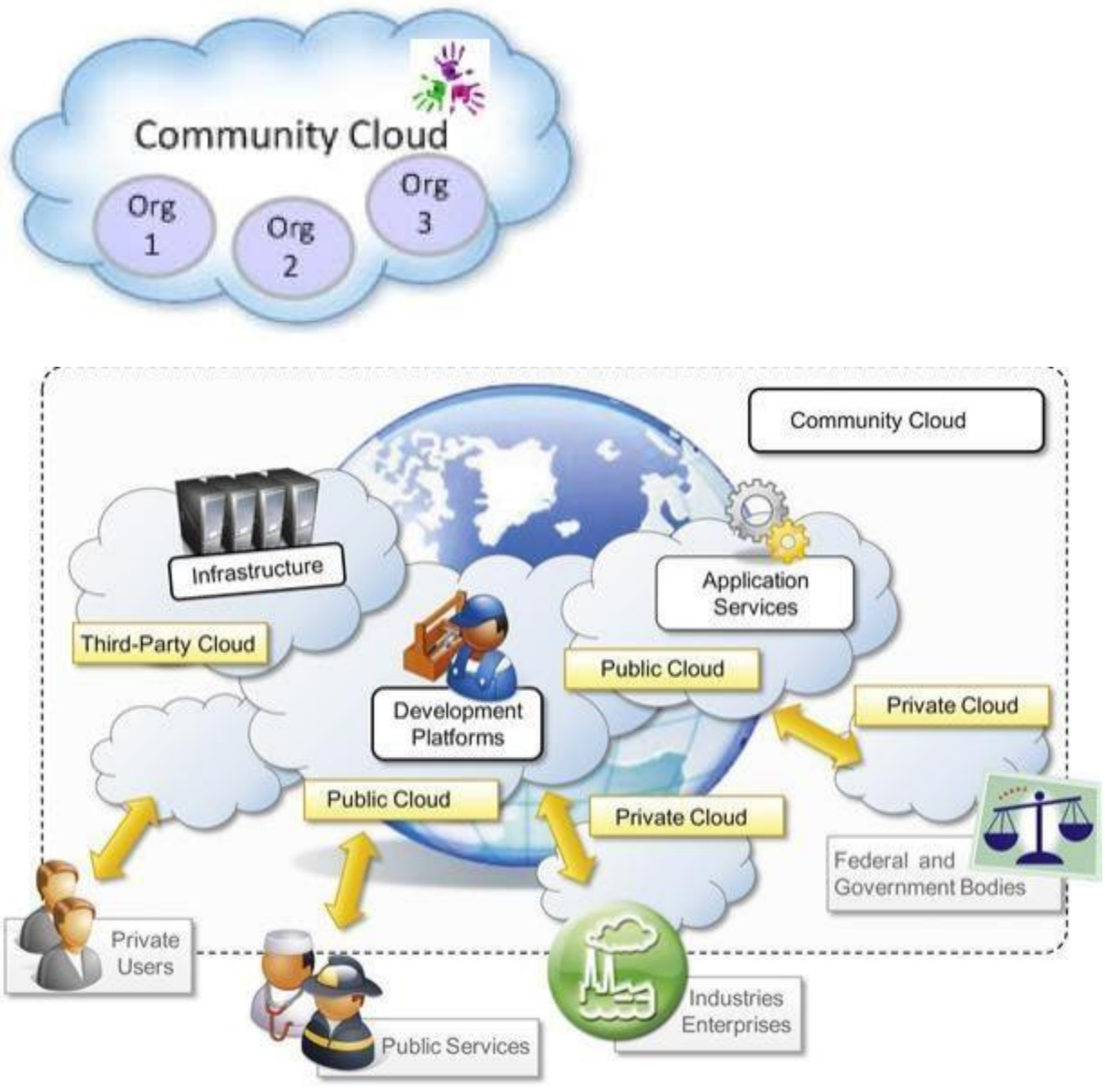


FIGURE 4.6 A community cloud.

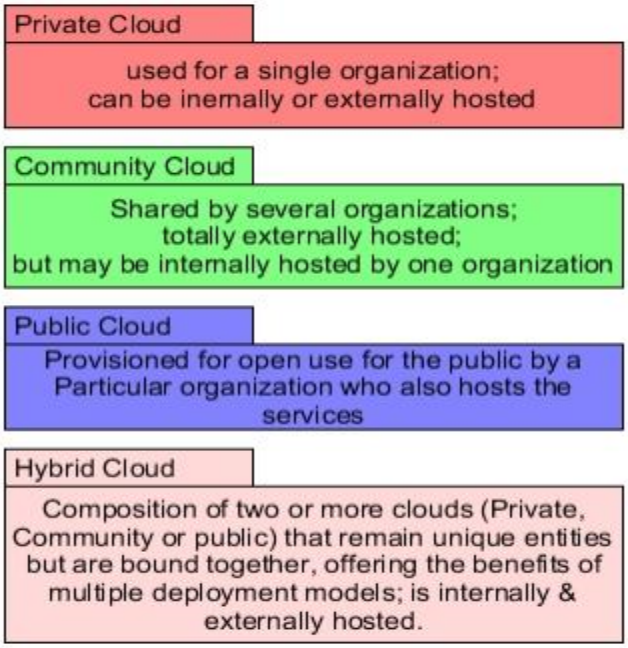


Fig 2.3: Types of Cloud

Services models: Cloud service provider's begins with the boundary between a client's network, management, and responsibilities. With the development of cloud computing different vendors comes in the market that have different services associated with them. The collection of services offered by them is called the service model.

Three service types have been accepted universally which are as such:

• **Infrastructure as a Service:**IaaS provides virtual machines, virtual storage, virtual infrastructure, and other hardware assets as resources that clients can run. The IaaS service provider manages all the infrastructure and the client is responsible for all other aspects of the deployment. This can include the operating system, applications, and user interactions with the system. Examples of IaaS service providers include:

o Amazon Elastic Compute Cloud (EC2)

o Eucalyptus

oGoGrid

oFlexiScale

oLinode

oRackSpace Cloud

oTerremark

• **Platform as a Service:**PaaS provides virtual machines, operating systems, applications, services, development frameworks, transactions, and control structures. The client can deploy its applications on the cloud infrastructure and use applications that were programmed using languages and tools that are supported by the PaaS service provider. The service provider manages the cloud infrastructure, the operating systems, and the enabling software. The client is responsible for installing and managing the application that they deployed. Examples of PaaS services are:

o Force.com

oGoGridCloudCenter

o Google AppEngine

o Windows Azure Platform

• **Software as a Service:** SaaS is a complete operating environment with applications, management, and the user interface. In the SaaS model the application is provided to the client through a thin client interface generally a web browser and the customer's responsibility begins and ends with entering and managing its data and user interaction. Everything from the application down to the infrastructure is the vendor's responsibility. Examples of SaaS cloud service providers are:

oGoogleApps

o Oracle On Demand

o SalesForce.com

o SQL Azure

When these three different service models taken together than it is known as the SPI model of cloud computing. Many other service models are as such:

StaaS, Storage as a Service

IdaaS, Identity as a Service

CmaaS, Compliance as a Service

Data Center Design and interconnection Networks:

A data centers are generally built by using a large number of servers over a huge interconnected network. Here, we will study about the design of large-scale data centers and small modular data

centers.

Data-Center Design

The cloud is built on huge datacenter as a shopping mall under one roof. This type of data center can house 400,000 to 1 million servers. The data centers are built on with the concept of lower unit cost for larger data centers. A small data center could have 1,000 servers. The larger the data center lower the operational cost. The network cost to function a small data center is about seven times greater and the storage cost is 5.7 times greater than the large data center.

Construction Requirements

Most data centers are built by using commercially available components.

A server consists of a processor sockets with a multicore CPU of locally shared internal cache hierarchy, coherent DRAM and a number of directly attached disk drives.The DRAM and disk resources within the rack are accessible through first-level rack switches and all resources in all racks are accessible through a cluster level switch.A large application must compact with large differences in latency, bandwidth and capacity. • In a very large scale data center components are relatively cheaper.The components used in data centers are very different from those in building supercomputer systems.With a scale of thousands of servers there may be concurrent or simultaneous failures like hardware failure or software failure of 1 percent of nodes in common.Many failures can happen in hardware. For example CPU failure, disk I/O failure and network failure.It is even fairly possible that the whole data center does not work in the case of a power crash.Also some failures are brought on by software.The service and data should not be lost in a failure situation.Thus the software must keep multiple copies of data in different locations and keep the data accessible while facing hardware or software errors.Reliability can be achieved by expendable hardware.

Cooling System

A data-center room consist of high floors for hiding cables, power lines and cooling supplies. The cooling system is simpler to the power system to some extent. The high floor has a steel grid resting on upright support of about 2 –4 ft above the concrete floor.

The under-floor area is frequently used to route power cables to racks but its primary use is to distribute cool air to the server rack. The CRAC (computer room air conditioning) unit forces the high floor addition containing gas at a higher pressure than the surrounding by blowing cold air into the session .The cold air outflows from the session over holed tiles that are placed in front of

server racks. Racks are arranged in long channel that alternate between cold passage and hot passage to avoid mixing hot and cold air.

The hot air produced by the servers circulates back to the intakes of the CRAC units that cool it and then exhaust the cool air into the high floor session again. Typically the incoming coolant is at 12–14°C and the warm coolant returns to a chiller. Newer data centers often insert a cooling tower to pre-cool the condenser water loop fluid. Water-based free cooling uses cooling towers in which water absorbs the coo lant's heat in a heat exchanger.

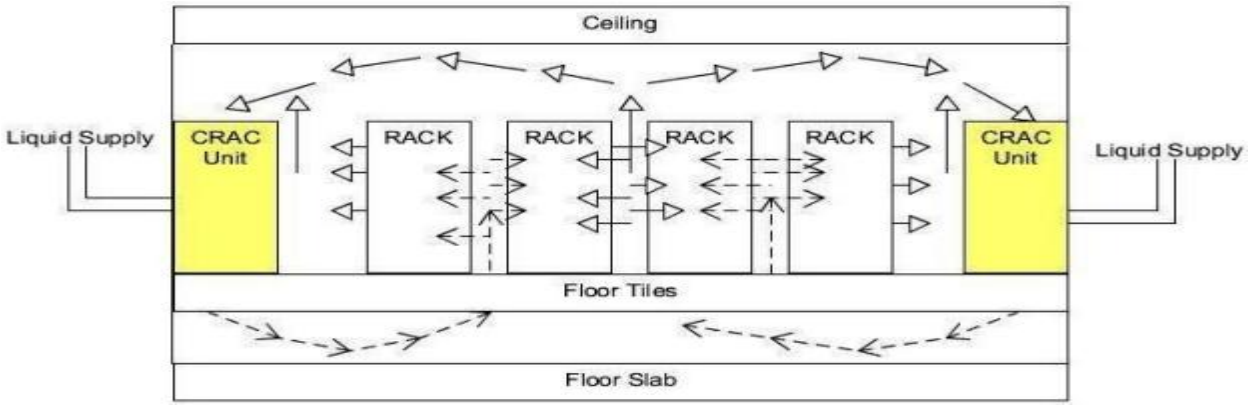


Fig 2.4: Data Center& It's Cooling Process

Data-Center Interconnection Networks:

A design of a data center is the interconnection network among all servers in the data center cluster. This design must run into five special requirements:

Low latency

High bandwidth

Low cost

Message passing interface (MPI) communication support

Fault tolerance

The design of an inter server network must satisfy both point-to-point and collective communication designs between all server nodes.

Application Traffic Support

The network topology should support all MPI communication patterns as both point-to-point and collective MPI communications. The network should have high division bandwidth to meet the requirements. One can use one or few servers as metadata master which need to communicate with slave server nodes in the cluster. The original network structure should support various

network traffic patterns required by user applications.

Network Expandability

The interconnection network should be expandable with thousands or even hundreds of thousands of server nodes. The cluster network interconnection should be allowed to increase more servers in the data center. The network topology must be restructured while facing such expected growth in the future. The network should be designed to support load balancing and data movement between the servers. None of the links should become a blockage that slows down application performance.

The fat tree and crossbar networks could be implemented with low cost Ethernet switches. The design could be very challenging when the number of servers increases suddenly. The most critical issue regarding expandability is support of modular network growth for building data-center containers. One single data center container contains hundreds of servers and is considered to be the building block of large scale data centers. The Cluster networks need to be designed for data center containers. Cable connections are then needed between multiple data center containers.

Data centers are not made by loading up servers in multiple racks now a days. Instead datacenter owners buy server containers while each container contains thousands of server nodes. The owners can just plug in the power supply outside connection link and cooling water to start the whole system. This is quite efficient and reduces the cost of purchasing and maintaining servers. One approach is to establish the connection support first and then extend the support links to reach the end servers. One can also connect multiple containers through external switching and cabling.

Fault Tolerance and Graceful Degradation

The interconnection network should provide some tool to bear link or switch failures. Multiple paths should be established between any two server nodes in a data center. Fault tolerance of servers is achieved by duplicating data and computing between redundant servers. Similar type of termination technology can be applied to the network structure to handle potential failures. In software termination the software layer should be aware of network failures. Packet forwarding should be avoided using broken links. The network support software drivers should handle this transparently without affecting cloud operations.

In case of failures the network structure should destroy gracefully during limited node failures and

hot swappable components are preferred. There should be no critical paths or critical points which may become a single point of failure that pulls down the entire system. Most design innovations are in the topology structure of the network. The network structure is often divided into two layers:

The lower layer close to the end servers

The upper layer to establish the backbone connections among the server groups or sub- clusters

This hierarchical interconnection approach requests to building data centers with modular containers.

witch-centric Data-Center Design here are two approaches to build data-center-scale networks:

• Switch-centric network: the switches are used to connect the server nodes. The switch-centric design does not affect the server side. No modifications to the servers are needed.

• Server centric design: modify the operating system running on the servers. Special drivers are designed for relaying the traffic. Switches still have to be organized to achieve the connections.

Container Data Center Construction

The container design includes the network, computer, storage and cooling gear. One needs to increase cooling efficiency by varying the water and airflow with better airflow management. Another concern is to meet regular load requirements. The construction may start with one server then move to a rack system design and finally to a container system. This staged development may take different amounts of time and demand increasing costs.

The container must be designed weatherproof and easy to transport. Modular data center construction and testing may take a few days to complete if all components are available. The modular data center approach supports many cloud service applications.

Data Center Management Issues

Here are basic requirements for managing the resources of a data center.

Making common users happy: The data center should be designed to provide quality service to the majority of users for at least 30 years.

Controlled information flow: Information flow should be streamlined. Sustained services and high availability are the primary goals.

Multiuser manageability: The system must be managed to support all functions of a data center including traffic flow, database updating and server maintenance.

Scalability to prepare for database growth: The system should allow growth as workload

increases. The storage, processing, I/O, power, and cooling subsystems should be scalable.

Reliability in virtualized infrastructure: Failover, fault tolerance, and VM live migration should be integrated to enable recovery of critical applications from failures or disasters.

Low cost to both users and providers: The cost to users and providers of the cloud system built over the data centers should be reduced, including all operational costs.

Security enforcement and data protection: Data privacy and security defense mechanisms must be deployed to protect the data center against network attacks and system interrupts and to maintain data integrity from network attacks.

Green information technology: Saving power consumption and upgrading energy efficiency are in high demand when designing and operating current and future data centers.

Architectural design of Compute and Storage Clouds:

Cloud architecture used to process massive amounts of data with a high degree of parallelism. In this section virtualization support, resource provisioning, infrastructure management, and performance modeling are discussed in detail for architecture design and storage.

A Generic Cloud Architecture Design:

Internet cloud is projected as a public cluster of servers provisioned on demand to perform collective web services or distributed applications using data-center resources.

• Cloud Platform Design Goals

o Scalability, virtualization, efficiency and reliability are four major design goals of a cloud computing platform and cloud supports Web 2.0 applications. Cloud management receives the user request to find the correct resources and then calls the provisioning services which appeal for the resources in the cloud.

o The cloud management software needs to support both physical and virtual machines. Security in shared resources and shared access of data centers is another design challenge.

o The platform needs to establish a very large-scale infrastructure. The hardware and software systems are combined to make it easy and efficient to operate.

o This architecture is system scalable. If one service takes a lot of processing power, storage capacity or network traffic than it is simple to add more servers and bandwidth.

o Another benefit is system reliability. Data can be put into multiple locations. For example, user e-mail can be put in three disks which expand to different geographically separate data centers. In such a situation if one of the data centers crashes than the user data is still accessible.

• Enabling Technologies for Clouds

o Cloud users are capable to demand more capacity at highest demand, reduce costs, experiment with new services and remove unneeded capacity. Whereas service providers can increase system utilization by multiplexing, virtualization, and dynamic resource provisioning.

o These technologies play vital role in making cloud computing a reality. In the hardware area the rapid progress in multicore CPUs, memory chips and disk arrays has made possible to build faster data centers with huge amounts of storage space. Resource virtualization supports quick cloud deployment and disaster recovery. Service-oriented architecture (SOA) used to plays a vital role in cloud development.

o Now a days clouds are designed to serve a large number of renters over massive volumes of data. The availability of large-scale distributed storage systems is the foundation of modern data centers. Cloud computing is greatly benefitted by the progress made in license management and automatic billing techniques.

• A Generic Cloud Architecture

o The Internet cloud is projected as a massive cluster of servers. These servers are provisioned on demand to perform collective web services or distributed applications using data-center resources.

o The cloud platform is formed dynamically by provisioning or de-provisioning servers, software and database resources. Servers in the cloud may be physical machines or VMs. User interfaces are applied to request services.

o The cloud platform demands distributed storage and supplementary services. The cloud computing resources are built into the data centers which are typically owned and operated by a third-party provider.

o In cloud software's are converted in to a service. The cloud demands a high degree of trust from huge amount of data saved on large data centers.

o Framework is to build to process large-scale data stored in the storage system. This demands a distributed file system over the database system.

o Other cloud resources are to be added into a cloud platform including storage area networks, database systems, firewalls and security devices.

o Web service providers offer special APIs that facilitate developers to exploit Internet clouds.

o The software infrastructure of a cloud platform need to handle all resource management and do most of the maintenance automatically. Software must detect the status of each node server

joining and leaving and to perform relevant tasks accordingly.

o The cloud physical platform developers are more concerned about the performance/price ratio and reliability issues than cutting speed performance.

o Private clouds are easier to manage and public clouds are easier to access. But developers used to develop hybrid cloud because many cloud applications requirement go further than the boundary of an intranet.

o Security is a critical issue in safeguarding the operation of all cloud types.

Layered Cloud Architectural Development:

• The layered cloud architecture is developed using three layers infrastructure, platform, and application.

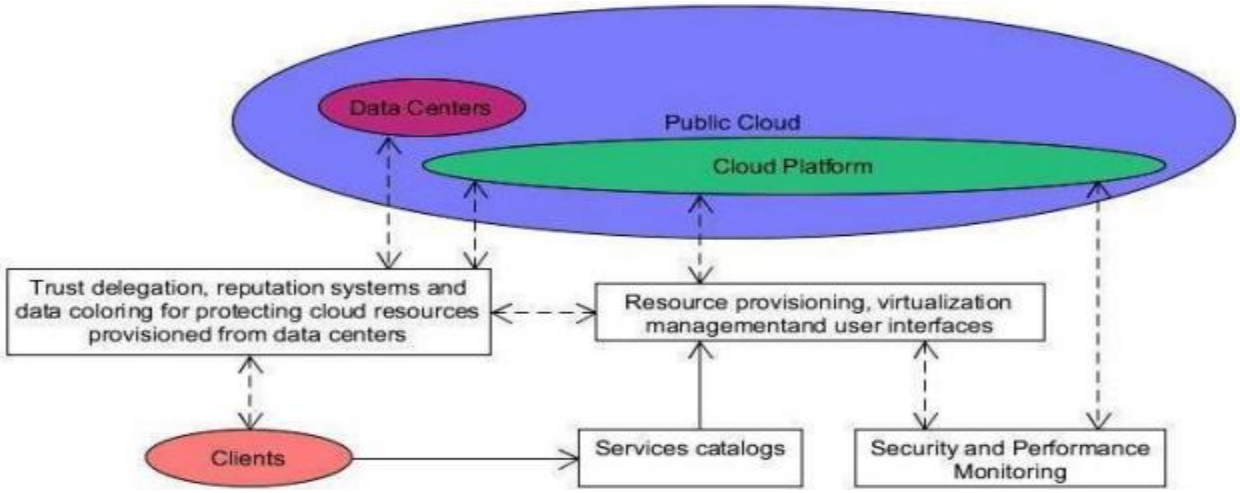


Fig 2.5: Generic Cloud Architecture

These three development layers are implemented with virtualization and standardization of hardware and software resources provisioned in the cloud.

The services of public, private and hybrid clouds are sent to users through networking support over the Internet and intranets involved.

The infrastructure layer is built with virtualized compute, storage and network resources. The generalization of hardware resources is meant to provide the flexibility to users. Internally virtualization provides automated provisioning of resources and optimizes the infrastructure management process.

The platform layer is for general purpose and repeated usage of the collection of software resources. This layer provides users with an environment to develop their applications to test operation flows and to monitor execution results and performance.

The application layer is designed with a collection of all needed software modules for SaaS applications. Service applications in this layer works as daily office management work and it is also used by enterprises in business marketing and sales, consumer relationship management (CRM), financial transactions and supply chain management.

Market-Oriented Cloud Architecture:

Cloud providers design different QoS parameters for every consumer as required by consumer in SLAs. To complete the design providers cannot deploy traditional system-centric resource

management architecture. But instead of that market-oriented resource management is used to fix the supply and demand of cloud resources market equilibrium between supply and demand. The designer needs to provide feedback on economic reasons of both consumers and providers. To promote QoS based resource allocation mechanisms.

Clients get benefit by the reduction in cost given by providers and which could lead more competition in the market and thus lower prices. Such type of clouds are basically built by using the following things: Users submit service requests from anywhere in the world to the data center and cloud to be processed. The SLA resource allocator acts as the interface between the data center, cloud service provider and external users.

The mechanisms to support SLA-oriented resource management is as follow:

When a service request is first submitted the service request examiner interprets the submitted request for QoS requirements before determining whether to accept or reject the request.

The request examiner ensures that there is no overloading of resources.

.It also needs the latest status information regarding resource availability and workload processing in order to make resource allocation decisions effectively.

Then it assigns requests to VMs and determines resource entitlements for allocated VMs. The Pricing mechanism decides how service requests are charged.

Submission time.

Pricing rates.

Availability of resources.

Architectural Design Challenges:

There are six open challenges in cloud architecture development and all these challenges are as such given below:

Service Availability and Data Lock-in Problem

Data Privacy and Security Concerns

Unpredictable Performance and Bottlenecks

Distributed Storage and Widespread Software Bugs

Cloud Scalability, Interoperability, and Standardization

Software Licensing and Reputation Sharing

Cloud Programming and Software

In this section we have discussed about programming real cloud platforms & software like MapReduce, Hadoop and Google App Engine. We have used real service examples to explain the implementation and application requirements in the cloud. We also reviewed about the Fractures in cloud programming & Parallel and distributed programming paradigm.

Fractures of cloud programming:

The fractures for developing a healthy and dependable cloud programming environment are as following with the solution for removing these fractures:

Use virtual clustering to achieve dynamic resource provisioning with minimum overhead cost.

Use stable and persistent data storage with fast queries for information retrieval.

Use special APIs for authenticating users and sending e-mail using commercial accounts.

Cloud resources are accessed with security protocols such as HTTPS and SSL.

Fine-grained access control is desired to protect data integrity and deter intruders or hackers. • Shared data sets are protected from malicious alteration, deletion, or copyright violations.

Features are included for availability enhancement and disaster recovery with life migration of VMs.

Use a reputation system to protect data centers& this system only authorizes trusted clients and stops pirates.

Parallel and distributed programming Paradigms:

Consider a distributed computing system consisting of a set of networked nodes or workers. The system issues for running a typical parallel program are:

• Partitioning: This is applicable to both computation and data as follows:

o Computation partitioning: Computation partitioning splits a given job or a program into smaller tasks. It significantly depends on properly identifying portions of the job or program that can be implemented concurrently. On identifying parallelism in the structure of the program than it can be divided into parts to be run on different workers. Different parts also process different data or a copy of the same data.

o Data partitioning: Data partitioning splits the input or intermediate data into smaller pieces. On identification of parallelism in the input data than it can also be divided into pieces to be processed on different workers. Data pieces also be processed by different parts of a program or a copy of the same program.

Mapping: Mapping assigns either the smaller parts of a program or the smaller pieces of data to

original resources. This process aims to properly assign parts or pieces to be run simultaneously on different workers and is commonly handled by resource allocators in the system.

Synchronization: It is necessary because different workers used to perform different tasks like synchronization and coordination between workers so that race conditions are prevented and data dependency between different workers is properly managed. Multiple accesses to a shared resource by different workers might raise race conditions. Data dependency take place when a worker needs the processed data of other workers.

Communication: Data dependency is one of the main reasons for communication among workers. So the communication is always triggered when the intermediate data is sent to workers.

Scheduling: When the number of computation parts of tasks or data pieces of the job or program are more than the number of available workers than scheduler selects tasks for the workers. Resource allocator does the actual mapping of the computation. While the scheduler only picks the next part from the queue of unassigned tasks based on a set of rules called the scheduling policy. For multiple jobs or programs scheduler selects a sequence of jobs or programs for running over the distributed computing system. Scheduling is also necessary when system resources are not sufficient to run multiple job all together.

Motivation for Programming Paradigms:

Handling the whole data flow of parallel and distributed programming is very time consuming and it requires specific knowledge of programming. Dealing with these issues might affect the productivity of the programmer. It also distract the programmer from focusing on the logic of the program. So parallel and distributed programming paradigms are offered to abstract many parts of the data flow from the users.

These models targeted to provide users with an abstraction layer to hide implementation details of the data flow. Simplicity of writing parallel programs is important for parallel and distributed programming paradigms.

Some more motivational points on parallel and distributed programming paradigms are:

Improvement in the productivity of programmers

Decreasing the program execution time

Holding of the resources with extra efficiently

Increasing the throughput of the system

Support for the higher levels of abstraction.

MapReduce and Hadoop are the example of the recently proposed parallel and distributed programming models. They are developed for information retrieval from the applications but have been exposed as the applicable. The loose coupling of components in these paradigms makes them fit for VM implementation and provides better fault tolerance and scalability for some applications than traditional parallel computing models such as MPI.

MapReduce:

MapReduce is a software framework which supports parallel and distributed computing on large data sets.

This software framework summaries the data flow of a program running in a parallel on a distributed computing system by providing users two functions: Map and Reduce. Users can dominate these two functions to interact and manipulate the data flow for running their programs.

The MapReduce software framework provides an abstraction layer with the data flow and control flow to users and hides the implementation process of all data flow steps such as data partitioning, mapping, synchronization, communication, and scheduling.

The MapReduce takes an important parameter which is a specification object "Spec". This object is first initialized inside the user's program and then the user writes code to fill it with the names of input and output files as well as other free tuning parameters. This object is also filled with the name of the Map and Reduce functions to add these functions to the MapReduce library.

The overall structure of a user's program containing the MapReduce and the Main functions is given below. The Map and Reduce are two major subroutines. They will be called to implement the desired function performed in the main program.

```
Map Function (….)
{
… …
}
Reduce Function (….)
{
… …
}
Main Function (….)
{
```

```
Initialize Spec object
… …
MapReduce (Spec, & Results)
}
```

MapReduce Logical View:

The Map and Reduce functions of MapReduce are both defined as the data structured in (key, value) pairs. Map takes one pair of data with a type in one data domain and returns a list of pairs in a different domain:

Map(k1,v1) → list(k2,v2)

The Map function is functional in parallel to each pair in the input dataset and produces a list of pairs for each call. After that MapReduce framework collects all pairs with the same key from all lists and groups them together and create one group for each key.

Then Reduce function is applied in parallel to each group to produces a collection of values in the same domain:

Reduce(k2, list (v2)) → list(v3)

Each Reduce call typically produces either one value v3 or an empty return. So one call is allowed to return more than one value. The returns of all calls are collected as the wanted result list.

MapReduce framework transforms a list of (key, value) pairs into a list of values. This process is different from the typical functional programming map and reduce combination which accepts a list of arbitrary values and returns one single value that combines all the values returned by map. It is necessary but not sufficient to implementation the map and reduce abstractions in order to implement MapReduce. Distributed implementations of MapReduce require a means of connecting the processes which are performing the Map and Reduce.

Examples:

The prototypical MapReduce example counts the appearance of each word in a set of documents:

```
function map(String name, String document):
// name: name of document
// document: contents of document
for each word w in document:
```

```
emit (w, 1)
function reduce(String word, Iterator partialCounts):
// word: a word
// partialCounts: a list of aggregated partial counts
sum = 0
for each pc in partialCounts:
sum += ParseInt(pc)
emit (word, sum)
```

In the above example each document is divided into words and each word is counted by the map function using the word as the result key. The framework puts together all the pairs with the same key and feeds them to the same call to reduce. So this function needs to sum all of its input values to find the total appearances of that word.

MapReduce Data and Control Flow:

The Data and Control used to flow in MapReduce by following these functions:

Input reader: The input reader divides the input into parts called splits and the framework assigns one split to each Map function. It reads data from distributed file system and generates key/value pairs. Example: Read a directory full of text files and return each line as a record.

Map function: The Map function takes a series of key/value pairs to processes each to generate zero or more output key/value pairs. The I/O of the map are may be different from each other. If the application is doing a word count the map function than it would break the line into words and output a key/value pair for each word. Each output pair would contain the word as the key and the no. of instances of that word in the line as the value.

Partition function: The partition function is given the key with the number of reducers and it returns the index of the desired reducer. And the key is hashed to modulo the number of reducers using hash value. Partition function gives an approximately uniform distribution of data per horizontal partition for load-balancing purposes else the MapReduce operation may be waiting for slow reducers to finish the process. The data is exchanged between the Map and Reduce to move the data from the map node to the fragment where it will be reduced.

Comparison function: The input for each Reduce is dragged from the machine where the Map works and it is sorted using the application's comparison function.

Reduce function: The Reduce function is called once by framework for each unique key in the sorted order. The Reduce can repeat the values that are associated with that key and produce zero or more outputs.

Output writer: The Output Writer writes the output of the Reduce to the stable storage as usually in the distributed file system.

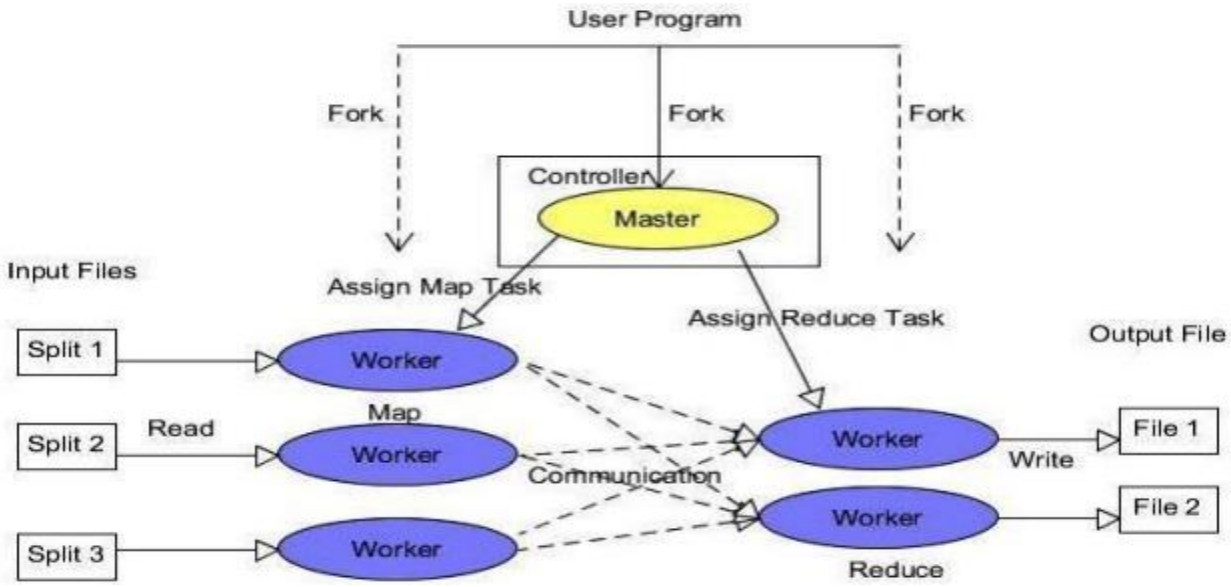


Fig 3.1: MapReduce Data and Control Flow

Map Reduce Benefits:

The MapReduce used to provide the following Benefits:

• Simplicity:

o Developers can write applications in their own language such as Java, C++ or Python. oMapReduce jobs are easy to run.

Scalability :MapReduce can process petabytes of data on one cluster.

Speed: Provides parallel processing means that it can solve the problem in hours or minutes. • Built-in recovery: It handles the failures when a machine with one copy of the data is unavailable and another machine has a copy of the same key/value pair which can be used to solve the same task.

Minimal data motion : It significantly reduces the network I/O load and keeps the I/O on the local disk or within the same rack.

Freedom to focus on the business logic : It maintains the routine details of deployment, resource management, monitoring and scheduling.

Hadoop:

Apache has developed Hadoop is an open source in JAVA for the implementation of MapReduce and it uses the Hadoop Distributed File System (HDFS). The Hadoop is divided into two fundamental layers one MapReduce engine and other HDFS.

Here are the details of these two fundamental layers.

HDFS Architecture: HDFS is a distributed file system that organizes files and stores their data on a distributed computing system. It has a master/slave architecture which contains a single Node as master and rest Data Nodes as workers. To store file it splits the file into fixed size blocks of around 64 MB and stores them on workers Data Nodes. The mapping between blocks with Data Nodes is maintained by the master. The master node also manages the file system's metadata.

HDFS Features: It needs some special requirements such as performance, scalability, concurrency control, fault tolerance and security requirements to operate efficiently. It only executes specific types of applications.

HDFS Fault Tolerance: It is designed to deploy on low cost hardware and hardware failure in this system is common. So Hadoop considers the following issues:

o Block replication: To consistently store data file blocks are simulated in this system. It stores a file as a set of blocks and each block is simulated and distributed on the full system. The

simulation factor is usually set by the user but it is '3' by default.

o Block report messages: Block reports are the messages sent to the master Node by other Data Node. Receipt of a report indicates that other Data Node are working correctly. Block report holds a list of all blocks in a Data Node.

• HDFS Operation: The role of master and slave nodes in managing operation is represented by the HDFS operations write and read.

o Reading a file: To read a file user sends an 'open' request to the master Node to get the location of file blocks. For each file block the master Node provides the address of the slave Nodes. After receiving information user calls the 'read' function to connect to the nearby slave Node which is containing the first block of the file. After that connection is terminated and the same process is repeated for all blocks of the requested file.

o Writing to a file: To write a file user sends a 'create' request to the master Node to create a new file in the file system. If file is not exists then the master Node informs the user to start writing data to the file by calling the 'write ' function. The very first block of the file is to be written to an internal data queue even though data streamer monitors writing process.

Running a Job in Hadoop:

These three components are needed for running a job in Hadoop:

User node

Job Tracker

Task Trackers

The data flow starts by calling the 'runJob(conf)' function in user program which is running on the user node. 'conf' contains some implementation factors of the MapReduce framework and HDFS.

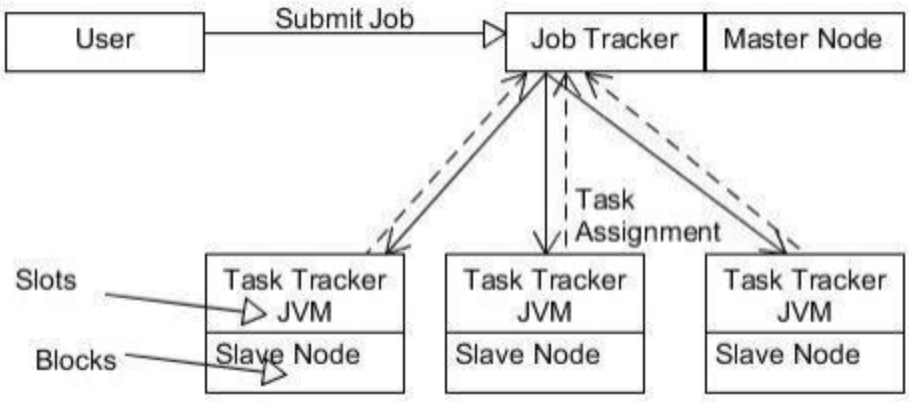


Fig 3.2: Data Flow in running the Hadoop

• Job Submission: Jobs has to be submitted from a user node to the Job Tracker node:

o A slave node asks for a new job ID from the Job Tracker and computes input file splits. o The slave node copies some resources like job's JAR file, configuration file and

computed input splits for the Job Tracker's file system.

o Slave node submits the job to the Job Tracker by calling the 'submitJob()' function.

• Task assignment: Job Tracker performs the localization operation on the data before assigning the tasks to the Task Trackers. The Job Tracker also creates reduce tasks and assigns them to the Task Trackers.

Task execution: The control of flow to execute task in Task Tracker is done by copying the job JAR file to its file system. Instructions are given in the job JAR file and are executed after launching Java Virtual Machine (JVM) to run reduce task.

Task running check: A task running check is done by after receiving messages by the Job Tracker and which are send from the Task Trackers. Each message informs Job Tracker that the sending Task Tracker is working properly and also informs Task Tracker is ready to run a new task.

High level Language for Cloud:

High Level Languages for Cloud are mainly developed by the Google and Yahoo. In this section we will discuss about the High level languages available in the market for cloud.

Sawzall:

Sawzall is a high level language which is written by Google for MapReduce framework.

Sawzall is a scripting language that can perform parallel data processing.

Sawzall provides fault-tolerates for processing of very large scale data which is collected from the Internet

It was developed by Rob Pike for processing Google's log files.

It has recently been released as an open source project.

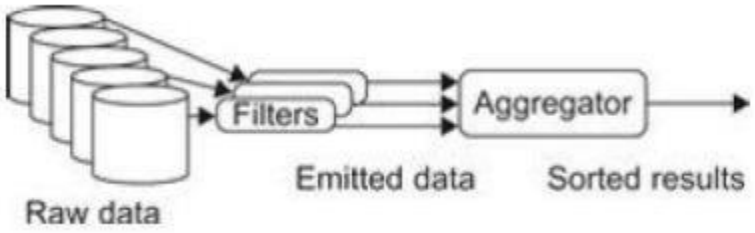


Fig 3.3: Flow of filtering, aggregating and collating in Sawzall

First the data is partitioned and processed locally with an on-site processing script.

The local data is filtered to get the necessary information.

The aggregator is used to get the final results based on the emitted data.

The Sawzall runtime engine translates the matching scripts for MapReduce.

The Sawzall program joins the power of cluster computing and also achieved reliability from redundant servers.

Program in Sawzall:

It is a simple program written in Sawzall for data processing on clusters:

Suppose a set of files with records is processed in each file. Each record contains one floating point number. We want to calculate the number of records, the sum of the values and the sum of the squares of the values.