Unit - 5

Cloud Technologies and Advancements

Hadoop - Map Reduce - Virtual Box - Google App Engine. Programming Environment for Google App Engine - Open Stack - Federation in the Cloud - Four levels of federation - Federated services and Applications - Future of federation.

Hadoop.

* With the evolution of internet and related technologies, the high computational power, large volumes of data storage and faster data processing becomes the basic need for most of the organization

* There is a need to acquire, analyze, process handle and store such a huge amount of data called big data.

* The different Challenges associated with such big data are given as below.

→ Volume
→ Variety
→ velocity
→ veracity.

Volume :

The volume is related to size of big data. According to IBM in the year 2000, 8 lakh petabytes of data were stored in the world. how to deal with such huge big data.

Variety :

The Variety is related to different formats of big data. Now a days most of the data Stored by organizations have no proper Structure called unstructured data. The Challenges here is how to Store different formats of data in databases.

velocity :

The velocity is related to speed of data generation which is very fast. It is a rate at which dat is captured, generated and Shared. The Challenges here is how to react to massive information generated in the time required by the application.

Veracity :

The veracity refers to uncertainty of data. The data Stored in database Sometimes is not accurate or consistent that makes poor data quality. The inconsistent data requires lot of efforts to process such data.

* These Challenges associated with Big data Can be solved using one of the most popular framework provided by Apache is called Hadoop.

* The Apache Hadoop is an open Source Software project that enables distributed processing of large data sets across clusters of Commodity servers using programming models.

Hadoop Ecosystem Components.

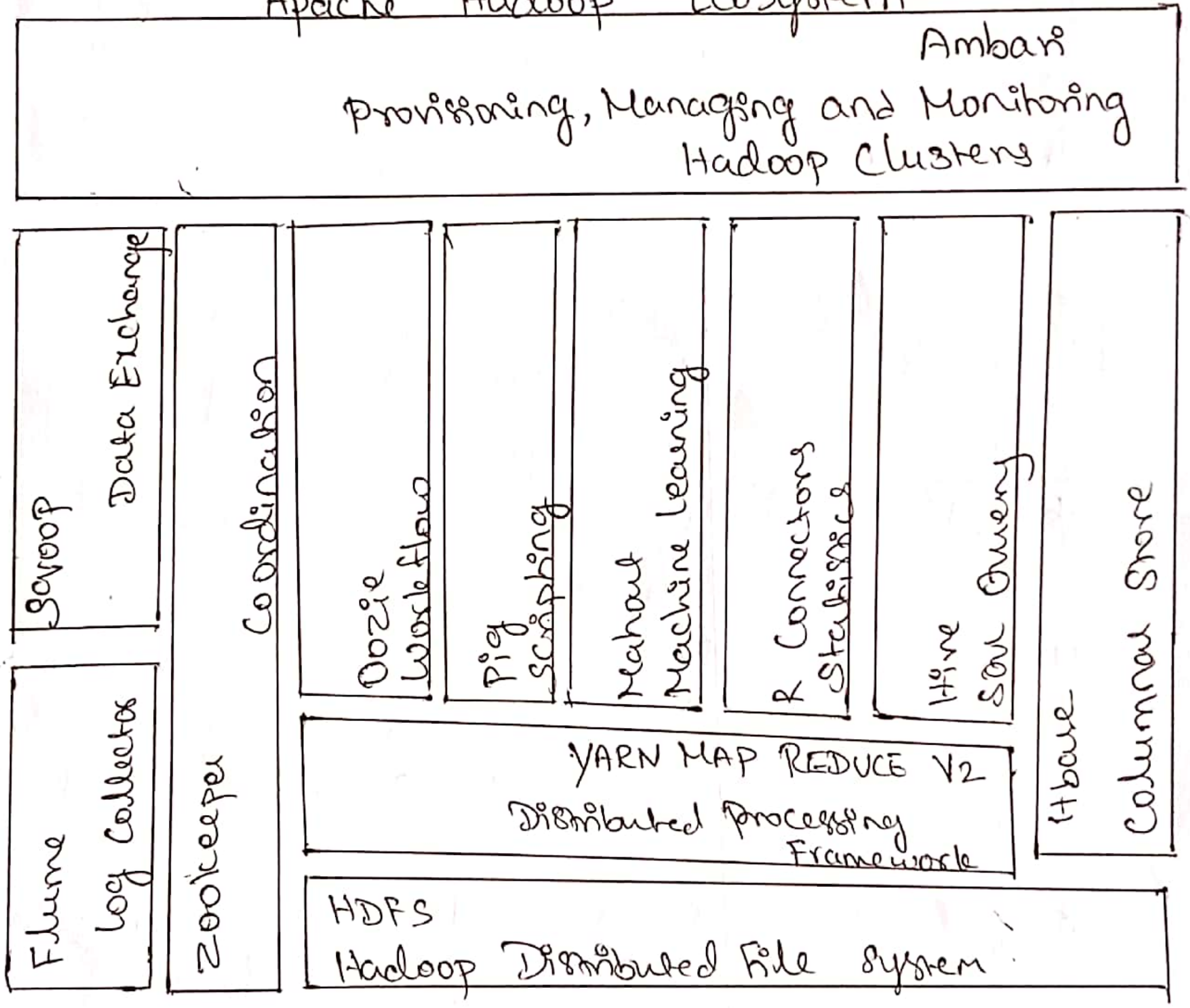
The two main Components in Hadoop architecture.

* Hadoop Distributed File System (HDFS)
* MapReduce.

* The HDFS is a distributed file system inspired by GFS that organizes files and Stores their data on a distributed Computing system

* Map Reduce is the Computation engine running on top of HDFS as its data Storage manager.

Apache Hadoop Ecosystem



S.NO	Name of component	Description.
1.	HDFS	It is a Hadoop distributed file system which is used to split the data in to blocks and stored amongst distributed servers for processing.
2	Map Reduce	It's programming model to process the big data. It mapper extracts data from HDFS and put in to maps while reducer aggregate the results generated by mappers.
3.	Zoo keeper	It is a Centralized service used for maintaining Configuration information with distributed synchronization and Co-ordination.
4.	HBase	It is a Column-oriented database service used as NoSQL solution for big data
5.	Pig	It is a platform used for analyzing the large data sets using a high level language.
6	Hive	It provides data warehouse infrastructure of big data.
7.	Flume	It provides distributed and reliable service for efficiently collecting, aggregating and moving large amount of log data.
8.	SCoop	It is a tool designed for efficiently transferring bulk data.

9.	Mahaout	It provides libraries for Scalable Machine learning algorithms implemented on the top of Hadoop implemented using MapReduce framework.
10	Oozie	It is a workflow Schedular System to manage the Hadoop jobs.
11	Ambari	It provides a Software framework for provisioning, managing and monitoring Hadoop clusters.

Hadoop Distributed File System (HDFS)

The Hadoop Distributed File System is the Hadoop implementation of distributed file system design that hold large amount of data.

Benefits:-

1. It provides Streaming access to file system data
2. It is Suitable for distributed Storage and Processing.
3. It is optimized to support high Streaming read operations with limited set.
4. It supports file operations like read, write delete but append not update.
5. It provides Java API and Command line interfaces to interact with HDFS.

6. It provides different file permissions and authentications for files on HDFS.

7. It uses checksums and digital signatures to manage the integrity of data stored in a file.

8. It has built-in metadata replication so as to recover data during the failure.

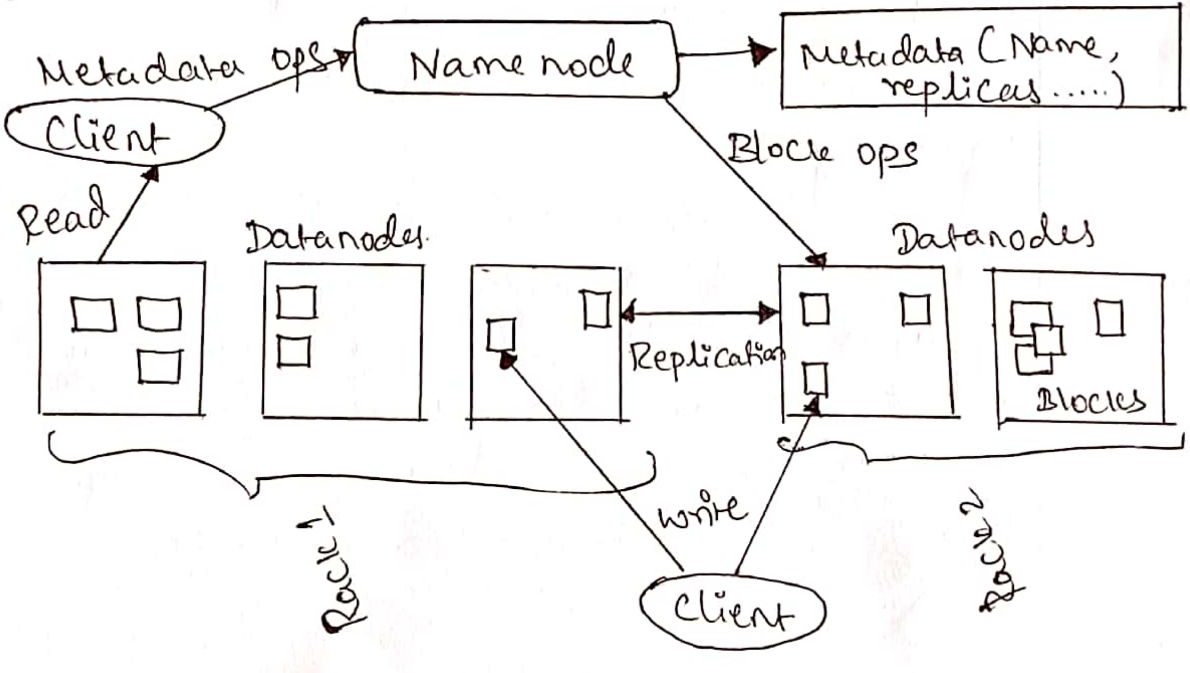
9. It also provides synchronous snapshots to facilitates rolled back during failure.

10. It provides continuous monitoring of name nodes and data nodes based on continuous "heartbeat" communication between the data nodes to the name nodes.

Architecture of HDFS:

* The HDFS follows Master-slave architecture using name node and data nodes.

* It is implemented as block structure file system where files are broken in to block of fixed size stored on Hadoop clusters.



1. Name node :-

→ An HDFS cluster consists of single name node called master server that manages the file system namespaces and regulate access to files by client.

→ It stores all metadata for the file system across the clusters.

→ The name node serves as single arbitrator which is kept in main memory for faster random access.

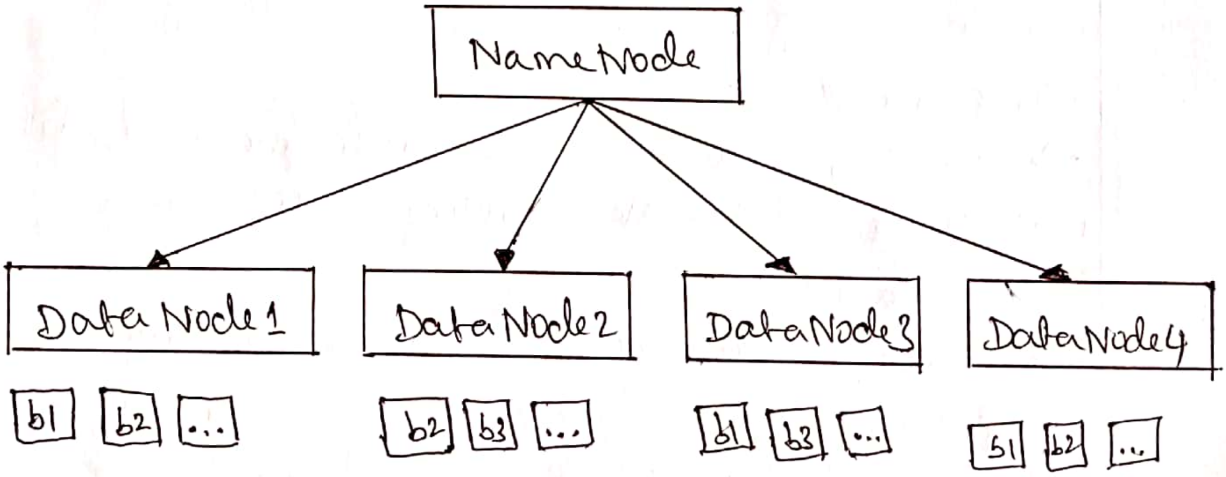
→ The entire file system name space is contained in a file called FsImage stored on name nodes file system.

2. Data Node :-

→ In HDFS there are multiple data nodes exist that manages storage attached to the node that they run on.

→ The data nodes are responsible for handling read / write request from clients.

→ It performs block creation, deletion. and replication upon instruction from name node.



3. HDFS Client:

→ The User applications access the file system using the HDFS client.

→ It supports various operations to read, write and delete files and operations to create & delete directories.

→ The user application does not need to aware that file system metadata and storage are on different servers.

→ when an application reads a file, the HDFS Client first asks the name node for the list of data nodes that host replicas of the blocks of the file.

4. HDFS Blocks:

→ The files in file system are divided in to one or more segments called blocks.

→ The default size of HDFS block is 64MB that can be increase as per need.

→ The Synchronization between name node and data node is done by heartbeats functions which are periodically generated by data node to name node.

→ The job tracker runs on name node like a master while task trackers runs on data nodes like slaves.

→ The job tracker is responsible for taking the request from a Client and assigning task trackers to it with tasks to be performed.

Scanned with CamScanner

Map Reduce

→The MapReduce is a programming model provided by Hadoop that allows expressing distributed computations on huge amount of data.

→ It provides easy scaling of data processing over multiple computational nodes or Clusters.

→ In MapReduce model the data processing primitives used are called mapper and reducer.

Features of MapReduce

* Synchronization : The MapReduce supports execution of concurrent task.

* Data locality : In MapReduce although the data resides on different Clusters, it appears like a local to the Users application.

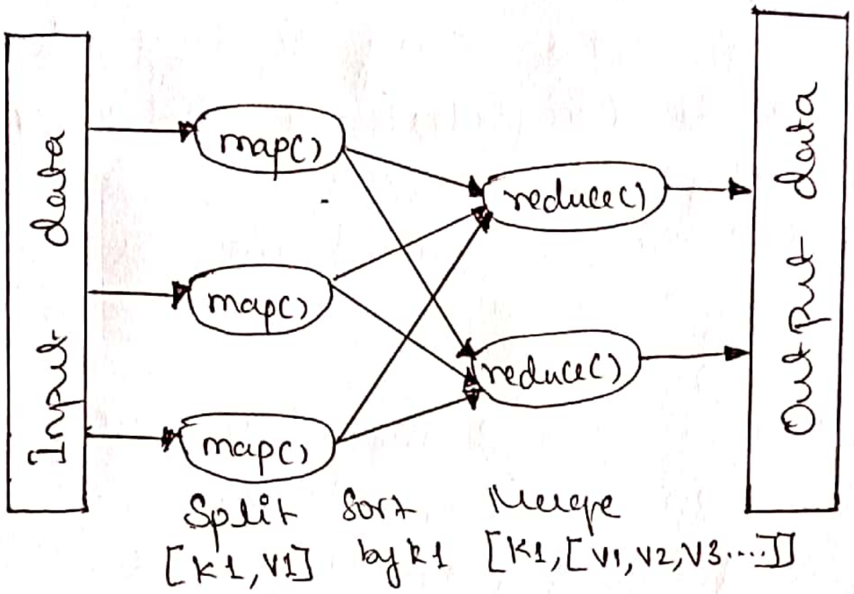
* Error handling : Map Reduce engine provides different fault tolerance mechanisms in case of failure.

* Scheduling : The MapReduce involves map and reduce operations that divide large problems in to Smaller Chunks and those are run in parallel by different Machines.

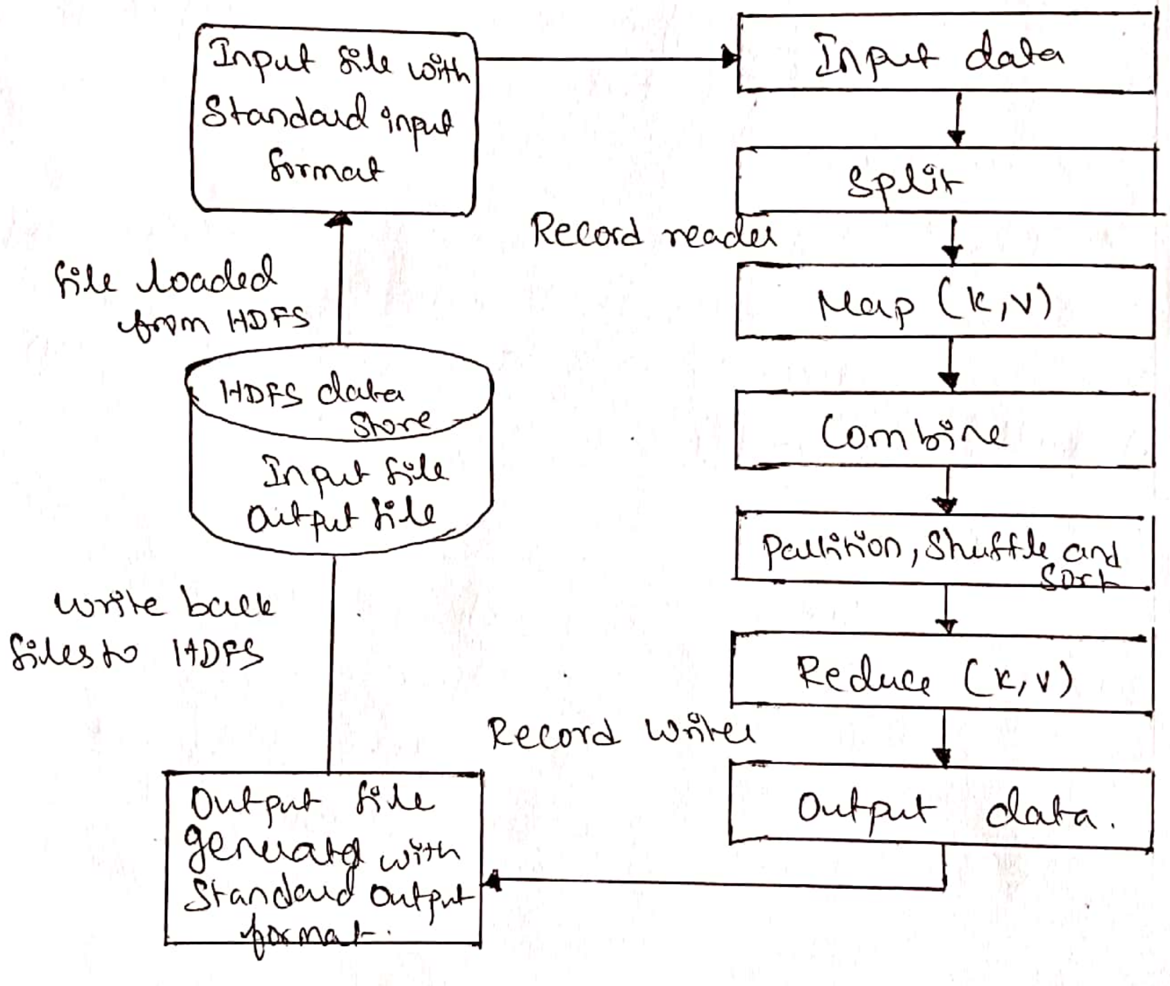
Working of MapReduce framework :-

The unit of work in MapReduce is a job. During map phase the input data is divided in to input splits for analysis where each split is an independent task.

The MapReduce takes a set of input <key, value> pairs and produces a set of output <key, value> pairs by supplying data through map and reduce functions.



The different phases of execution in MapReduce are .



→ Input phase - the large data set in the form of <key, value> pair is provided as a Standard input for MapReduce program.

→ Split phase - reads the input data and divided those in to Smaller Chunks.

→ Map - it extract the relevant data and generate intermediate key value Pairs.

→ Combiner - it is Used with both mapper and reducer to reduce the volume of data transfer it is also known as Semi reducer

→ Shuffle and Sort - The Components of reducer, the Shuffling is a process of partitioning and moving a mapped output to the reducer. each partition is Called Subset.

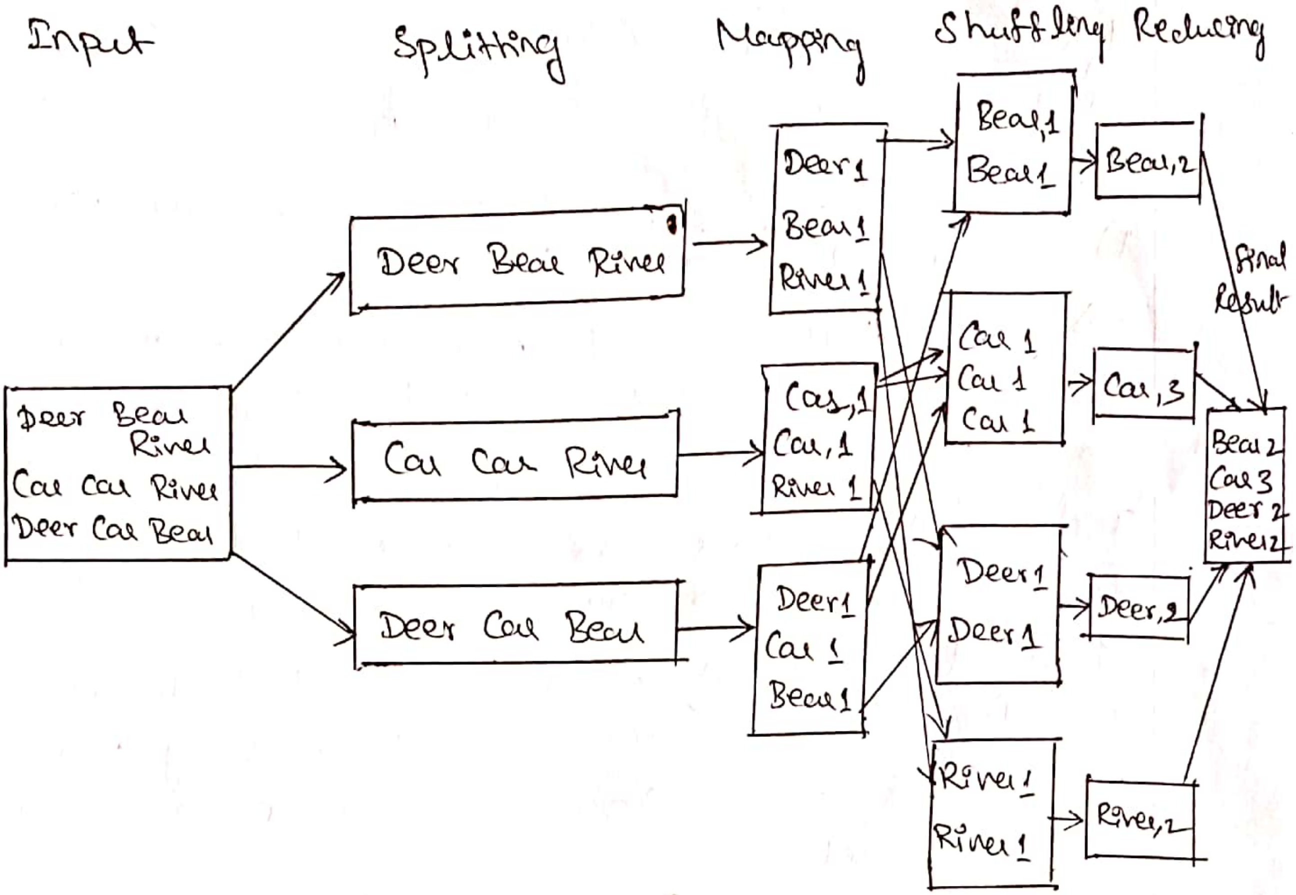
- The Sort Phase is responsible for Sorting the intermediate key on single node automatically before they are presented to the reducer.

→ Reducer - The reducer reduces a set of intermediate values which share unique keys with set of values.

→ Output phase - The output of each MapReduce program is generated with key value pairs written in output file which is written back to the HDFS Store.

-Example of word Count process Using MapReduce with all Phases of execution.

Word Count Process Using MapReduce.



Virtual Box

Virtual Box is an X86 Virtualization Software package, Created by Software Company Innotek.

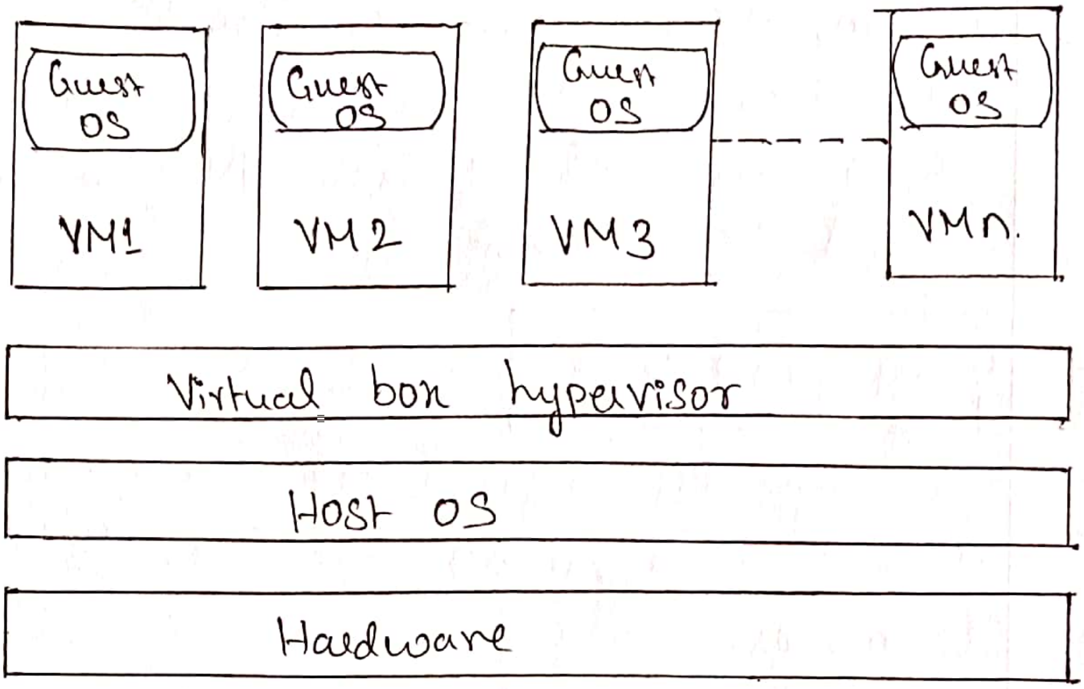
* It is Cross-platform Virtualization Software that allows Users to extend their existing Computer to run multiple Operating systems at the Same time.

* It is ideal for testing, developing, demonstrating and deploying solutions across multiple platforms on single Machine.

* It is Type II hypervisor that can be installed on an existing host Operating system as an application.

* This hosted application allows to run additional operating system as an application inside it known as a Guest OS.

* Each instance of guest OS is called a "Virtual machine"



* It has lightweight extremely fast and Powerful Virtualization engine.

Features :-

1) It supports fully para Virtualized environment along with Hardware Virtualization.

2) It provides device drivers from driver Stack which improves the performance of Virtualized input / output devices.

3) It provides shared folder support to copy data from host OS to guest OS and Vice versa.

4) It has latest Virtual USB Controller support.

5) It facilitates broad range of Virtual network driver Support along with host, bridge and NAT modes.

6) It Supports Remote Desktop protocol to Connect windows Virtual machine remotely on a thin, thick or mobile Client Seamlessly.

7) It has Support for Virtual DISK formats which are used by both VMware and Microsoft Virtual PC hypervisors.

Google App Engine

Google App Engine (GAE) is a platform as a Service Cloud Computing model that Supports many Programming languages.

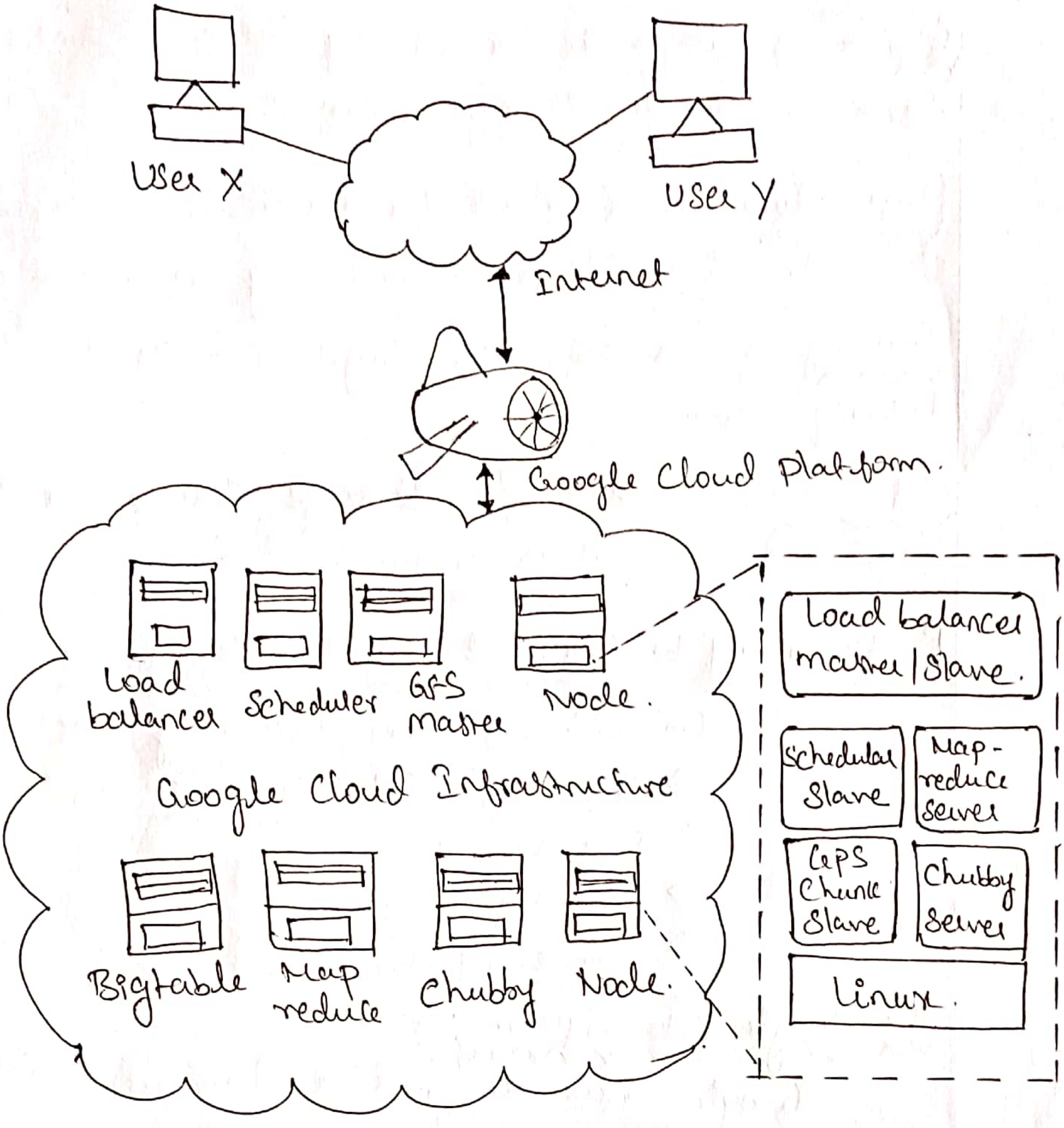
* GAE is a Scalable runtime environment mostly devoted to execute web applications.

* It supports developers to use readymade platform to develop and deploy web applications using development tools, runtime engine, databases and middleware solutions.

* It supports languages like Java, Python, .NET PHP, .Ruby, Node.js and GO

* GAE enables Users to run their applications on a large number of data Centers associated with Google's Search engine Operations.

Functional architecture of Google Cloud platform.



→ The infrastructure for google Cloud is managed inside .dataCenter.

→ All the Cloud Services and applications on Google runs through Servers inside dataCenters.

→ The infrastructure for GAE Composed of four main Components.

- * Google file System (GFS)
- * Map Reduce
- * Big Table
- * Chubby

GFS - It is used for storing large amounts of data on google Storage Clusters.

MapReduce - It is used for application program development with data processing on large clusters.

Chubby - It is used as distributed applications locking services.

Big Table - It offers a storage service for accessing Structured as well as unstructured data.

Programming Environment for Google App Engine.

It consists of the following sections.

- → The Google file System (GFS)
- → Big Table
- → Chubby
- → Google APIs.

The google file System :-

Google has designed a distributed file system, named GFS, for meeting its exacting demands off processing a large amount of data.

1) Automatic recovery from component failure on a routine basis.

2) Efficient Storage support for large-sized files

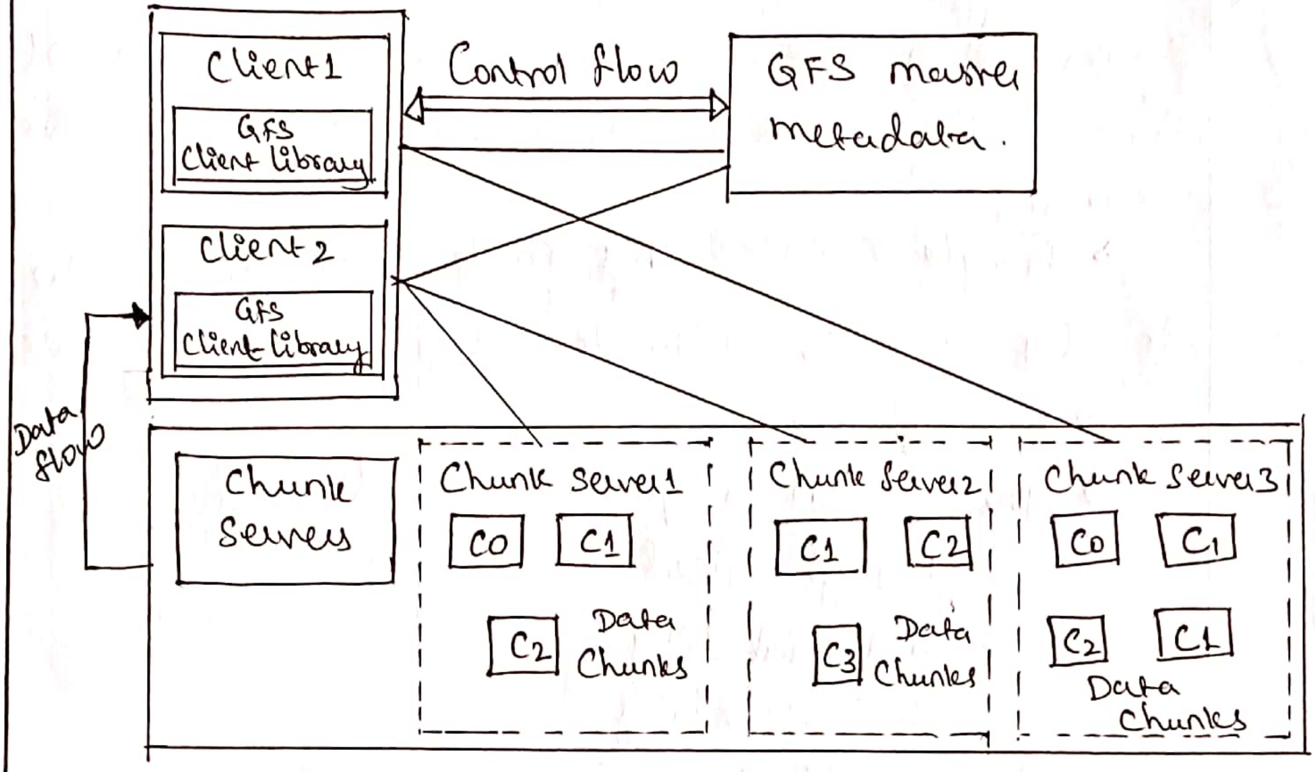
3) workloads that mainly consist of two large Streaming reads and small random reads.

4) The System supports Small writes without being inefficient.

5) Semantics that are defined well are implemented

6) Atomicity is maintained with the least Overhead due to Synchronization.

7) Provisions for Sustained bandwidth is given priority rather than a reduced latency.



Features :-

* Large-Scale data processing and Storage support
* Normal treatment for Components that stop responding.
* Optimization for large-sized files.
* fault tolerance by Constant monitoring, data replication, and automatic recovering.

* Data corruption detections at the disk.

* High throughput for Concurrent readers and writers.

* Simple designing of the master that is Centralized and not ~~bbottle~~ bottlenecked.

Big Table:-

→Googles Big table is a distributed Storage system that allows Storing huge Volumes of Structured as well as unstructured data on Storage mediums.

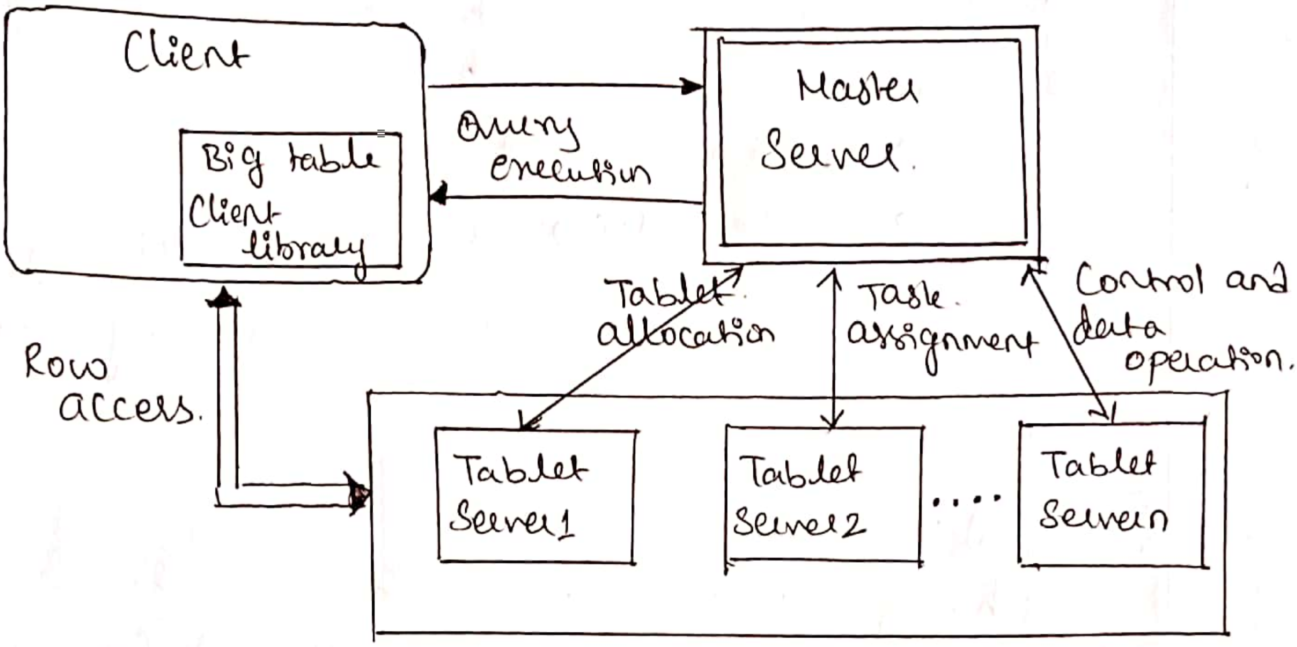
→ Google created Big Table with an aim to develop a fast, reliable, efficient and Scalable Storage system.

→ The design requirements of Big Table are.

1. High Speed
2. Reliability
3. Scalability
4. Efficiency
5. High performance.
6. Examination of Changes. that take Place in data over a period of time.

→ Big Table is a Popular, distributed data Storage system that is highly Scalable and Selfmanaged.

→ It involves thousands of servers, terabytes of data storage for in-memory operations, millions of read/write requests by users in a second and petabytes of data stored on disks.



→ It is composed of three entities, namely

* Client

* Big table master.

* Tablet servers.

→ Big table are implemented over one or more clusters.

→ The client application uses libraries to execute Big table Queries on the master server.

→ Big table is broken up into one or more slave servers called tablets.

→ Master server is responsible for allocating tablets to tasks, clearing garbage collections and Monitoring the performance of tablet servers.

* The common operations are
  -> Creation & deletion of tables
  -> Creation & deletion of column families
  -> writing or deleting cell values
  -> Accessing data from rows.

* The functions that are used for atomic write operations are as follows.
  -> set() is used for writing cells in a row.
  -> DeleteCells() is used for deleting cells from a row.
  -> DeleteRow() is used for deleting the entire row.

Chubby:-

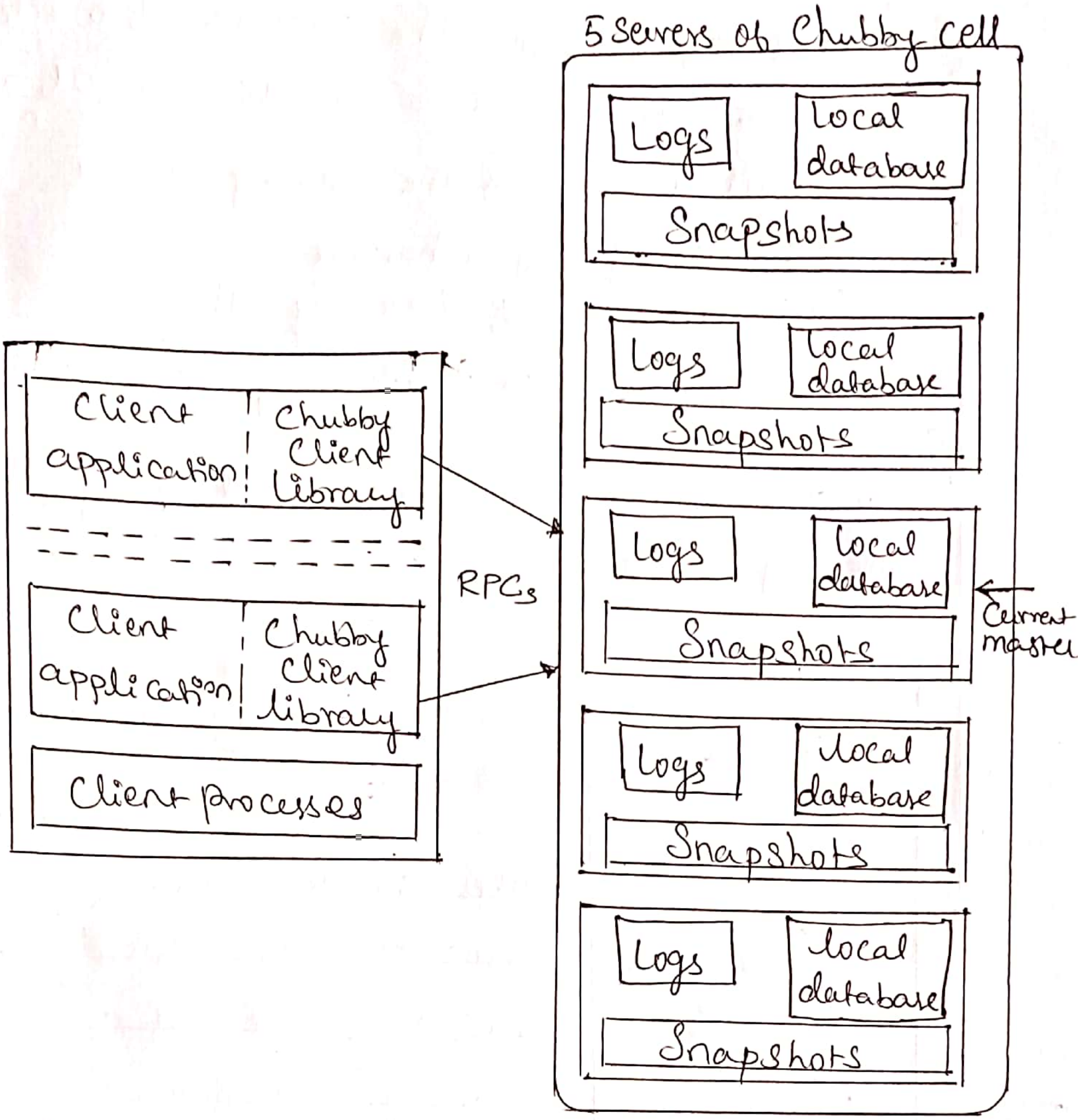
Chubby is the crucial service in the Google infrastructure that offers storage and coordination for other infrastructure services such as GFS and Bigtable.

* The Chubby interface is similar to the interfaces that are provided by distributed systems with advisory locks.

* Aim of Chubby is to efficiently handle a large set of clients by providing them a highly reliable and available system.

* The Chubby architecture involves two primary components, namely server and client library.

* Both components communicate through a Remote ~~Pro~~cedure Call (RPC)



* The library has a Special purpose i.e linking the Client againt the Chubby cell.

* A Chubby cell contains a small set of servers.

* The servers are also called replicas, and usually, five servers are used in every cell.

* The Chubby contains events which is used to do following activities.

a) Modification in the Contents of a file.

b) Addition, removal or modibication of a Child node.

c) Failing over of the Chubby Master

d) Invalidity of a handle.

e) Acquisition of lock by others.

f) Request for a Conflicting lock from another Client.

* The Chubby is implemented Using the following APIs :

API	Description.
Open	Opens the file or directory and returns a handle
Close	Closes the file or directory and returns the associated handle.
Delete	Deletes the file or directory
ReadDir	Returns the contents of a directory
Set Contents	Writes the Contents of a file.
Get Stat	Returns the metadata.
GetContent And Stat	Writes the file contents and return metadata
Acquire	Acquires a lock on a file.
Release	Release a lock on a file :

Google APIs :-

Google developed a set of Application Programming Interfaces (APIs) that can be used to communicate with Google services.

Open Stack

OpenStack is an open-source cloud operating system that is increasingly gaining admiration among data centers.

The main objective of openStack is to provide a cloud computing platform

* Global
* Open-source
* freely available
* Easy to use
* Highly and easily scalable
* Easy to implement
* Interoperable.

Components of OpenStack :-

OpenStack consists of many different components.

-> Nova :- This is one of the primary services of OpenStack, which provides numerous tools for the deployment and management of a large number of virtual machines.

-> Swift :- Swift provide storage services for storing files and objects.

-> Cinder :- This components provides block storage to Nova Virtual Machines.

-> Glance :- Glance is openStack image service component that provides virtual templates of hard disks.

-> Neutron :- This component of OpenStack provides networking as a service, load Balancer as a service and firewall-as a service.

-> Heat :- It is the Orchestration component of openstack.

-> keystone :- This component provides identity management in openstack.

-> Horizon :- This is a dashboard of OpenStack which provides a graphical interface.

-> Ceilometer :- This component of OpenStack provisions meters and billing models for users of the Cloud services.

-> Trove :- It provides Database-as-a service. It provisions relational databases and big data engines.

-> Sahara :- It provides Hadoop to enable the management of data Processors.

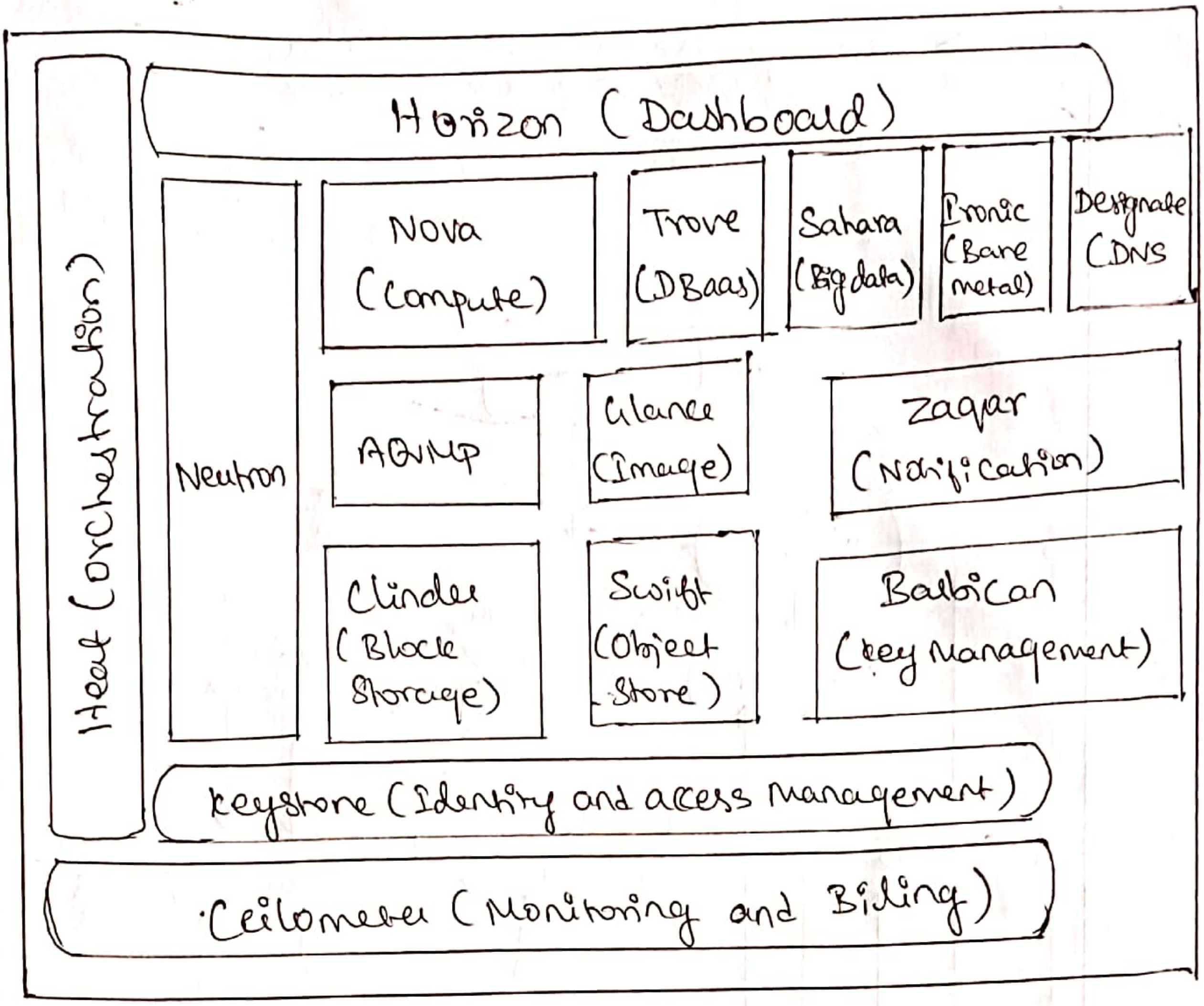
-> Zaqar :- This component allows messaging between distributed application components.

-> Ironic :- Ironic provisions bare-metals, which can be used as a substitute to VMs.

-> Dessignate :- It offers DNS services analogous to Amazon's Route 53.

-> Barbican :- Barbican is the key management Service open stack that is comparable to KMS from AWS.

-> AMQP :- Advanced Message Queue protocol and is a messaging Mechanism used by OpenStack.



Features and Benefits :-

* Compatibility :- OpenStack supports both private and public cloud and is very easy to deploy and manage.

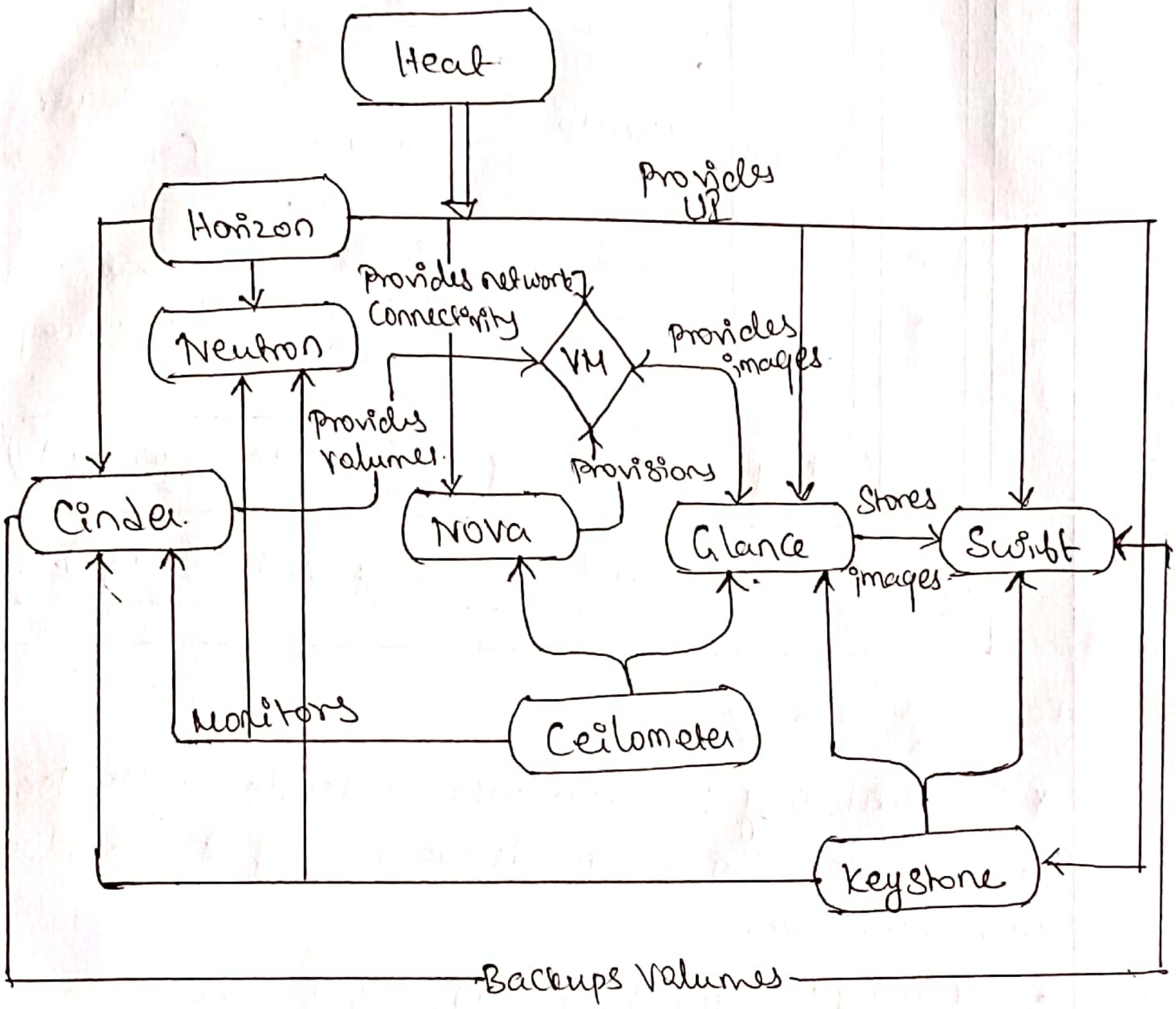
* Security :- OpenStack addresses the security concerns, which are the top most concerns for most organisations.

* Real time visibility :- OpenStack provides real-time client visiblity to administrators.

* Live upgrades :- OpenStack has enabled upgrading systems while they are running by requiring only individual components to shutdown.

Conceptual OpenStack Architecture.

· It is relationships among different Services and between the Services and VMs.



→ Horizon is providing user interface to the users or administrators

→ keystone is providing authentication to the user by mapping the Central directory.

→ Ceilometer is monitoring, the Open Stack Cloud for the purpose of Scalability, billing, benchmarking.

→ Glance is registering Hadoop images, providing image services to OpenStack and allowing retrieval and Storage of disk images

→ Swift is responsible for providing reading service and Storing data in the form of objects and files.

→ Cinder offers permanent block Storage or volumes to VMs also stores backup volumes in Swift.

→ Trove stores backup databases in Swift and boots databases instance.

→ Neutron enables network connectivity for VMs

→ Heat which is the orchestration Component of openStack

→ Sahara is used to offers a simple means of of providing a data processing framework to the Cloud users.

Modes of operations of OpenStack :-

OpenStack majorly operates in two modes.

* Single host
* multi host

Single host mode of operation is that in which the network services are based on a central server.

Multi host operation mode, the compute nodes also individually host floating IPs and security groups.

Federation in the Cloud

As many of the Cloud Computing environments present difficulties in Creating and managing decentralized provisioning of Cloud services along with maintaining Consistent Connectivity between untrusted components and fault-tolerance. To overcome such Challenges the federated Cloud ecosystem is introduced by associating multiple Cloud computing providers using a Common Standard.

Federation in Cloud by using IETF (Internet Engineering Task force) Standard protocols for interdomain federation Called

→ Jabber XCP (Extensible Communications Platform)
→ XMPP (Extensible Messaging and presence protocol)

Jabber XCP :-

* Instant Messaging allows users to exchange messages that are delivered synchronously.

* It is a highly programmable presence and messaging platform.

* It supports the exchange of information between applications in real time.

* It supports multiple protocols such as session Initiation Protocol for Instant Messaging and presence leveraging Extensions (SIMPLE) and Instant messaging and presence service (IMPS)

XMPP (Extensible Messaging and presence protocol)

* It is an open Standard for instant messaging.

* The protocol messages are formatted using XML

* In XMPP, users are identified by a Username and domain name.

* web services play an important role in provisioning the resources and services.

* protocols used by Current Cloud services like SOAP or HTTP based.

Advantage

1) It is decentralized and supports easy two way Communication.

2) It doesn't require polling for synchronization

3) It has built-in publish Subscribe.

4) It works on XML based open Standards

5) It is perfect for Instant Messaging features.

6) It is efficient and scales up to millions of Concurrent users on a single service.

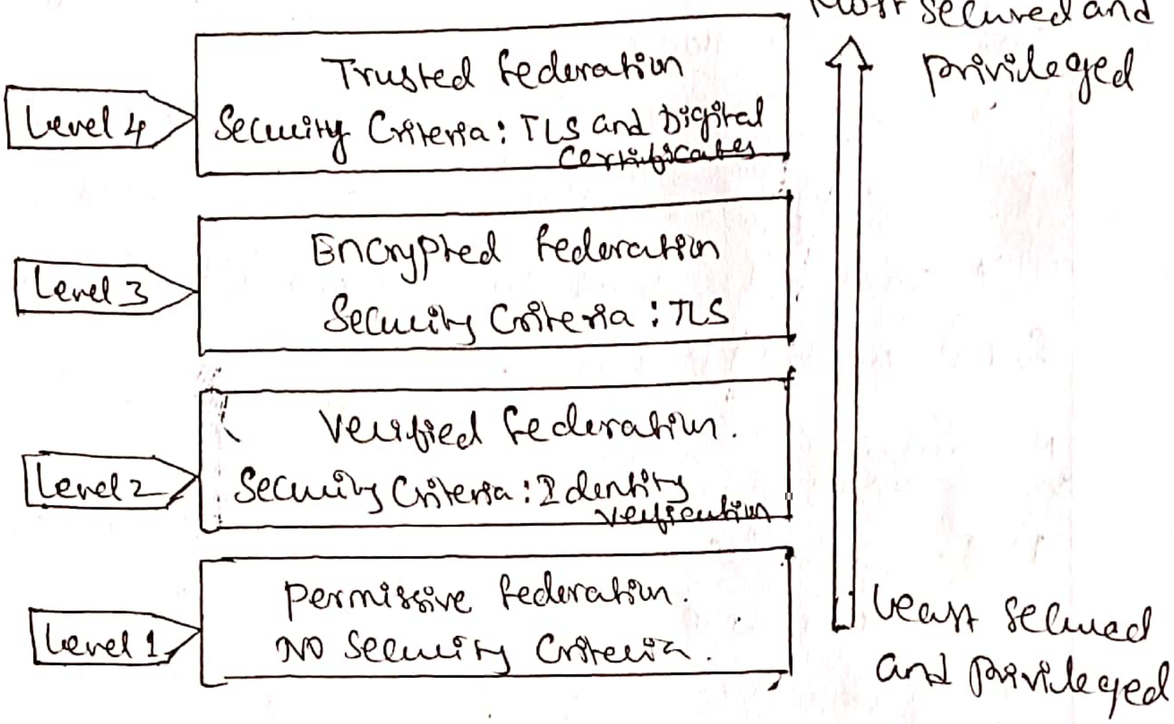
7) It supports world wide federation models

8) It provides strong security using Transport layer security (TLS)

9) It is flexible and extensible.

Four levels for Federation

In a real time, federation defines the way how XMPP servers in different domains exchange the XML based messages.



Level 1: Permissive federation:-

The Permissive federation is the lowest level of federation where server accepts a connection from a peer network server.

It has least security Mechanisms due to which it makes the way for widespread spam and other abuses.

Permissive federation was the only solution to work with web application but with the arrival of the Open Source.

Level 2 : Verified federation

The Server accepts a connection from a peer network Server only when the identity of peer is verified or validated.

Peer verification is the minimum criteria to run verified federation.

It utilizes information acquired by DNS for domain-specific keys exchange in advance.

Level 3 : Encrypted federation

The Server accepts a connection from peer network servers if and only if peer supports the Transport layer Security (TLS)

The TLS is the advancement to secure Sockets layer (SSL) which was developed to make secure communications over HTTP.

Level 4 : Trusted federation

The Encrypted federation is the top most level of federation, a server accepts a connection from peer network server if and only if the peer supports TLS and the speer can present a digital certificate issued by a root

Certification Authority (CA) that is trusted by the authenticating server.

The trusted root CAs are identified based on one or more factors like their Operating System environment, XMPP server Software, or local service policy.

Federated Services and Applications:-

The server to server federation is required to head toward building a constant real time communication in the cloud.

The cloud comprises the considerable number of clients, devices, services and applications connected to the network.

XMPP utilizes service discovery to find the stated entities.

As XMPP learned from past problems, it helps email systems to prevent address spoofing, inline scripts, unlimited binary attachments and other attack tactics.

The use of point-to-point federation can avoid problem that occured in traditional multi hop federation as it can restricts injection attacks, data loss and unencrypted intermediate links.

The Future of Federation :-

* The accomplishment of federated communications is an antecedent to build a consistent cloud that can interact with individuals, devices, information feeds, documents, application interfaces and so on..

* The procedure of server to server federation with the end goal of interdomain communication has assumed an enormous role in the success of XMPP.

* It depends on a small set of simple but incredible Mechanisms for domain checking and security to produce verified, encrypted and trusted connections between any two deployed servers.