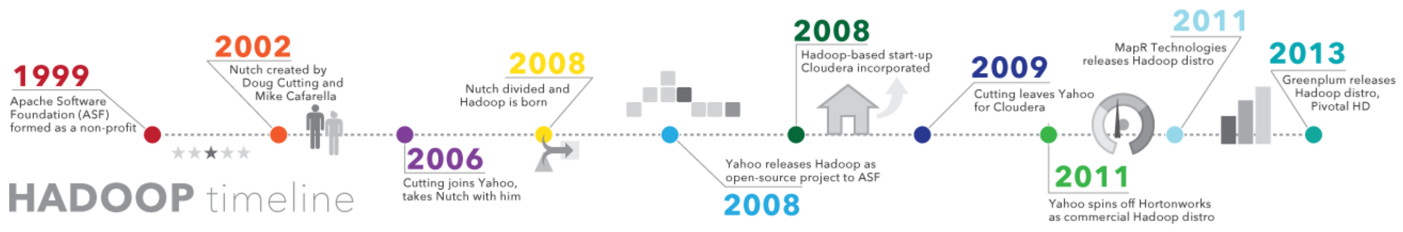
UNIT 5 NOTES CLOUD COMPUTING

Hadoop

Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

Hadoop History

As the World Wide Web grew in the late 1900s and early 2000s, search engines and indexes were created to help locate relevant information amid the text-based content. In the early years, search results were returned by humans. But as the web grew from dozens to millions of pages, automation was needed. Web crawlers were created, many as university-led research projects, and search engine start-ups took off (Yahoo, AltaVista, etc.).


1999
Apache Software Foundation (ASF) formed as a non-profit
2002
Nutch created by Doug Cutting and Mike Cafarella
HADOOP timeline
2006
Cutting joins Yahoo, takes Nutch with him
2008
Nutch divided and Hadoop is born
2008
Yahoo releases Hadoop as open-source project to ASF
2008
Hadoop-based start-up Cloudera incorporated
2009
Cutting leaves Yahoo for Cloudera
2011
Yahoo spins off Hortonworks as commercial Hadoop distro
2011
MapR Technologies releases Hadoop distro
2013
Greenplum releases Hadoop distro, Pivotal HD

One such project was an open-source web search engine called Nutch – the brainchild of Doug Cutting and Mike Cafarella. They wanted to return web search results faster by distributing data and calculations across different computers so multiple tasks could be accomplished simultaneously. During this time, another search engine project called Google was in progress. It was based on the same concept – storing and processing data in a distributed, automated way so that relevant web search results could be returned faster.

In 2006, Cutting joined Yahoo and took with him the Nutch project as well as ideas based on Google's early work with automating distributed data storage and processing. The Nutch project was divided – the web crawler portion remained as Nutch and the distributed computing and processing portion became Hadoop (named after Cutting's son's toy elephant). In 2008, Yahoo released Hadoop as an open-source project. Today, Hadoop's framework and ecosystem of technologies are managed and maintained by the non-profit Apache Software Foundation (ASF), a global community of software developers and contributors.

Why is Hadoop important? (FEATURES OF HADOOP/ ADVANTAGES OF HADOOP)

- **Ability to store and process huge amounts of any kind of data, quickly.** With data volumes and varieties constantly increasing, especially from
- **Computing power.** Hadoop's distributed computing model social media and the Internet of Things (IoT), that's a key consideration.processes big data fast. The more computing nodes you use, the more processing power you have.
- **Fault tolerance.** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically.
- **Flexibility.** Unlike traditional relational databases, you don't have to preprocess data before storing it. You can store as much data as you want and decide how to use it later. That includes unstructured data like text, images and videos.
- **Low cost.** The open-source framework is free and uses commodity hardware to store large quantities of data.
- **Scalability.** You can easily grow your system to handle more data simply by adding nodes. Little administration is required.

MODULES OF HADOOP

The base Apache Hadoop framework is composed of the following modules:

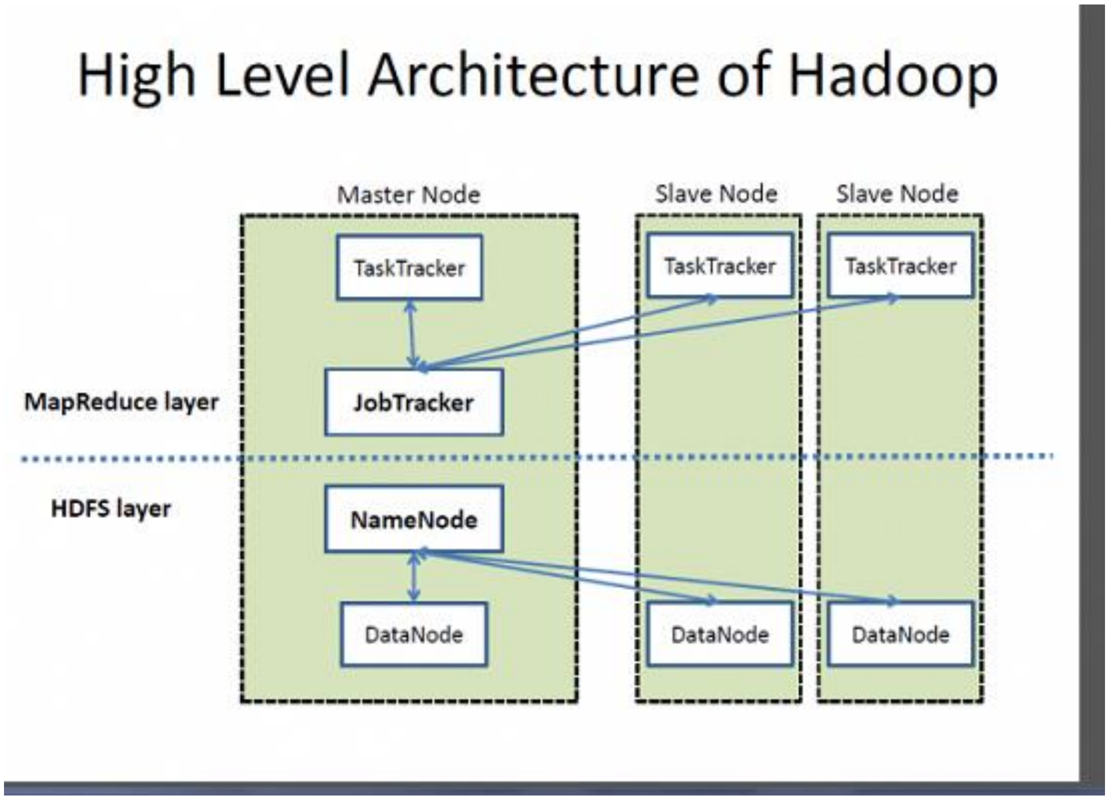
- *Hadoop Common* – contains libraries and utilities needed by other Hadoop modules;
- *Hadoop Distributed File System (HDFS)* – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- *Hadoop YARN* – (introduced in 2012) a platform responsible for managing computing resources in clusters and using them for scheduling users' applications;[10][11]
- *Hadoop MapReduce* – an implementation of the MapReduce programming model for large-scale data processing.

Hadoop Architecture

The 3 important hadoop components that play a vital role in the Hadoop architecture are -

i. Hadoop Distributed File System (HDFS) – Patterned after the UNIX file system
ii. Hadoop MapReduce
iii. Yet Another Resource Negotiator (YARN)

Hadoop follows a master slave architecture design for data storage and distributed data processing using HDFS and MapReduce respectively. The master node for data storage is hadoop HDFS is the NameNode and the master node for parallel processing of data using Hadoop MapReduce is the Job Tracker. The slave nodes in the hadoop architecture are the other machines in the Hadoop cluster which store data and perform complex computations. Every slave node has a Task Tracker daemon and a DataNode that synchronizes the processes with the Job Tracker and NameNode respectively. In Hadoop architectural implementation the master or slave systems can be setup in the cloud or on-premise.



Role of Distributed Storage - HDFS in Hadoop Application Architecture Implementation

A file on HDFS is split into multiple bocks and each is replicated within the Hadoop cluster. A block on HDFS is a blob of data within the underlying file system with a default size of 64MB.The size of a block can be extended up to 256 MB based on the requirements.

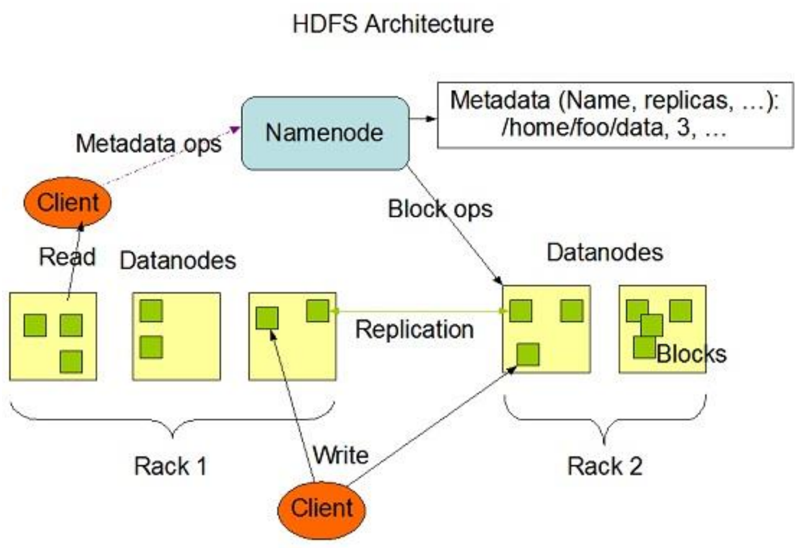


Image Credit :Apache.org

Hadoop Distributed File System (HDFS) stores the application data and file system metadata separately on dedicated servers. NameNode and DataNode are the two critical components of the Hadoop HDFS architecture. Application data is stored on servers referred to as DataNodes and file system metadata is stored on servers referred to as NameNode. HDFS replicates the file content on multiple DataNodes based on the replication factor to ensure reliability of data. The NameNode and DataNode communicate with each other using TCP based protocols. For the Hadoop architecture to be performance efficient, HDFS must satisfy certain pre-requisites –

- All the hard drives should have a high throughput.
- Good network speed to manage intermediate data transfer and block replications.

NameNode

All the files and directories in the HDFS namespace are represented on the NameNode by Inodes that contain various attributes like permissions, modification timestamp, disk space quota, namespace quota and access times. NameNode maps the entire file system structure into memory. Two files fsimage and edits are used for persistence during restarts.

- Fsimage file contains the Inodes and the list of blocks which define the metadata.It has a complete snapshot of the file systems metadata at any given point of time.
- The edits file contains any modifications that have been performed on the content of the fsimage file.Incremental changes like renaming or appending data to the file are stored in the edit log to ensure durability instead of creating a new fsimage snapshot everytime the namespace is being altered.

When the NameNode starts, fsimage file is loaded and then the contents of the edits file are applied to recover the latest state of the file system. The only problem with this is that over the time the edits file grows and consumes all the disk space resulting in slowing down the restart process. If the hadoop cluster has not been restarted for months together then there will be a huge downtime as the size of the edits file will be increase. This is when Secondary NameNode comes to the rescue. Secondary NameNode gets the fsimage and edits log from the primary NameNode at regular intervals and loads both the fsimage and edit logs file to the main memory by applying each operation from edits log file to fsimage. Secondary NameNode copies the new fsimage file to the primary NameNode and also will update the modified time of the fsimage file to fstime file to track when then fsimage file has been updated.

DataNode

DataNode manages the state of an HDFS node and interacts with the blocks .A DataNode can perform CPU intensive jobs like semantic and language analysis, statistics and machine learning tasks, and I/O intensive jobs like clustering, data import, data export, search, decompression, and indexing. A DataNode needs lot of I/O for data processing and transfer.

On startup every DataNode connects to the NameNode and performs a handshake to verify the namespace ID and the software version of the DataNode. If either of them does not match then the DataNode shuts down automatically. A DataNode verifies the block replicas in its ownership by sending a block report to the NameNode. As soon as the DataNode registers, the first block report is sent. DataNode sends heartbeat to the NameNode every 3 seconds to confirm that the DataNode is operating and the block replicas it hosts are available.

What is MapReduce?

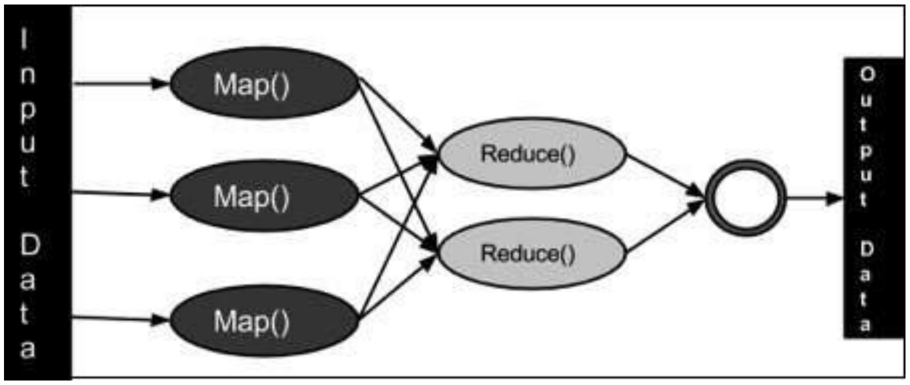
MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
  - **Map stage** − The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
  - **Reduce stage** − This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Google App Engine

- Google App Engine is a Platform as a Service (PaaS) product that provides Web app developers and enterprises with access to Google's scalable hosting and tier 1 Internet service.
- The App Engine requires that apps be written in Java or Python, store data in Google BigTable and use the Google query language. Non-compliant applications require modification to use App Engine.
- Google App Engine provides more infrastructure than other scalable hosting services such as Amazon Elastic Compute Cloud (EC2). The App Engine also eliminates some system administration and developmental tasks to make it easier to write scalable applications.
- Google App Engine is free up to a certain amount of resource usage. Users exceeding the per-day or per-minute usage rates for CPU resources, storage, number of API calls or requests and concurrent requests can pay for more of these resources.

Advantages of GAE

- The main advantage of GAE is it doesn’t scale. However, even after you empower billing, the whole system is augmented to support only 500 requests per second. If you want more, you can reach Google’s disposal to increase your thresholds, so you can have millions of users, but more than 500 requests per second. If you want to do more than 500 requests and your app is legit, they will remove those caps or make them higher.

- GAE feature set is good enough to build a decent website and you don't need to do the maintenance work.
- It doesn't require any server administration. It has free usage quotas and provides scalability. GAE has better access to Google user accounts and deployment process is very easy.
- GAE has the highest admin load. Once you are set up, deploying and re-deploying is quick and they will auto-everything. For example, you don't have to worry about the number of servers your app is using, how to share the data and how to load-balance.
- You can get any feature from the store with GAE. But with Azure, you get the feeling of SQL Azure database but at the same time it would be too expensive. Azure storage is likely to have more gotchas. No relational integrity, no order-by, you will scam with the in-memory setting more. GAE's store has far rarer margins and more features than Azure tables.

Disadvantages of GAE

- GAE is yet not stable enough. Even Google says that GAE is the same infrastructure of Google self-internal project. And the budget would increase much when your website becomes bulky.
- Without native file system read/write access, it is hard to process some data transform with existing library, and it doesn't support some native file system base library as well.
- It does not provide full text search API. Also, the SDK/Java is unfavorable with Maven, it is unsatisfactory to accomplish lots of external libraries. The SDK/Java depth rest on on IDE, and the default project directory structure is some difference with normal web app.
- It is not easy to process unit test. It cannot fix the root cause and does not support add SSL to web site. The GAE may be the development for future web application, but it is not equipped for building a modern web site now.
- It suffers from the inability to tweak server software. The Filesystem and many standard library modules are inaccessible. Only Python and a few runs of Java Virtual machine.

OpenStack

OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds. Backed by some of the biggest companies in software development and hosting, as well as thousands of individual community members, many think that OpenStack is the future of cloud computing. OpenStack is managed by the OpenStack Foundation, a non-profit that oversees both development and community-building around the project.

OpenStack lets users deploy virtual machines and other instances that handle different tasks for managing a cloud environment on the fly. It makes horizontal scaling easy, which means that tasks that benefit from running concurrently can easily serve more or fewer users on the fly by just spinning up more instances. For example, a mobile application that needs to communicate with a remote server might be able to divide the work of communicating with each user across many different instances, all communicating with one another but scaling quickly and easily as the application gains more users.

And most importantly, OpenStack is open source software, which means that anyone who chooses to can access the source code, make any changes or modifications they need, and freely share these changes back out to the community at large. It also means that OpenStack has the benefit of thousands of developers all over the world working in tandem to develop the strongest, most robust, and most secure product that they can.

What are the components of OpenStack?

OpenStack is made up of many different moving parts. Because of its open nature, anyone can add additional components to OpenStack to help it to meet their needs. But the OpenStack community has collaboratively identified nine key components that are a part of the "core" of OpenStack, which are distributed as a part of any OpenStack system and officially maintained by the OpenStack community.

- **Nova** is the primary computing engine behind OpenStack. It is used for deploying and managing large numbers of virtual machines and other instances to handle computing tasks.

- **Swift** is a storage system for objects and files. Rather than the traditional idea of a referring to files by their location on a disk drive, developers can instead refer to a unique identifier referring to the file or piece of information and let OpenStack decide where to store this information. This makes scaling easy, as developers don’t have the worry about the capacity on a single system behind the software. It also allows the system, rather than the developer, to worry about how best to make sure that data is backed up in case of the failure of a machine or network connection.

- **Cinder** is a block storage component, which is more analogous to the traditional notion of a computer being able to access specific locations on a disk drive. This more traditional way of accessing files might be important in scenarios in which data access speed is the most important consideration.

- **Neutron** provides the networking capability for OpenStack. It helps to ensure that each of the components of an OpenStack deployment can communicate with one another quickly and efficiently.

- **Horizon** is the dashboard behind OpenStack. It is the only graphical interface to OpenStack, so for users wanting to give OpenStack a try, this may be the first component they actually “see.” Developers can access all of the components of OpenStack individually through an application programming interface (API), but the dashboard provides system administrators a look at what is going on in the cloud, and to manage it as needed.

- **Keystone** provides identity services for OpenStack. It is essentially a central list of all of the users of the OpenStack cloud, mapped against all of the services provided by the cloud, which they have permission to use. It provides multiple means of access, meaning developers can easily map their existing user access methods against Keystone.

- **Glance** provides image services to OpenStack. In this case, "images" refers to images (or virtual copies) of hard disks. Glance allows these images to be used as templates when deploying new virtual machine instances.

- **Ceilometer** provides telemetry services, which allow the cloud to provide billing services to individual users of the cloud. It also keeps a verifiable count of each user's system usage of each of the various components of an OpenStack cloud. Think metering and usage reporting.

- **Heat** is the orchestration component of OpenStack, which allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application. In this way, it helps to manage the infrastructure needed for a cloud service to run.

Benefits of Using OpenStack in Business

Well, first of all, OpenStack comes with practically all of the benefits of cloud computing. That means it:

- **Enables rapid innovation**

OpenStack's orchestration and self-service capabilities offers developers and IT staff with faster and better access to IT resources. Because developers can provision machines rapidly and on-demand, they can significantly reduce development and testing periods and have more freedom to experiment with new ideas.

- **Cuts down time-to-market**

Faster deployment of IT resources also means end users and business units no longer have to wait days or weeks to start using the network services and applications they need. In turn, they would be more capable of rolling out and completing projects earlier than before.

- **Boosts scalability and resource utilization**

Although not as scalable as public clouds, OpenStack private clouds still offer a significant degree of scalability. You can still spin up and spin down servers on-demand. So, for example, if one department encounters a surge in demand for computing resources, IT resources may be temporarily redirected from other departments to the one that currently needs it the most.In addition, OpenStack also provides the following advantages over public clouds and proprietary cloud solutions:

- **Eases regulatory compliance**

Because OpenStack enables the construction of private, on-premise clouds, it can help in regulatory compliance endeavors. If your cloud is in your own data center, you'll have more control of access privileges, security measures, and security policies. You can personally take charge of ensuring that policies for securing personal data, financial data, and other confidential and regulated information are actually enforced and not just printed on a piece of paper.

- **Devoid of vendor lock-in**

One major problem with using a proprietary solution is vendor lock-in. If you're not happy with the vendor's services or the vendor closes shop, you cannot easily hop on to the next. OpenStack supports a variety of proprietary technologies and can operate in a smorgasbord of hypervisor and bare metal environments. Its ability to work with commodity hardware gives you more flexibility in choosing solutions based on a wider range of costs and competencies.

Cloud federation

Cloud federation is the practice of interconnecting the cloud computing environments of two or more service providers for the purpose of load balancing traffic and accommodating spikes in demand.

Cloud federation requires one provider to wholesale or rent computing resources to another cloud provider. Those resources become a temporary or permanent extension of the buyer's cloud computing environment, depending on the specific federation agreement between providers.

Cloud federation offers two substantial benefits to cloud providers. First, it allows providers to earn revenue from computing resources that would otherwise be idle or underutilized. Second, cloud federation enables cloud providers to expand their geographic footprints and accommodate sudden spikes in demand without having to build new points-of-presence (POPs).

Service providers strive to make all aspects of cloud federation—from cloud provisioning to billing support systems (BSS) and customer support— transparent to customers. When federating cloud services with a partner, cloud providers will also establish extensions of their customer-facing service-level agreements (SLAs) into their partner provider's data centers.

Advantages of Federated Cloud model

Some of the advantages of Federated Cloud model are

(i) Cost-effectiveness- Federated Clouds provide a larger amount of resources, which helps to improve cost-effectiveness and quality. This includes improvement for both the user and provider such as reducing completion time, increasing system throughput and optimizing the resource utilization.

(ii) Underutilized- A cloud can decide to provide resources to other clouds when it realizes that its data center is under-utilized at a given time.

(iii) Diverse geographical locations- Cloud service providers establish their data centers worldwide.Hence, there is a possibility of load sharing and performance improvement.

(iv) Avoidance of vendor lock-in - Federated cloud can freely transit workload among service providers to avoid vendor lock-in. In case a provider changes a policy or pricing that impact negatively, clients can easily migrate to some other providers.

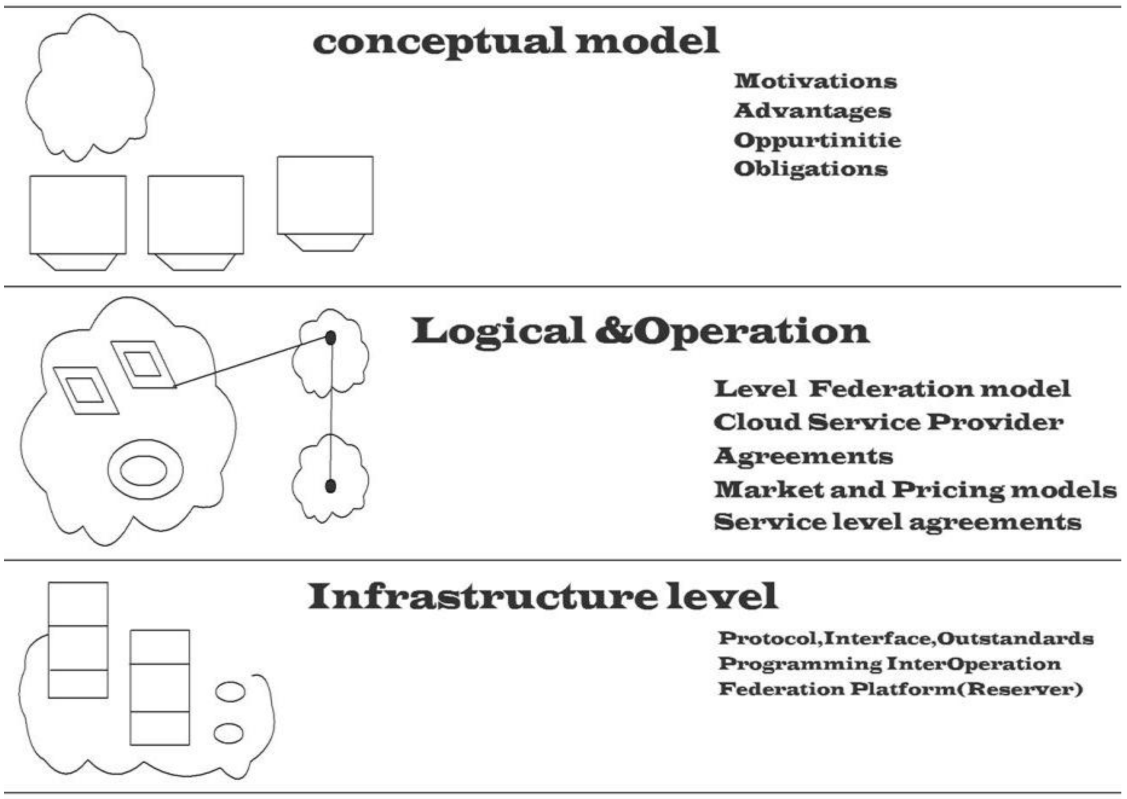
(v) Better SLA to customers- Cloud provider can provide better Service Level Agreements (SLA) to its customers, as a result of competition.

(vi) Guaranteed availability- During unexpected disasters, the cloud system is able to recover the services by federating with other service providers. The main aim of federated cloud service brokering has received a lot of attention in academic and industry in recent years. Broker based federated cloud processing is an emerging concept in cloud based services.

It connects a set of technologies, protocols and languages to communicate between customers and providers.

There are four basic types/levels of federation:
1)Permissive
2)Verified
3)Encrpted
4)Trusted

Permissive Federation:



1) It occurs when the server accepts a connection from a peer network servers without verifying its identy using DNS look as are certificate checking.
2) The lack of verification are authentication may let to domain schooling that is the unauthorized use of third party domain name in an e-mail messaage in order to pertent to be someone else.
3) Which opens the door to white spread spram and other
with the relafese of the jaberd 1.2 servers which included support for the server dialware protocol premissive federation mirt is device on the xmpp network.

Verified Federation:
1) This type of federation occurs from a peer has been verified it users information obtain we are DNS and by means of domain specificas exchange before hand.
2) The connection is not encrypted and the use of identity verification effectively prevence domains pooling make this works.

3) Fedreation requires proper DNS setup and that is still subjective DNS voisoning attacks.
4) Verified federation has been the default service policy on the open XMPP since the release of the open source jaberd 1.2 server.

Encrypted Federation:
1) In this mode a server accepts a connection from a peer if an only if the peer supports TLS(Transport Layer Security)as define for XMPP in RFS(Request For Comments)3920.
2) The peer must prevent a digital certificate the certificate may be selfsine but this prevence using mutal authendication.If this is the case both parties procede to weekely verify identity using server dial pair.
3) XEP0220 define the server dialup protocol which is used between XMPP servers to provide identity verification servers dial pair uses the DNS of the basis for verifying identity the basic approaach is that when a receiving server receives a server to server connection request from an orginatting server although server dialpair does not provide strong authendication are trusted federation and although it is subjective DNS voisoning attacks this results in an encrypted connection with we identity verification.

Trusted Federation:
1) Hear a server accepts a connection from a peer under only the stipulation that the peer supports TLS and the peer can present a digital certificate issued by a root certification authority(CA) that is trusted by the authendicating server.
2) The list of trusted root CAAS may be detemine by one or more factors such as the os,xmpp server or local service policy.
3) In trusted federation use of digital certificates results not only in channel encryption but also in strong authendication.
4) The use of trusted domain certificate prevense DNS voisoning effectively attacks but makes federation more difficult since such certificates have treditionally not being easy to uptain.

Future of federation

Some see the future of cloud computing as one big public cloud. Others believe that enterprises will ultimately build a single large cloud to host all their corporate services. This is, of course, because the benefit of cloud computing is dependent on large – very large – scale infrastructure, which provides administrators and service administrators and consumers the ability for ease of deployment, self service, elasticity, resource pooling and economies of scale. However, as cloud continues to evolve – so do the services being offered.

Cloud Services & Hybrid Clouds

Services are now able to reach a wider range of consumers, partners, competitors and public audiences. It is also clear that storage, compute power, streaming, analytics and other advanced services are best served when they are in an environment tailored for the proficiency of that service.

One method of addressing the need of these service environments is through the advent of hybrid clouds. Hybrid clouds, by definition, are composed of multiple distinct cloud infrastructures connected in a manner that enables services and data access across the combined infrastructure. The intent is to leverage the additional benefits that hybrid cloud offers without disrupting the traditional cloud benefits. While hybrid cloud benefits come through the ability to distribute the work stream, the goal is to continue to realize the ability for managing peaks in demand, to quickly make services available and capitalize on new business opportunities.

While the hybrid cloud has a variety of benefits, the expansion of cloud into a hybrid model may also increase risk to an enterprise through data exposure and violation of corporate policy. The question then becomes, How do we garner the additive benefits of hybrid cloud while minimizing risks? The answer is through Federation.

The Solution: Federation

Federation creates a hybrid cloud environment with an increased focus on maintaining the integrity of corporate policies and data integrity. Think of federation as a pool of clouds connected through a channel of gateways; gateways which can be used to optimize a cloud for a service or set of specific services. Such gateways can be used to segment service audiences or to limit access to specific data sets. In essence, federation has the ability for enterprises to service their audiences with economy of scale without exposing critical applications or vital data through weak policies or vulnerabilities.

Many would raise the question: if Federation creates multiples of clouds, doesn't that mean cloud benefits are diminished? I believe the answer is no, due to the fact that a fundamental change has transformed enterprises through the original adoption of cloud computing, namely the creation of a flexible environment able to adapt rapidly to changing needs based on policy and automation.