UNIT V

5.1 Introduction to Hadoop Framework

- Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models.
- Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.
- Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes.

Hadoop = MapReduce + HDFS

(MapReduce→ Processing ; HDFS → Storage)

Users of Hadoop:

- Hadoop is running search on some of the Internet's largest sites:
    - Amazon Web Services: Elastic MapReduce
    - AOL: Variety of uses, e.g., behavioral analysis & targeting
    - Ebay: Search optimization (532-node cluster)
    - Facebook: Reporting/analytics, machine learning (1100 m.)
    - LinkedIn: People You May Know (2x50 machines)
    - Twitter: Store + process tweets, log files, other data Yahoo: >36,000 nodes; biggest cluster is 4,000 nodes

Hadoop Architecture

- Hadoop has a Master Slave Architecture for both Storage & Processing
- Hadoop framework includes following four modules:
- Hadoop Common: These are Java libraries and provide file system and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.

- ❖ Hadoop YARN: This is a framework for job scheduling and cluster resource management.
- ❖ Hadoop Distributed File System (HDFS): A distributed file system that provides high-throughput access to application data.
- ❖ HadoopMapReduce: This is system for parallel processing of large data sets.

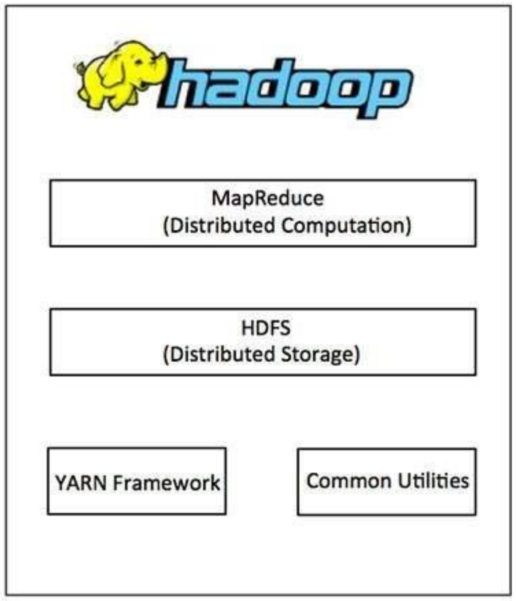


Figure 5.1 Hadoop Architecture

- ☐ The Hadoop core is divided into two fundamental layers:
  - MapReduce engine
  - HDFS
- ☐ The MapReduce engine is the computation engine running on top of HDFS as its data storage manager.
- ☐ HDFS: HDFS is a distributed file system inspired by GFS that organizes files and stores their data on a distributed computing system.
- ☐ HDFS Architecture: HDFS has a master/slave architecture containing a single Name Node as the master and a number of Data Nodes as workers (slaves).

<u>HDFS</u>

- ☐ To store a file in this architecture,
- ☐ HDFS splits the file into fixed-size blocks (e.g., 64 MB) and stores them on workers (Data Nodes).
- ☐ The mapping of blocks to Data Nodes is determined by the Name Node.
- ☐ The NameNode (master) also manages the file system’s metadata and namespace.

- Namespace is the area maintaining the metadata, and metadata refers to all the information stored by a file system that is needed for overall management of all files.
- NameNode in the metadata stores all information regarding the location of input splits/blocks in all DataNodes.
- Each DataNode, usually one per node in a cluster, manages the storage attached to the node.
- Each DataNode is responsible for storing and retrieving its file blocks

<u>HDFS- Features</u>

Distributed file systems have special requirements

- Performance
- Scalability
- Concurrency Control
- Fault Tolerance
- Security Requirements

<u>HDFS Fault Tolerance</u>

Block replication:

- To reliably store data in HDFS, file blocks are replicated in this system.
- HDFS stores a file as a set of blocks and each block is replicated and distributed across the whole cluster.
- The replication factor is set by the user and is three by default.
- Replica placement: The placement of replicas is another factor to fulfill the desired fault tolerance in HDFS.
- Storing replicas on different nodes (DataNodes) located in different racks across the whole cluster.
- HDFS stores one replica in the same node the original data is stored.
- One replica on a different node but in the same rack
- One replica on a different node in a different rack.
- Heartbeats and Blockreports are periodic messages sent to the NameNode by each DataNode in a cluster.

- ☐ Receipt of a Heartbeat implies that the DataNode is functioning properly.
- ☐ Each Blockreport contains a list of all blocks on a DataNode .
- ☐ The NameNode receives such messages because it is the sole decision maker of all replicas in the system.

<u>HDFS High Throughput</u>

- ☐ Applications run on HDFS typically have large data sets.
- ☐ Individual files are broken into large blocks to allow HDFS to decrease the amount of metadata storage required per file.
- ☐ The list of blocks per file will shrink as the size of individual blocks increases.
- ☐ By keeping large amounts of data sequentially within a block, HDFS provides fast streaming reads of data.

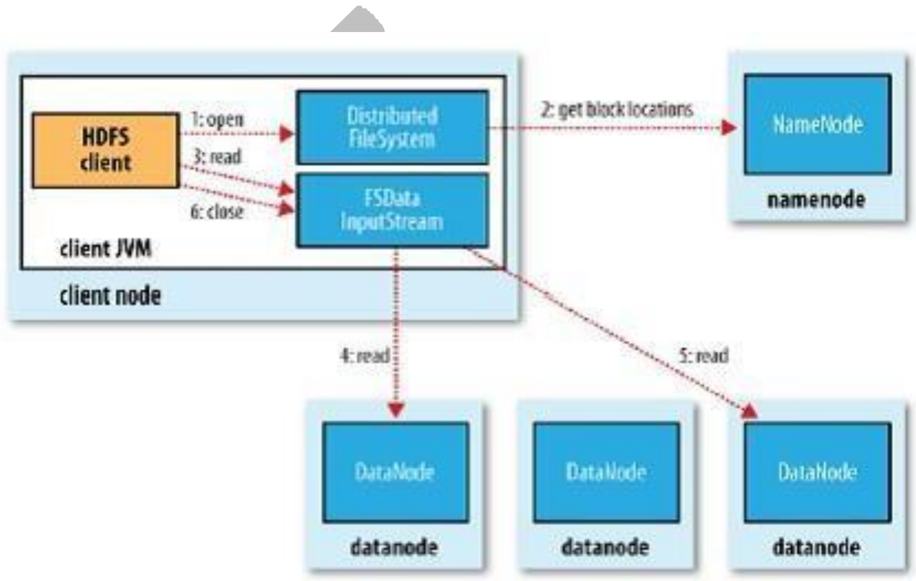
<u>HDFS- Read Operation</u>

<u>Reading a file :</u>

- ☐ To read a file in HDFS, a user sends an "open" request to the NameNode to get the location of file blocks.
- ☐ For each file block, the NameNode returns the address of a

set of DataNodes containing replica information for the requested file.

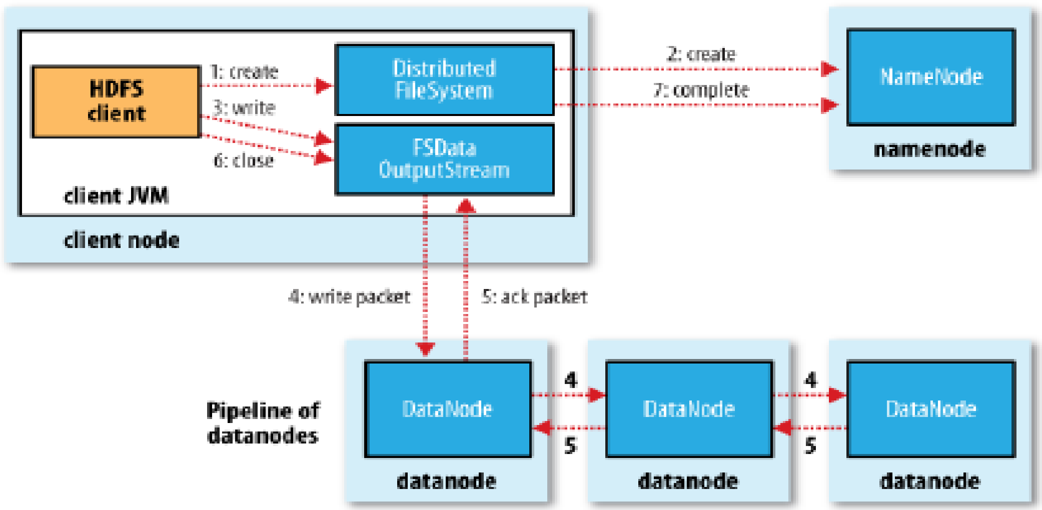
- ☐ The number of addresses depends on the number of block replicas.
- ☐ The user calls the read function to connect to the closest DataNode containing the first block of the file.
- ☐ Then the first block is streamed from the respective DataNode to the user.
- ☐ The established connection is terminated and the same process is repeated for all blocks of the requested file until the whole file is streamed to the user.



HDFS-Write Operation

Writing to a file:

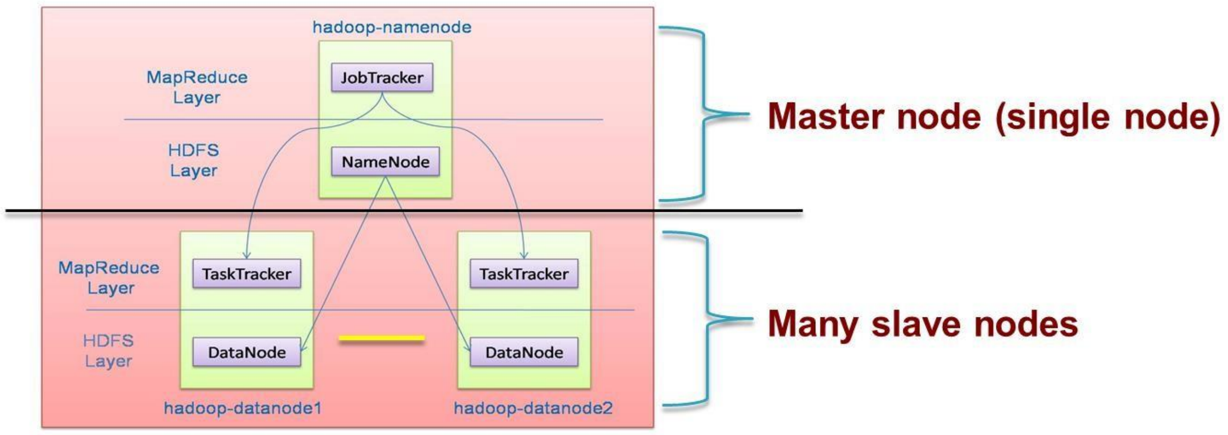
- ☐ To write a file in HDFS, a user sends a "create" request to the NameNode to create a new file in the file system namespace.
- ☐ If the file does not exist, the NameNode notifies the user and allows him to start writing data to the file by calling the write function.
- ☐ The first block of the file is written to an internal queue termed the data queue.
- ☐ A data streamer monitors its writing into a DataNode.
- ☐ Each file block needs to be replicated by a predefined factor.
- ☐ The data streamer first sends a request to the NameNode to get a list of suitable DataNodes to store replicas of the first block.
- ☐ The steamer then stores the block in the first allocated DataNode.
- ☐ Afterward, the block is forwarded to the second DataNode by the first DataNode.
- ☐ The process continues until all allocated DataNodes receive a replica of the first block from the previous DataNode.
- ☐ Once this replication process is finalized, the same process starts for the second block.



4

5.1.1 Architecture of Mapreduce in Hadoop

- Distributed file system (HDFS)
- Execution engine (MapReduce)



Properties of Hadoop Engine

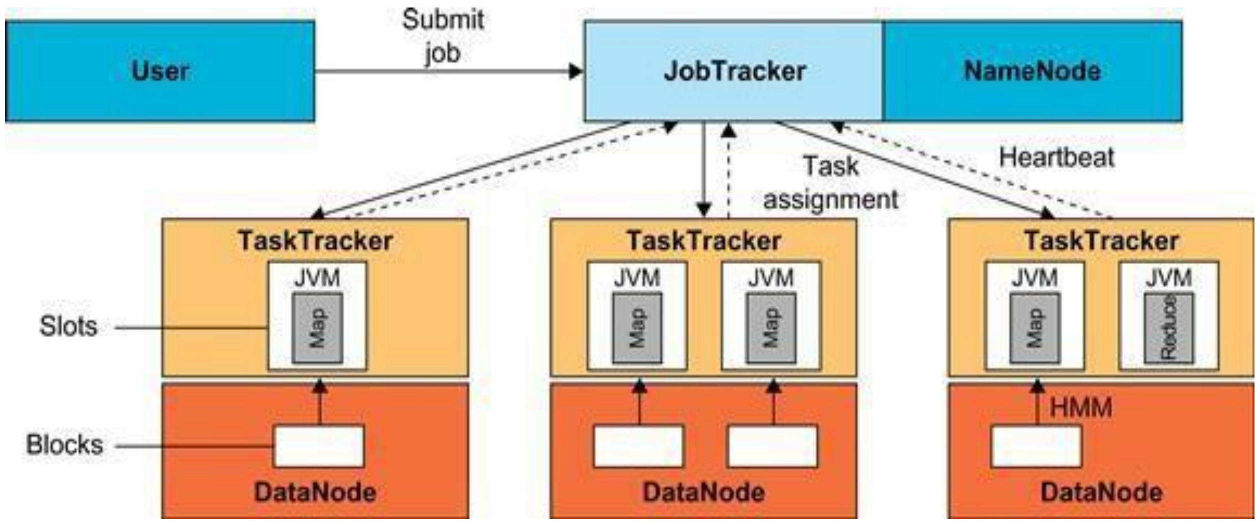
- HDFS has a master/slave architecture containing
- A single NameNode as the master and
- A number of DataNodes as workers (slaves).
- To store a file in this architecture, HDFS splits the file into fixed-size blocks (e.g., 64 MB) and stores them on workers (DataNodes).
- The NameNode (master) also manages the file system's metadata and namespace.
- Job Tracker is the master node (runs with the namenode)

- Receives the user’s job
- Decides on how many tasks will run (number of mappers)
- Decides on where to run each mapper (concept of locality)

➢ Task Tracker is the slave node (runs on each datanode)

- Receives the task from Job Tracker
- Runs the task until completion (either map or reduce task)
- Always in communication with the Job Tracker reporting progress (heartbeats)

Running a Job in Hadoop



❖ Three components contribute in running a job in this system:

- a user node,
- a JobTracker, and
- severalTaskTrackers.

- The data flow starts by calling the runJob(conf) function inside a user program running on the user node, in which conf is an object containing some tuning parameters for the MapReduce
- Job Submission: Each job is submitted from a user node to the JobTracker node.
- Task assignment : The JobTracker creates one map task for each computed input split
- Task execution : The control flow to execute a task (either map or reduce) starts inside the TaskTracker by copying the job JAR file to its file system.
- Task running check : A task running check is performed by receiving periodic heartbeat messages to the JobTracker from the TaskTrackers.
- Heartbeat:notifies the JobTracker that the sending TaskTracker is alive, and whether the sending TaskTracker is ready to run a new task.

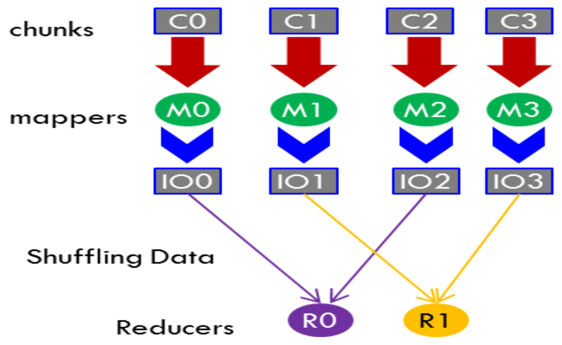
The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing, including:

- HadoopCore, our flagship sub-project, provides a distributed filesystem (HDFS) and support for the MapReduce distributed computing metaphor.
- HBase builds on Hadoop Core to provide a scalable, distributed database.
- Pig is a high-level data-flow language and execution framework for parallel computation.It is built on top of Hadoop Core.
- ZooKeeperis a highly available and reliable coordination system. Distributed applications use ZooKeeper to store and mediate updates for critical shared state.
- Hive is a data warehouse infrastructure built on Hadoop Core that provides data summarization, adhoc querying and analysis of datasets.

MAP REDUCE

- MapReduce is a programming model for data processing.
- MapReduce is designed to efficiently process large volumes of data by connecting many commodity computers together to work in parallel
- Hadoop can run MapReduce programs written in various languages like Java, Ruby, and Python
- MapReduce works by breaking the processing into two phases:
    - The map phase and
    - The reduce phase.

- ❖ Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer.
- ❖ The programmer also specifies two functions:
  - o The map function and
  - o The reduce function.
- ❖ In MapReduce, chunks are processed in isolation by tasks called Mappers
- ❖ The outputs from the mappers are denoted as intermediate outputs (IOs) and are brought into a second set of tasks called Reducers
- ❖ The process of bringing together IOs into a set of Reducers is known as shuffling process
- ❖ The Reducers produce the final outputs (FOs)



- ☐ Overall, MapReduce breaks the data flow into two phases, map phase and reduce phase

<u>Mapreduce Workflow</u>

Application writer specifies

- ❖ A pair of functions called Mapper and Reducer and a set of input files and submits the job
- ❖ Input phase generates a number of FileSplits from input files (one per Map task)
- ❖ The Map phase executes a user function to transform input key-pairs into a new set of key-pairs
- ❖ The framework Sorts & Shuffles the key-pairs to output nodes
- ❖ The Reduce phase combines all key-pairs with the same key into new keypairs
- ❖ The output phase writes the resulting pairs to files as “parts”

Characteristics of MapReduce is characterized by:

- ❖ Its simplified programming model which allows the user to quickly write and test distributed systems

- Its efficient and automatic distribution of data and workload across machines
- Its flat scalability curve. Specifically, after a Mapreduce program is written and functioning on 10 nodes, very little-if any- work is required for making that same program run on 1000 nodes

The core concept of MapReduce in Hadoop is that input may be split into logical chunks, and each chunk may be initially processed independently, by a map task. The results ofthese individual processing chunks can be physically partitioned into distinct sets, which are then sorted. Each sorted chunk is passed to a reduce task.

A map task may run on any compute node in the cluster, and multiple map tasks may berunning in parallel across the cluster. The map task is responsible for transforming the input records into key/value pairs. The output of all of the maps will be partitioned, and each partition will be sorted. There will be one partition for each reduce task. Each partition's sorted keys and thevalues associated with the keys are then processed by the reduce task. There may be multiple reduce tasks running in parallel on the cluster.

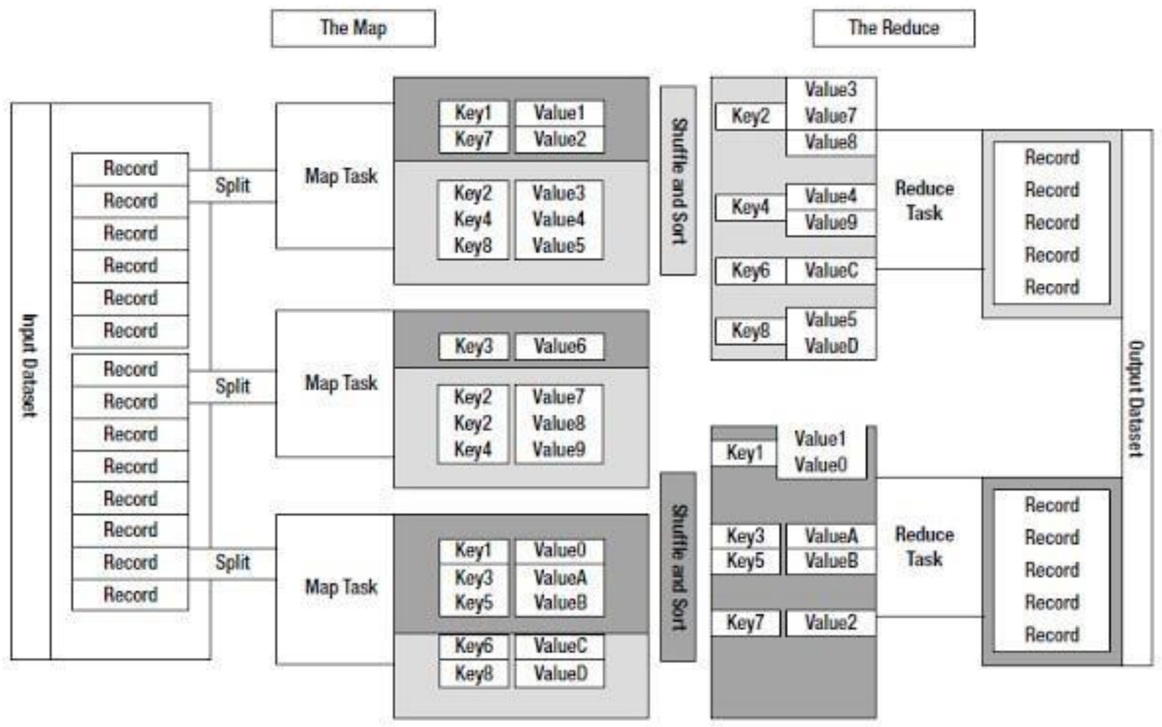
The application developer needs to provide only four items to the Hadoop framework: the class that will read the input records and transform them into one key/value pair per record, a map method, a reduce method, and a class that will transform the key/value pairs that the reduce method outputs into output records.

My first MapReduce application was a specialized web crawler. This crawler received as input large sets of media URLs that were to have their content fetched and processed. The media items were large, and fetching them had a significant cost in time and resources.

The job had several steps:

1. Ingest the URLs and their associated metadata.
2. Normalize the URLs.
3. Eliminate duplicate URLs.
4. Filter the URLs against a set of exclusion and inclusion filters.
5. Filter the URLs against a do not fetch list.
6. Filter the URLs against a recently seen set.
7. Fetch the URLs.
8. Fingerprint the content items.
9. Update the recently seen set.

10. Prepare the work list for the next application.

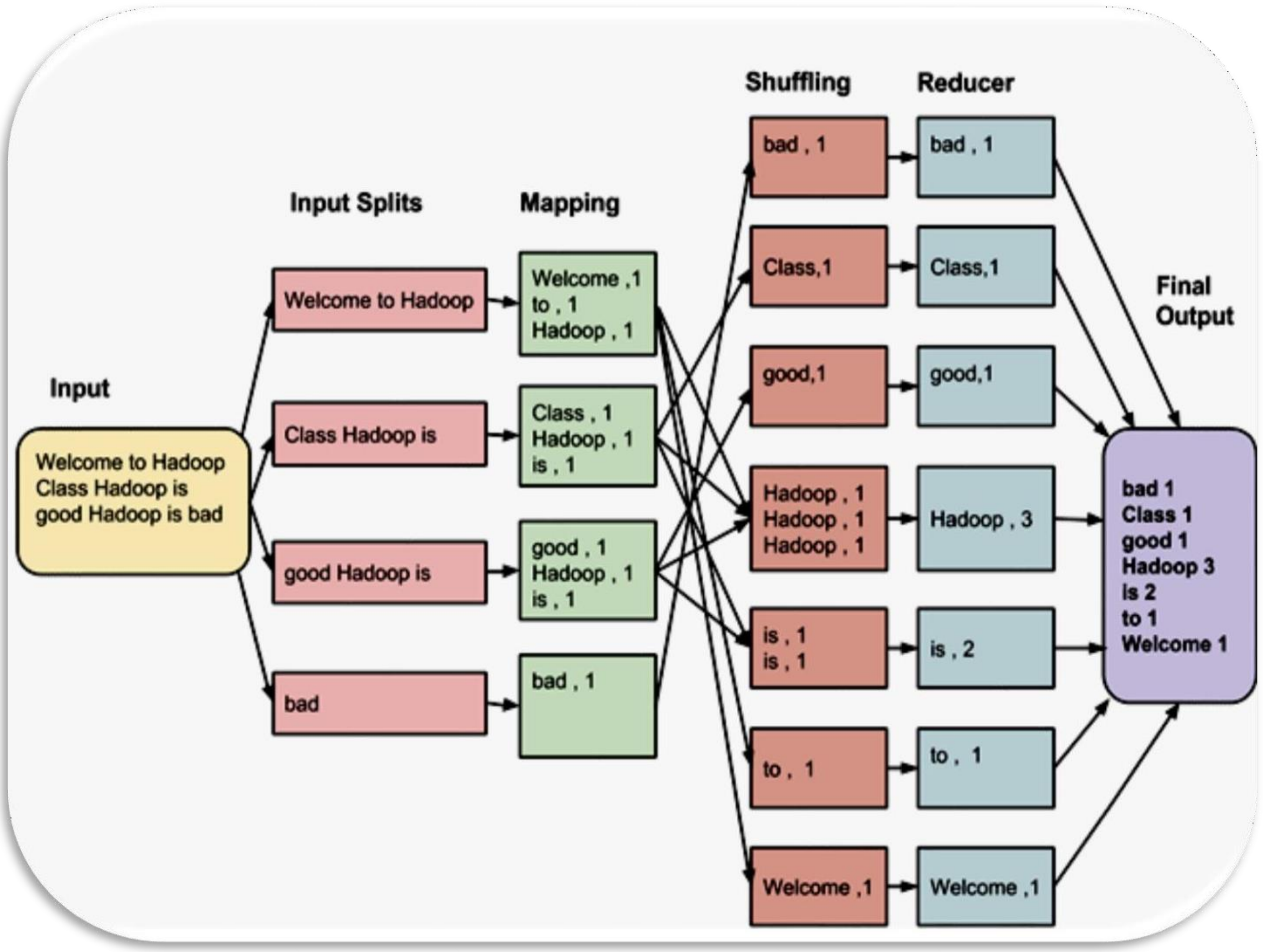


<u>Input File:</u>

Welcome to Hadoop Class

Hadoop is good

Hadoop is bad


Input
Welcome to Hadoop Class Hadoop is good Hadoop is bad
Input Splits
Welcome to Hadoop
Class Hadoop is
good Hadoop is
bad
Mapping
Welcome ,1 to , 1 Hadoop , 1
Class , 1 Hadoop , 1 is , 1
good , 1 Hadoop , 1 is , 1
bad , 1
Shuffling
bad , 1
Class,1
good,1
Hadoop , 1 Hadoop , 1 Hadoop , 1
is , 1 is , 1
to , 1
Welcome ,1
Reducer
bad , 1
Class,1
good,1
Hadoop , 3
is , 2
to , 1
Welcome ,1
Final Output
bad 1 Class 1 good 1 Hadoop 3 is 2 to 1 Welcome 1

5.2 VirtualBox

VirtualBox is a general-purpose full virtualizer for x86 hardware, targeted at server, desktop, and embedded use. Developed initially by Innotek GmbH, it was acquired by Sun Microsystems in 2008, which was, in turn, acquired by Oracle in 2010.

VirtualBox is an extremely feature rich, high-performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. It supports Windows, Linux, Macintosh, Sun Solaris, and FreeBSD. Virtual Box has supported Open Virtualization Format (OVF) since version 2.2.0 (April 2009)

Operating system virtualization allows your computer’s hardware to run many operating system images simultaneously. One of the most used instances of this is to test software or applications in a different environment, rather than on a different computer. This can potentially save you quite a bit of money by running multiple servers virtually on one computer.

Pros and Cons of Virtual Box over VMWare

- Virtual Box is a general-purpose full virtualizer for x86 hardware, targeted at server, desktop, and embedded use.
- This product is a Type 2 hypervisor, so it’s virtualization host software that runs on an already established operating system as an application.
- With VirtualBox, it’s also possible to share your clipboard between the virtualized and host operating system.
- While VMWare functions on Windows and Linux, not Mac, Virtual Box works with Windows, Mac, and Linux computers.
- Virtual Box truly has a lot of support because it's open-source and free. Being open-source means that recent releases are sometimes a bit buggy, but also that they typically get fixed relatively quickly.
- With VMWare player, instead, you have to wait for the company to release an update to fix the bugs.
- Virtual Box offers you an unlimited number of snapshots.
- Virtual Box is easy to install, takes a smaller amount of resources, and is many people’s first choice.
- VMWare often failed to detect my USB device Besides, VirtualBox can detect as well as identify USB devices after installing Virtual Box Extension Pack.
- With VirtualBox Guest Addition, files can be dragged and copied between VirtualBox and host.
- VMware outperforms VirtualBox in terms of CPU and memory utilization.
- VirtualBox has snapshots and VMware has rollback points to which you can revert back to in case you break your virtual machine.
- VMware calls it Unity mode and VirtualBox calls it the seamless mode and they both enable you to open application windows on the host machine, while the VM supporting that app is running in the background quietly.

- In the case of VirtualBox, the UI is simple and clean. Your settings are split into Machine Tools and Global Tools and the former is for creating, modifying, starting, stop and deleting virtual machines. VMware, on the other hand, has a much more complicated UI, menu items are named with technical terms which may seem like jargon to average users. This is primarily because the VMware folks cater to cloud providers and server-side virtualizations more.
- In case of VirtualBox, PCIe pass through can be accomplished, although you might have to jump through some hoops. VMware on the other offers excellent customer support and would help you out if you are in a fix.
- VirtualBox is basically a highly secure program that allows users to download and run OS as a virtual machine. With Virtual Box, users are able to abstract their hardware via complete virtualization thus guaranteeing a higher degree of protection from viruses running in theguest OS.
- Virtual Box offers limited support for 3D graphics. And VMWare has a high-level 3Dgraphics support with DX10 and OpenGL 3.3 support.
- The real advantage of VirtualBox over VMware server lies in its performance. VirtualBox apparently runs faster than VMware server. A timed experiment of an installation of Windows XP as the guest OS took 20 mins in VirtualBox and 35 mins on VMware server. A similar test on the booting time of the guest OS also shows favor to VirtualBox with timing of 45secs compared to 1min 39 secs on VMware server.
- In VirtualBox, the remote file sharing feature is built right in the package. Setting up remote file sharing is easy and you only need to do it once: point the file path to the directory that you want to share.

5.3 GOOGLE APPLICATION ENGINE (GAE)

Google App Engine is a PaaS cloud that provides a complete Web service environment(Platform)

GAE provides Web application development platform for users.

All required hardware, operating systems and software are provided to clients.

Clients can develop their own applications, while App Engine runs the applications on Google's servers.

GAE helps to easily develop an Web Application

App Engine only supports the Java and Python programming languages.

The Google App Engine (GAE) provides a powerful distributed data storage service.

GOOGLE CLOUD INFRASTRUCTURE

Google has established cloud development by making use of large number of data centers.

Eg: Google established cloud services in

- ❖ Gmail
- ❖ Google Docs
- ❖ Google Earth etc.

These applications can support a large number of users simultaneously with High Availability (HA).

In 2008, Google announced the GAE web application platform.

GAE enables users to run their applications on a large number of data centers.

Google App Engine environment includes the following features :

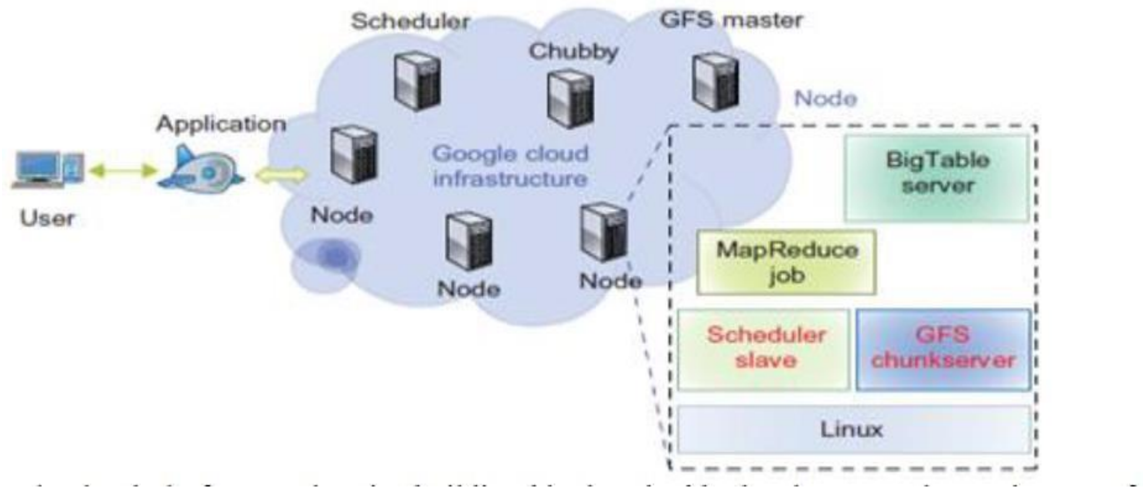
- ❖ Dynamic web serving
- ❖ Persistent(constant) storage with queries, sorting, and transactions
- ❖ Automatic scaling and load balancing

Provides Application Programming Interface(API) for authenticating users.

Send email using Google Accounts.

Local development environment that simulates(create) Google App Engine on your computer.

GAE ARCHITECTURE



TECHNOLOGIES USED BY GOOGLE ARE

Google File System(GFS) ->for storing large amounts of data.

MapReduce->for application program development.

Chubby-> for distributed application lock services.

BigTable-> offers a storage service.

Third-party application providers can use GAE to build cloud applications for providing services.

Inside each data center, there are thousands of servers forming different clusters.

GAE runs the user program on Google's infrastructure.

Application developers now do not need to worry about the maintenance of servers.

GAE can be thought of as the combination of several software components.

GAE supports Python and Java programming environments.

FUNCTIONAL MODULES OF GAE

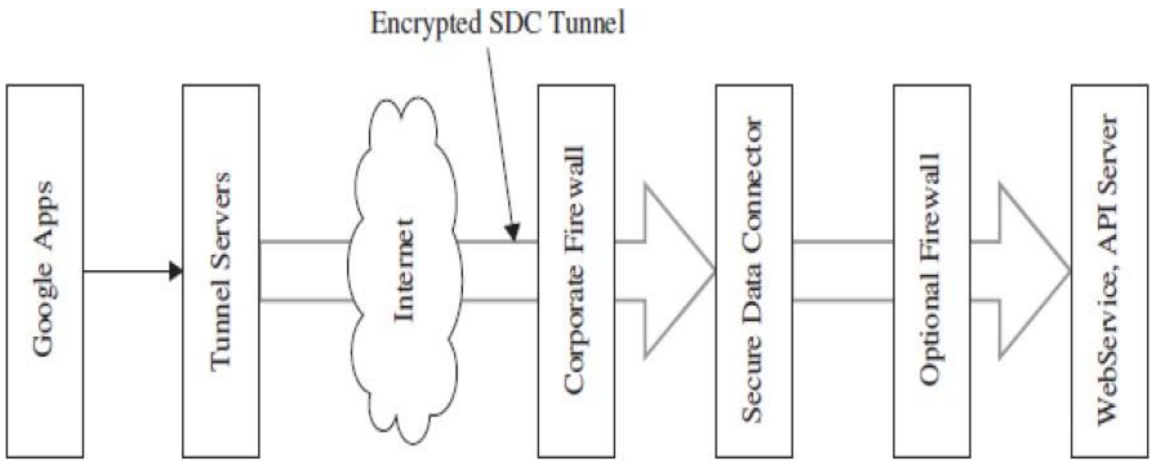
The GAE platform comprises the following five major components.

DataStore: offers data storage services based on BigTable techniques.

The Google App Engine (GAE) provides a powerful distributed data storage service.

This provides a secure data Storage.

GOOGLE SECURE DATA CONNECTOR (SDC)



FUNCTIONAL MODULES OF GAE

When the user wants to get the data, he/she will first send an authorized data requests to Google Apps.

It forwards the request to the tunnel server.

The tunnel servers validate the request identity.

If the identity is valid, the tunnel protocol allows the SDC to set up a connection, authenticate, and encrypt the data that flows across the Internet.

SDC also validates whether a user is authorized to access a specified resource.

Application runtime environment offers a platform for web programming and execution.

It supports two development languages: Python and Java.

Software Development Kit (SDK) is used for local application development.

The SDK allows users to execute test runs of local applications and upload application code.

Administration console is used for easy management of user application development cycles.

GAE web service infrastructure provides special guarantee flexible use and management of storage and network resources by GAE.

Google offers essentially free GAE services to all Gmail account owners.

We can register for a GAE account or use your Gmail account name to sign up for the service.

The service is free within a quota.

If you exceed the quota, extra amount will be charged.

Allows the user to deploy user-built applications on top of the cloud infrastructure.

They are built using the programming languages and software tools supported by the provider (e.g., Java, Python)

GAE APPLICATIONS

Well-known GAE applications

Google Search Engine

Google Docs

Google Earth

Gmail

These applications can support large numbers of users simultaneously.

Users can interact with Google applications via the web interface provided by each application.

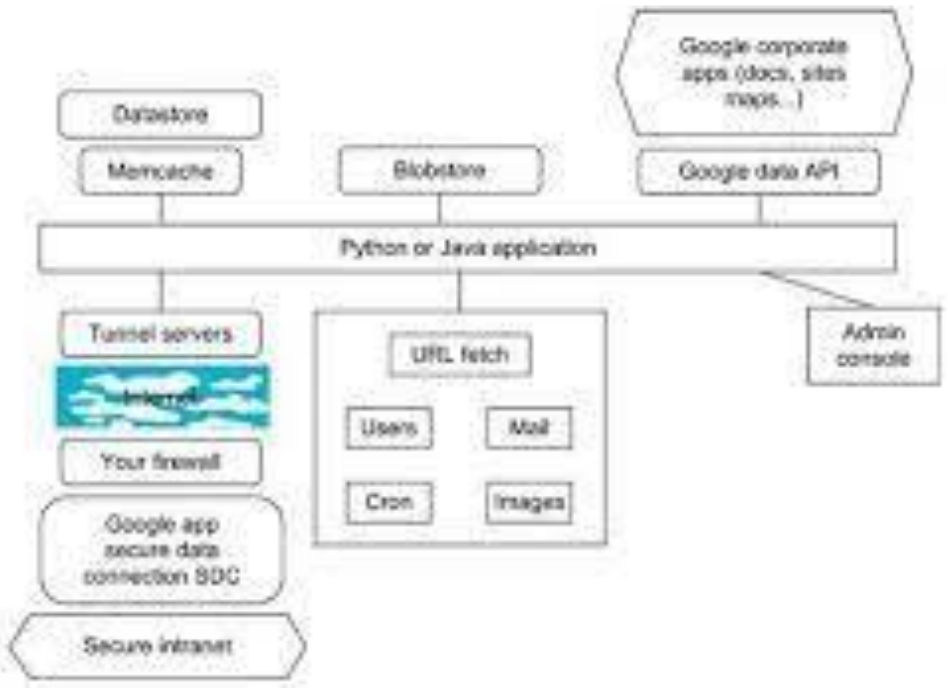
Applications run in the Google data centers.

Inside each data center, there might be thousands of server nodes to form different clusters.

Each cluster can run multipurpose servers.

5. 3.1 Programming Support of Google App Engine

GAE programming model for two supported languages: Java and Python. A client environment includes an Eclipse plug-in for Java allows you to debug your GAE on your local machine. Google Web Toolkit is available for Java web application developers. Python is used with frameworks such as Django and CherryPy, but Google also has webapp Python environment.



There are several powerful constructs for storing and accessing data. The data store is a NOSQL data management system for entities. Java offers Java Data Object (JDO) and Java Persistence API (JPA) interfaces implemented by the Data Nucleus Access platform, while Python has a SQL-like query language called GQL. The performance of the data store can be enhanced by in-memory caching using the memcache, which can also be used independently of the data store.

Recently, Google added the blobstore which is suitable for large files as its size limit is 2 GB. There are several mechanisms for incorporating external resources. The Google SDC Secure Data Connection can tunnel through the Internet and link your intranet to an external GAE application. The URL Fetch operation provides the ability for applications to fetch resources and communicate with other hosts over the Internet using HTTP and HTTPS requests.

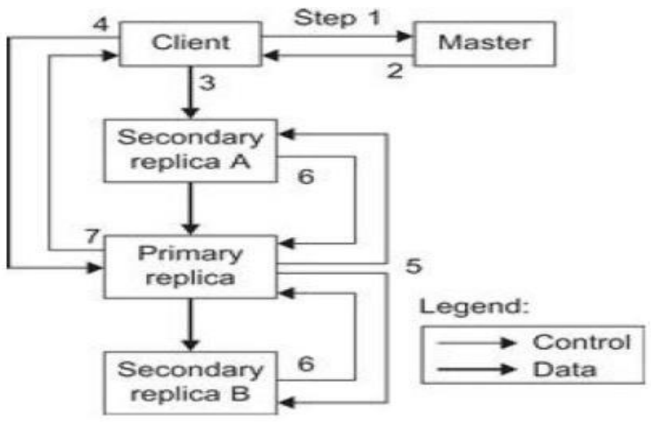
An application can use Google Accounts for user authentication. Google Accounts handles user account creation and sign-in, and a user that already has a Google account (such as a Gmail account) can use that account with your app. GAE provides the ability to manipulate image data using a dedicated Images service which can resize, rotate, flip, crop, and enhance images. A GAE application is configured to consume resources up to certain limits or quotas. With quotas, GAE ensures that your application won't exceed your budget, and that other applications running on GAE won't impact the performance of your app. In particular, GAE use is free up to certain quotas.

Google File System (GFS)

GFS is a fundamental storage service for Google's search engine. GFS was designed for Google applications, and Google applications were built for GFS. There are several concerns in GFS. rate). As servers are composed of inexpensive commodity components, it is the norm rather than the exception that concurrent failures will occur all the time. Other concerns the file size in GFS. GFS typically will hold a large number of huge files, each 100 MB or larger, with files that are multiple GB in size quite common. Thus, Google has chosen its file data block size to be 64 MB instead of the 4 KB in typical traditional file systems. The I/O pattern in the Google application is also special. Files are typically written once, and the write operations are often the appending data blocks to the end of files. Multiple appending operations might be concurrent. The customized API can simplify the problem and focus on Google applications.

Figure shows the GFS architecture. It is quite obvious that there is a single master in the whole cluster. Other nodes act as the chunk servers for storing data, while the single master stores the metadata. The file system namespace and locking facilities are managed by the master. The master periodically communicates with the chunk servers to collect management information as well as give instructions to the chunk servers to do work such as load balancing or fail recovery.

The master has enough information to keep the whole cluster in a healthy state. Google uses a shadow master to replicate all the data on the master, and the design guarantees that all the data operations are performed directly between the client and the chunk server. The controlmessages are transferred between the master and the clients and they can be cached for future use. With the current quality of commodity servers, the single master can handle a cluster of more than 1,000 nodes.



The mutation takes the following steps:

1. The client asks the master which chunk server holds the current lease for the chunk and the

locations of the other replicas. If no one has a lease, the master grants one to a replica it chooses (not shown).

2. The master replies with the identity of the primary and the locations of the other (secondary) replicas. The client caches this data for future mutations. It needs to contact the master again only when the primary becomes unreachable or replies that it no longer holds a lease.
3. The client pushes the data to all the replicas. Each chunk server will store the data in an internal LRU buffer cache until the data is used or aged out. By decoupling the data flow from the control flow, we can improve performance by scheduling the expensive data flow based on the network topology regardless of which chunk server is the primary.
4. Once all the replicas have acknowledged receiving the data, the client sends a write request to the primary. The request identifies the data pushed earlier to all the replicas. The primary assigns consecutive serial numbers to all the mutations it receives, possibly from multiple clients, which provides the necessary serialization. It applies the mutation to its own local state in serial order.
5. The primary forwards the write request to all secondary replicas. Each secondary replica applies mutations in the same serial number order assigned by the primary.
6. The secondaries all reply to the primary indicating that they have completed the operation.
7. The primary replies to the client. Any errors encountered at any replicas are reported to the client. In case of errors, the write corrects at the primary and an arbitrary subset of the secondary replicas. The client request is considered to have failed, and the modified region is left in an inconsistent state. Our client code handles such errors by retrying the failed mutation

Big Table

BigTable was designed to provide a service for storing and retrieving structured and semistructured data. BigTable applications include storage of web pages, per-user data, and geographic locations. The database needs to support very high read/write rates and the scale might be millions of operations per second. Also, the database needs to support efficient scans over all or interesting subsets of data, as well as efficient joins of large one-to-one and one-to-many data sets. The application may need to examine data changes over time.

The BigTable system is scalable, which means the system has thousands of servers, terabytes of in-memory data, petabytes of disk-based data, millions of reads/writes per second, andefficient scans. BigTable is used in many projects, including Google Search, Orkut, and Google Maps/Google Earth, among others.

The BigTable system is built on top of an existing Google cloud infrastructure. BigTable uses the following building blocks:

1. GFS: stores persistent state
2. Scheduler: schedules jobs involved in BigTable serving
3. Lock service: master election, location bootstrapping
4. MapReduce: often used to read/write BigTable data.

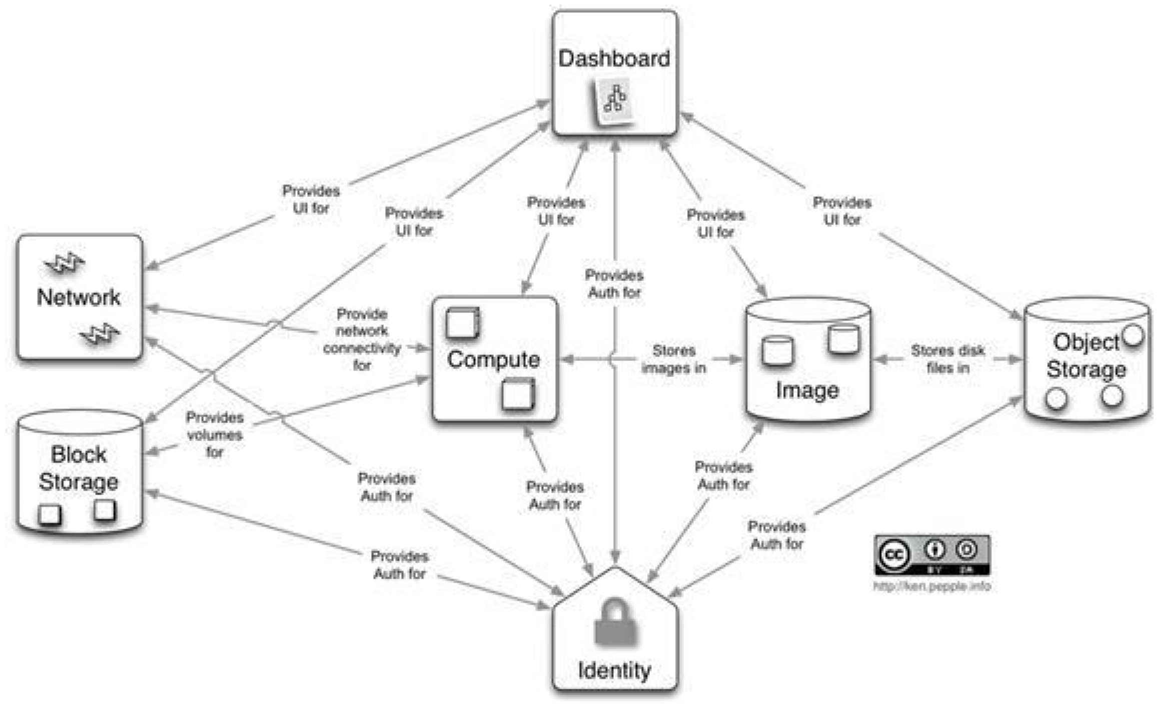
5.4 Open Stack

□ OpenStack is a free and open-source software platform for cloud computing.

□ OpenStack is a virtualization tool to manage your virtual infrastructure.

□ **OpenStack consists of multiple components.**

- Compute (Nova)
- Image Service (Glance)
- Object Storage (Swift)
- Dashboard (Horizon)
- Identity Service (Keystone)
- Networking (Neutron)
- Block Storage (Cinder)
- Telemetry (Ceilometer)
- Orchestration (Heat)
- Workflow (Mistral)
- Database (Trove)
- Elastic map reduce (Sahara)
- Bare metal (Ironic)
- Messaging (Zaqar)
- Shared file system (Manila)
- DNS (Designate)
- Search (Searchlight)
- Key manager (Barbican)
- Root Cause Analysis (Vitrage)
- Rule-based alarm actions (Aodh)



Compute (Nova)

 OpenStack Compute is also known as OpenStack Nova.

 Nova is the primary compute engine of OpenStack, used for deploying and managing virtual machine.

 OpenStack Compute manages pools of computer resources and work with virtualization technologies.

 Nova can be deployed using hypervisor technologies such as KVM, VMware, LXC, XenServer, etc.

Image Service (Glance)

 OpenStack image service offers storing and retrieval of virtual machine disk images.

 OpenStack Compute makes use of this during VM provisioning.

 Glance has client-server architecture which allows querying of virtual machine image.

 While deploying new virtual machine instances, Glance uses the stored images as templates.

 OpenStack Glance supports VirtualBox, VMWare and KVM virtual machine images.

Object Storage (Swift)

 OpenStack Swift creates redundant (repetition), scalable data storage to store petabytes of accessible data.

 The data can be included,retrieved and updated.

 It has a distributed architecture, providing greater redundancy, scalability, and performance, with no central point of control.

 It helps organizations to store lots of data safely, cheaply and efficiently.

Dashboard (Horizon)

 OpenStack Horizon is a web-based graphical interface that cloud administrators and users can access tomanage OpenStack compute, storage and networking services.

 To service providers it provides services such as monitoring,billing, and other management tools.

Identity Service (Keystone)

 Provides an authentication and authorization service for other OpenStack services. Provides a catalog of OpenStack Services.

 Keystone provides a central list of users, mapped against all the OpenStack services, which they can access.

 Keystone supports various forms of authentication like standard username & password credentials.

Networking (Neutron)

 Neutron provides networking capability like managing networks and IP addresses for OpenStack.

 OpenStack networking allows users to create their own networks and connects devices and servers to one or more networks.

 Neutron also offers an extension framework, which supports deploying and managing of other network services such as virtual private networks (VPN), firewalls, load balancing, and intrusion detection system (IDS)

Block Storage (Cinder)

- Orchestrates multiple composite cloud applications by using templates.
- It creates and manages service that provides persistent data storage to cloud computing applications.
- Provides persistent block storage to running virtual machine.
- Cinder also provides a self-service application programming interface (API) to enable users to request and consume storage resources.
- A cloud user can manage their storage needs by integrating block storage volumes with Dashboard and Nova.
- It is appropriate for expandable file systems and database storage.

Telemetry (Ceilometer)

- It provides customer billing, resource tracking, and alarming capabilities across all OpenStack core components.

Orchestration (Heat)

- Heat is a service to orchestrate (coordinates) multiple composite cloud applications using templates.

Workflow (Mistral)

- Mistral is a service that manages workflows.
- User typically writes a workflow using workflow language and uploads the workflow definition.
    - The user can start workflow manually.

Database (Trove)

- Trove is Database as a Service for OpenStack.
- Allows users to quickly and easily utilize the features of a database without the burden of handling complex administrative tasks.

Elastic map reduce (Sahara)

- Sahara is a component to easily and rapidly provision Hadoop clusters.

- Users will specify several parameters like the Hadoop version number, the cluster topology type, node flavor details (defining disk space, CPU and RAM settings), and others.

Bare metal (Ironic)

- Ironic provisions bare metal machines instead of virtual machines.

Messaging (Zaqar)

- Zaqar is a multi-tenant cloud messaging service for Web developers.

Shared file system (Manila)

- Manila is the OpenStack Shared Filesystems service for providing Shared Filesystems as a service.
- Allows to create, delete, and give/deny access to a file.

DNS (Designate)

- Designate is a multi-tenant API for managing DNS.

Search (Searchlight)

- Searchlight provides advanced and consistent search capabilities across various OpenStack cloud services.

Key manager (Barbican)

- It provides secure storage, provisioning and management of secret data.
- This includes keying material such as Symmetric Keys,Asymmetric Keys and Certificates.

Root Cause Analysis (Vitrage)

- Vitrage is the OpenStack RCA (Root Cause Analysis) service for organizing, analyzing and expanding OpenStack alarms & events, yielding insights regarding the root cause of problems and deducing their existence before they are directly detected.

Rule-based alarm actions (Aodh)

- This alarming service enables the ability to trigger actions based on defined rules against an event data collected by Ceilometer.

5.5 Federation in the cloud

Inter cloud:

- The Inter-Cloud is an interconnected global "cloud of clouds" and an extension of the Internet "network of networks" on which it is based.
- Inter-Cloud computing is interconnecting multiple cloud providers' infrastructures.
- The main focus is on direct interoperability between public cloud service providers.
- To provide cloud services as utility successfully, interconnected clouds are required.
- Interoperability and portability are important factors.
- The limitations of cloud are that they have limited physical resources.
- If a cloud has exhausted all the computational and storage resources, it cannot provide service to the clients.
- The Inter-Cloud environment provides benefits like diverse Geographical locations, better application resilience and avoiding vendor lock-in to the cloud client.
- Benefits for the cloud provider are expand-on-demand and better service level agreements (SLA) to the cloud client.

Types of Inter-Cloud

- ✓ Federation Clouds
- ✓ Multi-Cloud

Federation Clouds

- ☐ A Federation cloud is an Inter-Cloud where a set of cloud providers willingly interconnect their cloud infrastructures in order to share resources among each other.
- ☐ The cloud providers in the federation voluntarily collaborate to exchange resources.
- ☐ This type of Inter-Cloud is suitable for collaboration of governmental clouds.
- ☐ Types of federation clouds are Peer to Peer and Centralized clouds.

Multi-Cloud

- ☐ In a Multi-Cloud, a client or service uses multiple independent clouds.
- ☐ A multi-cloud environment has no volunteer interconnection and sharing of the cloud service providers' infrastructures.
- ☐ Managing resource provisioning and scheduling is the responsibility of client or their representatives.
- ☐ This approach is used to utilize resources from both governmental clouds and private

cloud portfolios.

- Types of Multi-cloud are Services and Libraries

Cloud Federation

- Provides Federated cloud ecosystem by connecting multiple cloud computing providers using a common standard.
- The combination of disparate things, so that they can act as one.
- Cloud federation refers to the unionization of software infrastructure and platform services from disparate networks that can be accessed by a client.
- The federation of cloud resources is facilitated through network gate ways that connect public or external clouds like private or internal clouds
- It is owned by a single entity and/or community clouds owned by several co-operating entities.
- Creating a hybrid cloud computing environment.
- It is important to note that federated cloud computing services still relay on the existing of physical data centers.

Benefits of cloud federation:

- The federation of cloud resources allows client to optimize enterprise IT service delivery.
- The federation of cloud resources allows a client to choose best cloud service providers
- In terms of flexibility cost and availability of services to meet a particular business or technological need within their organization.
- Federation across different cloud resources pools allows applications to run in the most appropriate infrastructure environments.
- The federation of cloud resources allows an enterprise to distribute workload around the globe and move data between desperate networks and implement innovative security models for user access to cloud resources

Cloud Federation and Implementation

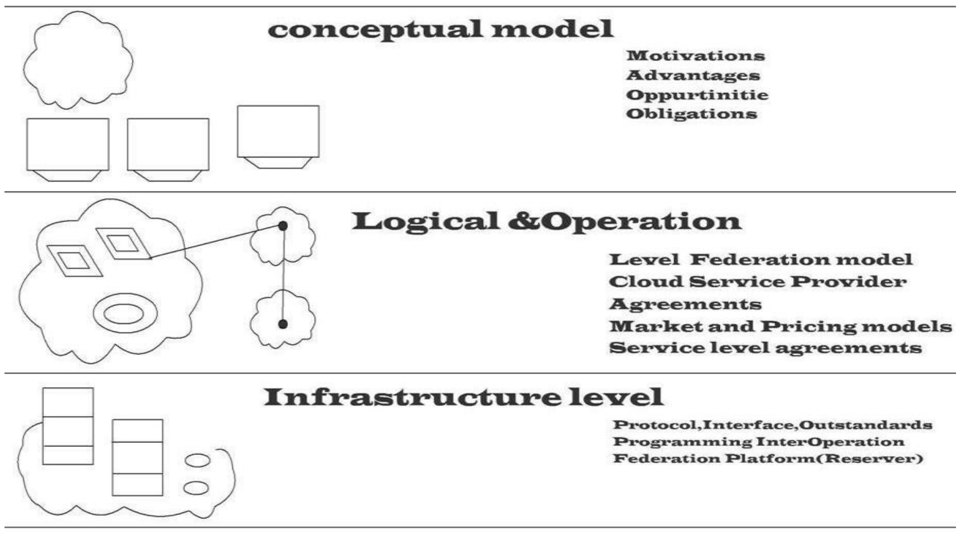
- One weakness that exist in the federation of cloud resources is the difficulty in programming connectivity.
- Connection between a client and a given external cloud provider is difficult as they each possess their own unique network addressing scheme.

- Cloud providers must grant clients the permission to specify an addressing scheme for each server the cloud provider has external to the internet.
- This provides customers to with the ability to access cloud services without the need for reconfiguration when using resources from different service providers.
- Cloud federation can also be implemented behind a firewall which providing clients with the menu of cloud services provided by one or more trusted entities

<u>5.5.2</u> <u>Four Levels of Federation:</u>

- Permissive
- Verified
- Encrypted
- Trusted

Permissive Federation:



- Permissive federation occurs when a server accepts a connection from a peer network server without verifying its identity using DNS lookups or certificate checking.
- The lack of verification or authentication may lead to domain spoofing.
- The unauthorized use of a third party domain name in an email message in order to pretend to be someone else), which opens the door to widespread spam and other abuses.

Verified Federation:

- This type of federation occurs when a server accepts a connection from a peer after the identity of the peer has been verified.
- It uses information obtained via DNS and by means of domain-specific keys exchanged beforehand.

- The connection is not encrypted, and the use of identity verification effectively prevents domain spoofing.
- Federation requires proper DNS setup, and that is still subject to DNS poisoning attacks.
- Verified federation has been the default service policy on the open XMPP since the release of the open-source jabberd 1.2 server.
- XMPP-real time communication protocol uses XML.
- Prevent Address spoofing

Encrypted federation:

- Server accepts a connection from a peer if and only if the peer supports Transport Layer Security (TLS) as defined for XMPP in Request for Comments (RFC) 3920.
- The peer must present a digital certificate.
- The certificate may be self-signed, but this prevents using mutual authentication.
- XEP-0220 defines the Server Dialback protocol, which is used between XMPP servers to provide identity verification.
- Server Dialback uses the DNS as the basis for verifying identity.
- The basic approach is that a receiving server receives a server-to- server connection request from an originating server.
- It does not accept the request until it has verified a key with an authoritative server for the domain asserted by the originating server.
- Server Dialback does not provide strong authentication or trusted federation
- Although it is subject to DNS poisoning attacks, it has effectively prevented most instances of address spoofing on the XMPP network

Trusted federation:

- A server accepts a connection from a peer only under 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 authenticating server.
- The list of trusted root CAs may be determined by one or more factors, such as the operating system, XMPP server software, or local service policy.
- The use of digital certificates results not only in a channel encryption but also in strong authentication.
- The use of trusted domain certificates effectively prevents DNS poisoning attacks.
- But makes federation more difficult, since such certificates have traditionally not been

easy to obtain.

5.5.3Federated Services and Applications:

- Clouds typically consist of all the users, devices, services, and applications connected to the network.
- In order to fully leverage the capabilities of this cloud structure, a participant needs the ability to find other entities of interest.
- Such entities might be end users, multiuser chat rooms, real-time content feeds, user directories, data relays, messaging gateways, etc.
- Finding these entities is a process called discovery.
- XMPP uses service discovery (as defined in XEP-0030) to find the aforementioned entities.
- The discovery protocol enables any network participant to query another entity regarding its identity, capabilities, and associated entities.
- When a participant connects to the network, it queries the authoritative server for its particular domain about the entities associated with that authoritative server.
- Then the authoritative server informs the inquirer about services hosted there and may also detail services that are available but hosted elsewhere.
- XMPP includes a method for maintaining personal lists of other entities, known as roster technology, which enables end users to keep track of various types of entities.

Future of Federation:

- The implementation of federated communications is a precursor to building a seamless cloud that can interact with people, devices, information feeds, documents, application interfaces, and other entities.
- It enables software developers and service providers to build and deploy such applications without asking permission from a large, centralized communications operator.
- Many big companies (e.g. banks, hosting companies, etc.) and also many large institutions maintain several distributed data-centers or server-farms, for example to serve to multiple geographically distributed offices, to implement HA, or to guarantee server proximity to the end user. Resources and networks in these distributed data-centers are usually configured as non-cooperative separate elements.
- Many educational and research centers often deploy their own computing infrastructures, that usually do not cooperate with other institutions, except in same punctual situations (e.g.

in joint projects or initiatives). Many times, even different departments within the same institution maintain their own non-cooperative infrastructures.

- Cloud end-users are often tied to a unique cloud provider, because of the different APIs, image formats, and access methods exposed by different providers that make very difficult for an average user to move its applications from one cloud to another, so leading to a vendor lock-in problem.
- Many SMEs have their own on-premise private cloud infrastructures to support the internal computing necessities and workloads. These infrastructures are often over-sizedto satisfy peak demand periods, and avoid performance slow-down. Hybrid cloud (orcloud bursting) model is a solution to reduce the on-premise infrastructure size, so that it can be dimensioned for an average load, and it is complemented with external resources from a public cloud provider to satisfy peak demands.
- The cloud consumer is often presented with "take-it-or-leave-it standard contracts that might be cost-saving for the provider but is often undesirable for the user”. The commission aims to develop with “stakeholders model terms for cloud computing service level agreements for contracts”.