Chapter-2 & 3

Apache hadoop

Apache hadoop is open source framework which is desined for (used for) distributed storing and Parallel Processing of Large dataset across commodity hardware computers.

Key Features of hadoop: (SCPFF)

→ Scalability - (handle massive dataset)

→ Cost effective - (Runs in local commodity hardware)

→ Parallel Processing → Effectively Process large dataset in distributed manner.

→ Fault tolerent → Create replication of each data.

→ Flexibility → Supports different types of data (structure, unstructured semistructure)

~~Component of hadoop:~~

Component of ~~hardware~~ Hadoop.

1) Hadoop Common
2) HDFS
3) YARN
4) Map Reduce

1) Hadoop Common

→ Hadoop common is core module which provide essential tools, libraries & JAVA API's for hadoop functionality.

→ It supports other hadoop components like HDFS, YARN & Map Reduce.

→ It provides JAR files & scripts for running hadoop

→ It helps to communicate seamlessly between nodes.

2) HDFS

→ It is storage layer of hadoop designed to store large dataset across a cluster of commodity hardware.

→ It is highly Fault tolerent using data replication across nodes.

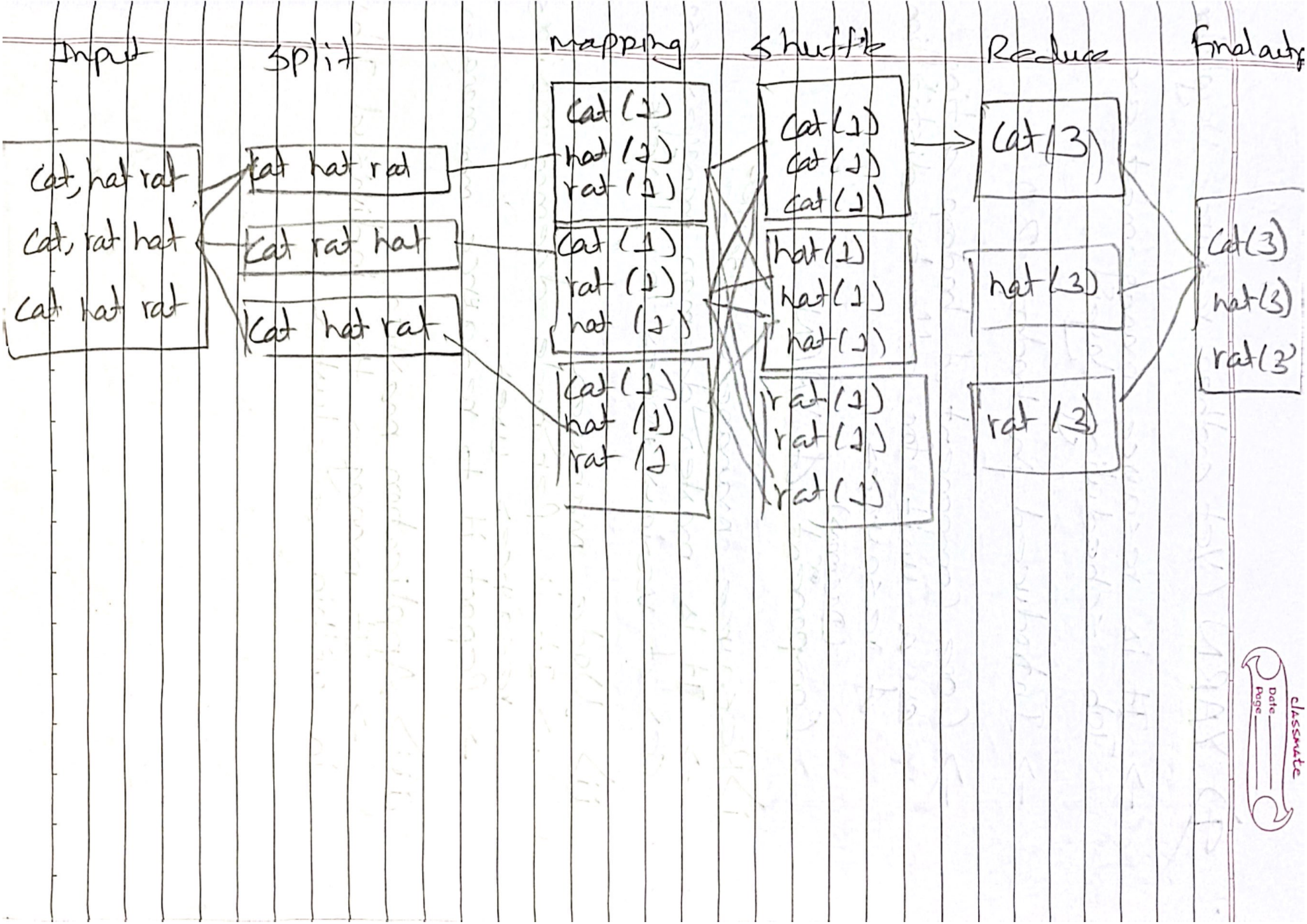
→ Optimize for high ~~throughput~~ troughput making easy for access & Processing.

→ It uses write once read many (~~WORM~~) (WORM) model, ideal for log Processing, web analytics.

3) Map Reduce (IISMS RF)

→ Map reduce is Programming framework designed for Parallel Processing.

→ Phases of map reduce
i) Input
ii) Split
iii) Mapping
iv) Shuffle
v) Reduce
vi) Final output


Input
Split
Mapping
Shuffle
Reduce
final outp
cat, hat rat
cat, rat hat
cat hat rat
cat hat rat
cat rat hat
cat hat rat
cat (1)
hat (1)
rat (1)
cat (1)
rat (1)
hat (1)
cat (1)
hat (1)
rat (1)
cat (1)
cat (1)
cat (1)
hat (1)
hat (1)
hat (1)
rat (1)
rat (1)
rat (1)
cat (3)
hat (3)
rat (3)
cat (3)
hat (3)
rat (3)

7) YARN (Yet another resource Negotiator)

→ It is resource management and Job scheduling layer.

→ Introduced in hadoop 2.0.

→ Core Component of hadoop acting as a unit for effective distribution of resources.

(Daemons) or Component of YARN

i) Resource manager (services)
→ It is master daemon manages resources of various nodes in clusters.

ii) Node manager
→ It is per node daemon monitors resources uses.
→ Report it to resource manager.

iii) Application master
→ It manages the individual Jobs in within a cluster.

Running Hadoop

→ It is the process of deploying & executing hadoop component on system or clusters of commodity hardware to perform big data Processing & storage task.

→ It involves starting the hadoop daemons & executing Job using map reduce & YARN

→ Running hadoop means launching it's services so we can use it for distributed data storage and parallel processing.

~~# Modes of Running hadoop (Hadoop Configuration~~

→

Modes of Running Hadoop (Hadoop Configuration)

1) Standalone mode
   → It runs on a single machine for testing or development.
   → It doesnot use HDFS, uses local file system
   → All process runs in single JVM (Java virtual machine)
   → It doesnot have replication of data.

2) Pseudo distributed mode
   → It simulates multinode cluster in one machine.
   → HDFS is enable with replication factor 1.
   → Each daemon runs on its own JVM.

3) Fully distributed mode
   → Runs hadoop across multiple nodes in a different real clusters.
   → HDFS is enable with replication factor 3 (can be customize according to requirement)

→ It has Production ready Setup.

→ Data blocks are distributed across nodes.

Advanced hadoop Modes

1) High Availability (HA) Mode.

→ Eliminates SPOF (Single Point of failure) by Providing a redundancy for Critical components like namenode resource Manager.

Key Components

Name Node HA

i) Active NameNode
   → Handles clients request

ii) Standby NameNode
   → Takes over during failure of active NameNode.

iii) Journal NameNode
   → Stores metadata synchronization between NameNodes.

ii) ZooKeeper
-> Coordinates NameNode failovers & NameNode health.

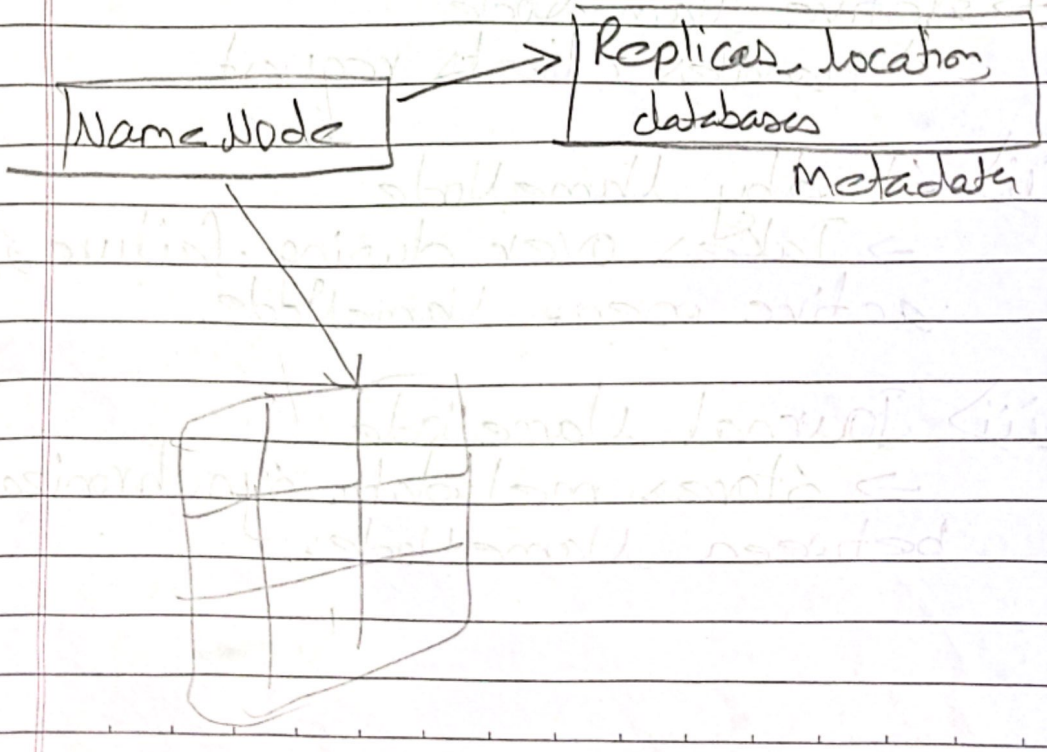
YARN High Availability

i) Active Resource Manager
-> Allocates resources, Job scheduli

ii) Standby Resource Manager
-> Taker over if active RM fails.

iii) ZooKeeper.
-> Coordinates ~~failov~~ failover and monitors Resource Manager(RM)

2) Federation Mode / HDFS federation



→ It scales HDFS metadata horizontally by splitting it across multiple independent NameNode, each managing a separate Namespace

Components

i) NameSpaces
→ Each NameNode manages it's own Namespace

ii) Blocks Pools
→ Data Blocks are stored in pool directries on DataNodes

iii) ViewFS
→ Client uses a unified Namespace to access federated NameSpace.

~~When to use~~

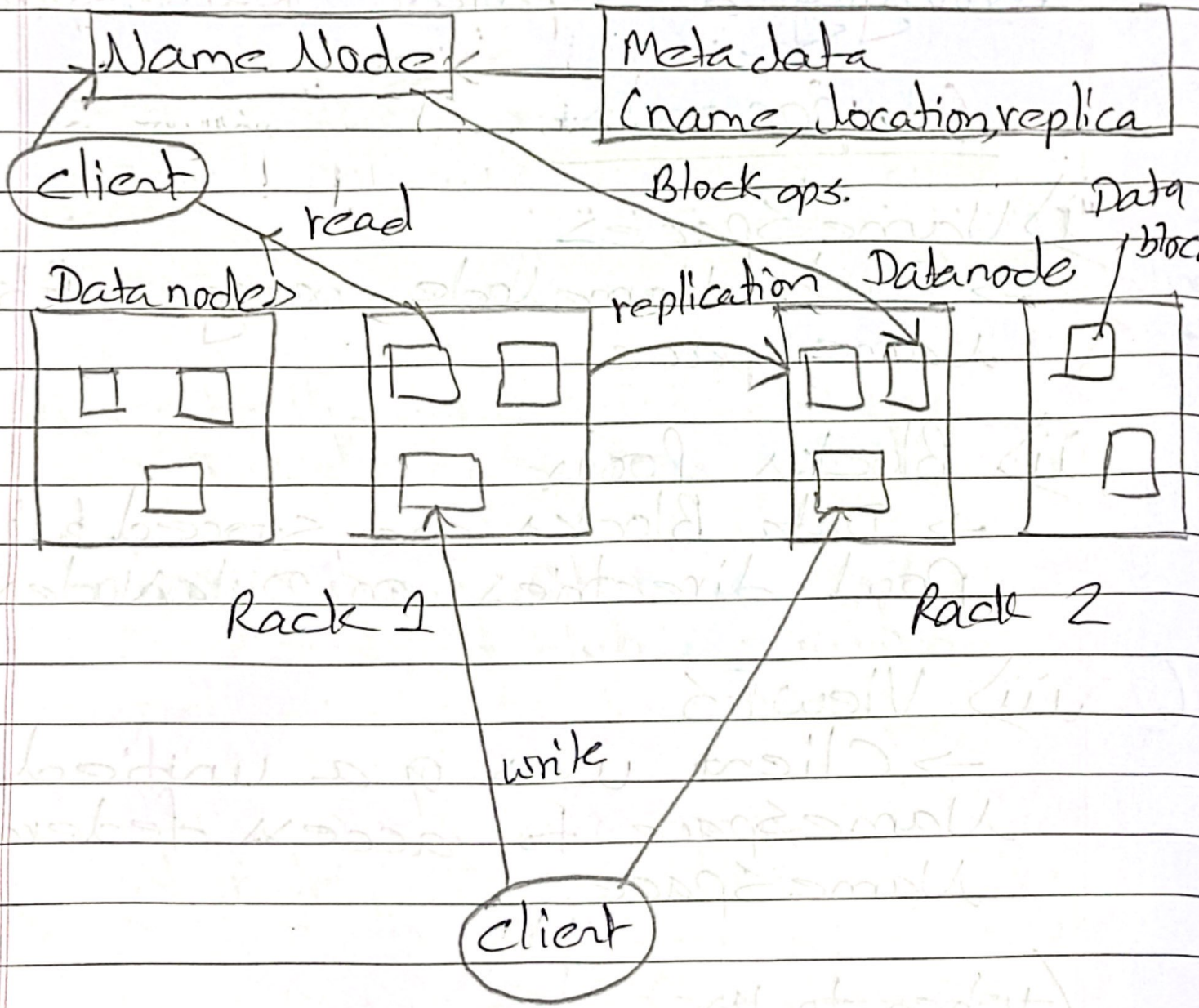
When to use Federation Mode

→ Massive metadata

→ Geographical Distribution

→ Multi-tenancy.

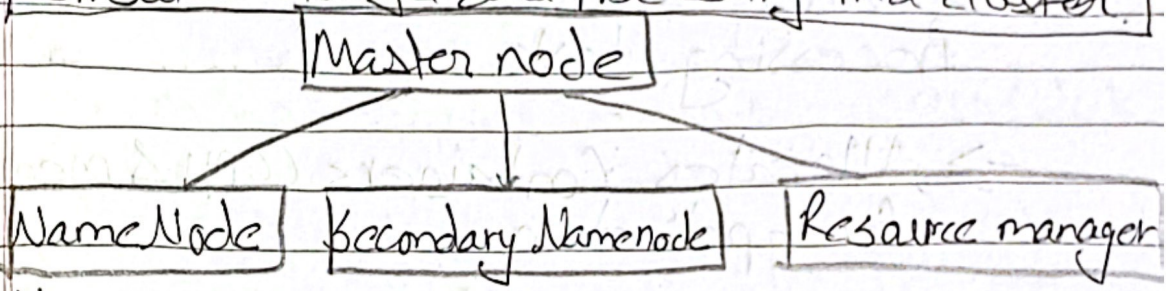
HDFS Architecture



#Hadoop Daemons

Daemons of Hadoop clusters

Background Processes or services that manage the distributed storage and processing in a cluster.



Master Node

1) Name Node

→ manages metadata, stores file to block mapping.

→ Doesnot store about data; only store location.

→ Critical daemon:
If namenode fails, HDFS becomes inaccessible

2) Secondary namenode

→ It is not backup node
: only a helper to namenode.

→ merges FS image with edit logs to keep name node efficient

→ Helps in faster recoveries after namenode restart

3) Resource Manager

-> Manages cluster resources for processing task.

-> Allocates Containers (CPU & Memory) for application.

Two Components

Scheduler
-> Allocates resources based on Policy

Application master
-> Manages Job execution.

Slave node Daemons

1) Data node

-> Store actual data blocks in HDFS.

-> Send heartbeat to ~~indicate~~ namenode to indicate that it is alive.

-> Report stored block information to namenode.

-> If datanode fails hadoop replicate it another node.

2) Node manager

→ Runs the task assigned by resource manager.

→ manages local memory, CPU & disk uses

→ Report resource availability to resource manager.

Replica Placement in HDFS

In HDFS, Replica Placement refers to how the file blocks are distributed and replicated across DataNodes in a cluster to ensure fault-tolerance, reliability, and high availability.

Two Policies

1) Rack-Aware Policy (Default Policy)

→ HDFS uses a rack-aware policy to place replicas across different nodes and racks to minimize data loss risk & improve network bandwidth usage.

→ Replication Factor: 3

→ Placement Strategy

i) First Replica
   → On the node where the client writes the data.

ii) Second Replica
   → On a node in a different rack

iii) Third Replica
   → On a different node in the same rack as second replica.

Rack-Aware Policy

This gives a balance between fault-tolerance & performance.

2) Risk-aware policy

Risk-aware policy is an improvement over the default policy to address more complex failure models and risk profiles (eg: Power supply, Network switches, correlated failures).

Features

→ Considers failure domains beyond just racks.

→ Models the risk of data loss using historical failure data & domain-specific knowledge

→ Places replicas to minimize correlated failures.

Benefits

→ Greater resilience

→ Improved data durability

→ More flexible.

Hadoop Streaming

-> It is a tool that allows us to run map reduce Job using any Programming language (not only JAVA)

-> It is useful for processing large scale data in a distributed manner.

(write about Map Reduce)

Chapter - 1

classmate
Date
Page

CAP Theorem

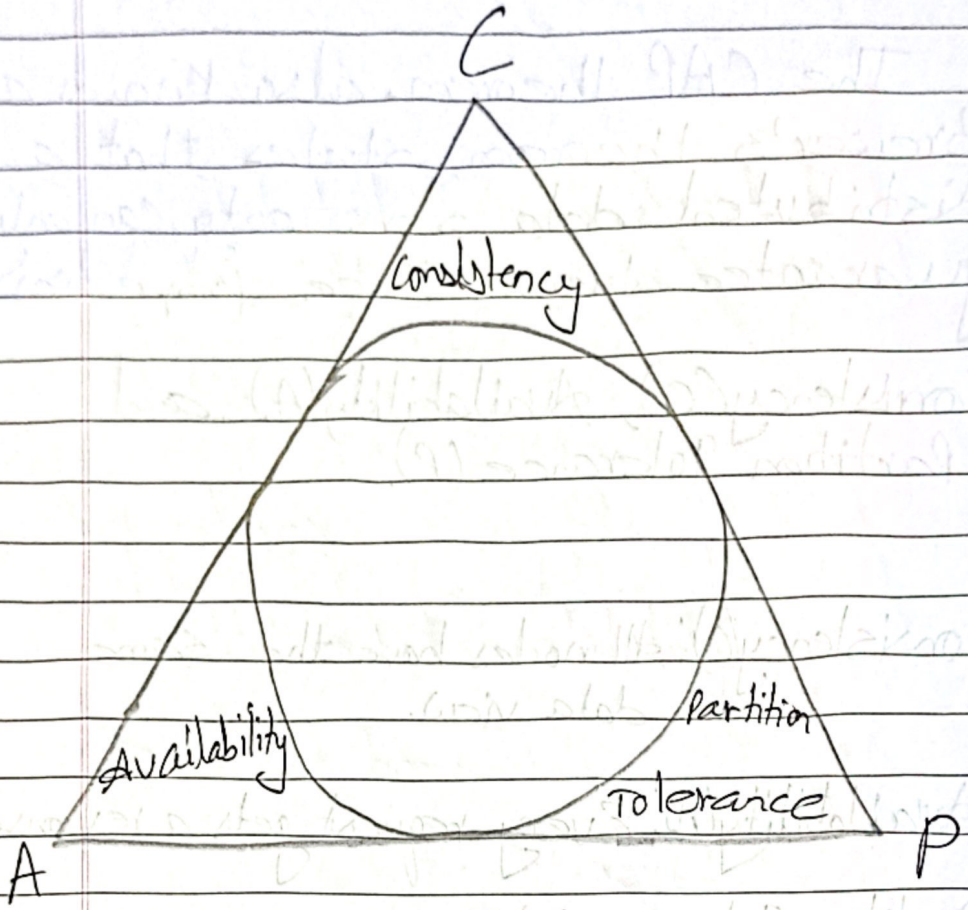
The CAP theorem, also known as Brewer's theorem, states that a distributed data store only can only guarantee two of three properties

Consistency (C), Availability (A) and Partition Tolerance (P).

Consistency (C): All nodes have the same data view.

Availability (A): Every request gets a response

Partition Tolerance (P):
-> System works despite network



CA

→ System responds last updated data & Promises higher availability

eg: RDBMS

CP

→ System can be distributed and Promise to respond last updated data.

eg: HBase, MongoDB

AP

→ System can be distributed and ~~fro~~ promise high ~~av~~ availability

eg: Cassandra, CouchDB

Why can't we have all three?

Network Failure

⇒ Network failures are ~~inevitable~~ sure to happen in distributed system. When partition occurs, some nodes may not be able to communicate with the other.

Trade-off

→ System designers must choose which two properties to prioritize based on their need.

Apache HBase

Overview

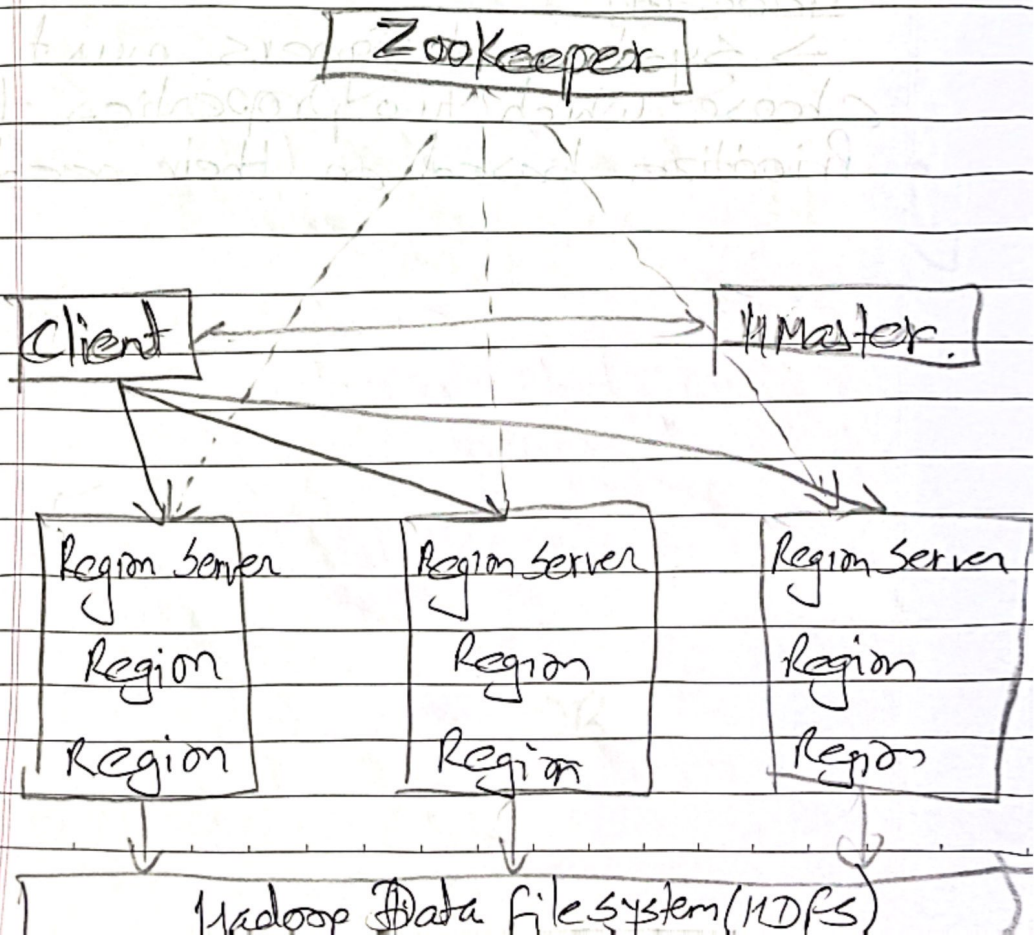
Distributed, column-oriented database built on Hadoop HDFS, inspired by Google Bigtable.

Features

-> Horizontal scalability

-> Real-time

(Architecture of H-base)



Components

HMaster
=> ~~Control~~
-> Manages metadata and region assignments

Region Server
=> Manages regions and handles client requests.

Regions
=> Partitioned parts of tables.

Zookeeper
=> Coordination and failure detection

HDFS
=> Underlying storage system.

Mongo DB

→ Distributed, document based database using BSON format

Key feature

1) Horizontally scalable

2) Schema flexible

Key element

1) Collection
→ group of document

2) Document
→ filed value pair which may vary in structure.

3) Object ID
→ Autogenerated unique identifie for each document

4) ~~Aggregation~~

Aggregation Pipeline

→ Aggregation Pipeline is a powerful framework in mongoDB for performing advance data processing and analysis on documents within the collection.

→ It allows
* transform documents
* Perform calculations
* Group data
* Reshape the output

Apache HIVE

→ Apache hive is a distributed data warehouse infrastructure built on top of Hadoop

→ It Provides:

* SQL-like querying using HiveQL
* Data summarization
* Analysis of large-scale datasets.

Use Cases

* Data warehousing and batch analytics
* ETL (Extract, Transform, Load) processes
* Ad-hoc querying and reporting
* Historical data analysis

Limitations

→ Not suitable for OLTP

→ High latency compared to RDBMS

→ Limited update/delete capabilities

→ Loose schema enforcement

HiveQL vs. Traditional SQL

Feature	HiveQL	Traditional SQL
Schema Model	Schema on read	Schema-on-write
Transaction support	Limited (ACID in newer versions)	Full support
Custom scripts	Yes (Python, Perl, etc)	No (requires UDFs)
~~Feature~~	~~HiveQL~~	
optimization	Map Reduce-based	Query plan Planner-based

Apache Pig

Difference betⁿ Pig & Hive

Execution Modes

→ Apache Pig supports multiple execution modes that determine how and where a Pig script will run.

→ The choice of execution mode typically depends on the environment and the nature of data processing task.

1) Local Mode
2) Tez Local Mode
3) Spark Local Mode
4) MapReduce Mode
5) Spark Mode
7) Tez Mode

1) Local Mode
→ In this mode, Pig runs on a single machine and access the local file system.

Pig -x local

2) Tez local Mode

→ This is an experimental mode that allows Pig to run on a local machine using the Tez engine.

→ Tez is a more efficient execution framework compared to mapreduce.

Pig -x tez_local

3) Spark local mode

→ Another experimental mode, it allows execution of Pig scripts using the Spark engine locally.

Pig -x spark_local

4) Map Reduce Mode

→ This is default execution mode. Here, Pig scripts are converted into MapReduce Jobs and run on a Hadoop cluster

Pig

5) Tez mode

→ If Tez is configured on the Hadoop cluster, Pig can use it as the execution engine.

Pig -x tez

6) Spark mode

→ Similar to Tez, Spark can also be used as an execution engine for Pig scripts. However, it requires a Spark cluster setup.

Pig -x spark.

Pig vs Hive

Aspect	Pig	Hive
Language	Pig Latin	HiveQL
Model	Procedural	declerative
Target Users	Developers	Analysts
Best use	ETL workflows	Data warehousing
Extensibility	Flexible UDF's (Java, Python)	UDFs supported
Schema Enforcement	optional	Required
Execution Engine	Mapreduce Tez, Spark	Tez, Spark