⇒ Unit-2

⇒ The Big Data Technology Landscape:-

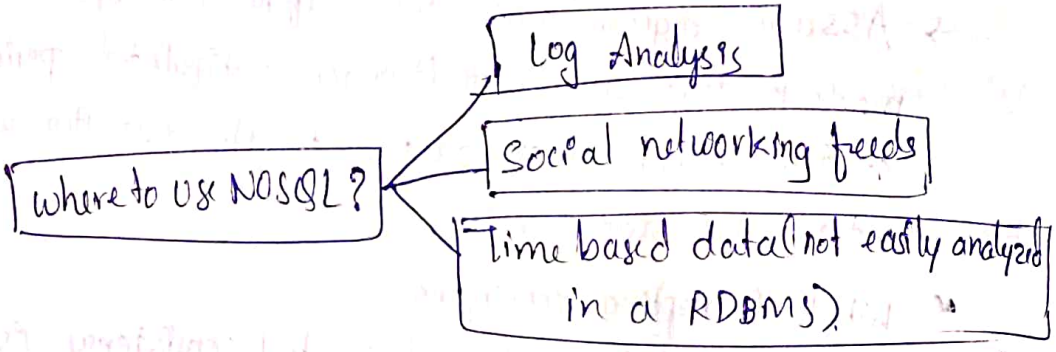
→ NoSQL (Not Only SQL)

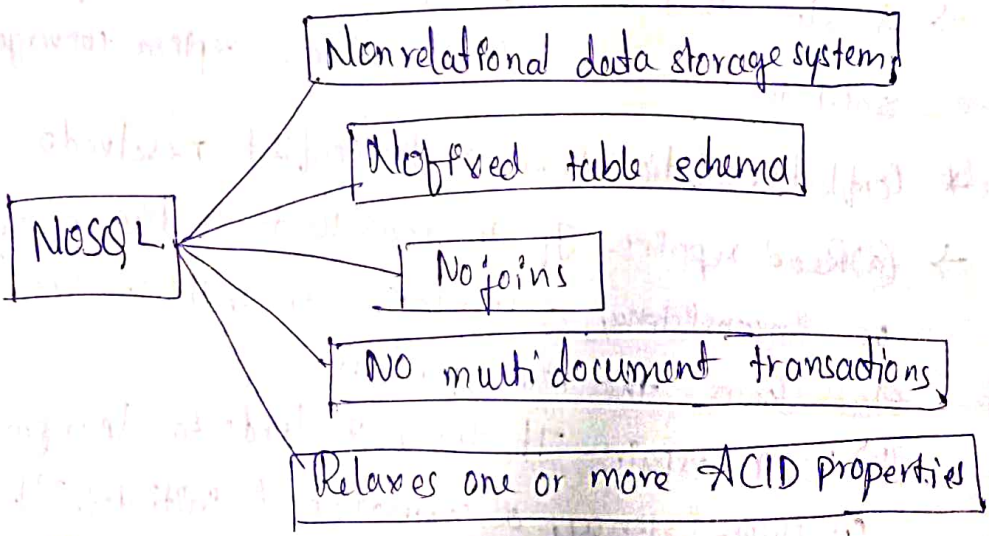
→ features of NoSQL:-

* Opensource
* non-relational (graph-based, column/document-oriented)
* distributed
* schema-less
* Cluster friendly
* Do not support ACID properties

→ * No SQL are widely used in bigdata & other real-time web applications

→ * These are non-relational, open source, distributed databases.
* They deal with rich variety of data: Structured, semi-structured & unstructured.





→ Types of NoSQL Databases:-

① Key-value or big hash table
② Schema-less.

① Key-value:- It maintains a big hash table of keys & values. Ex:- Dynamo, Redis, etc.

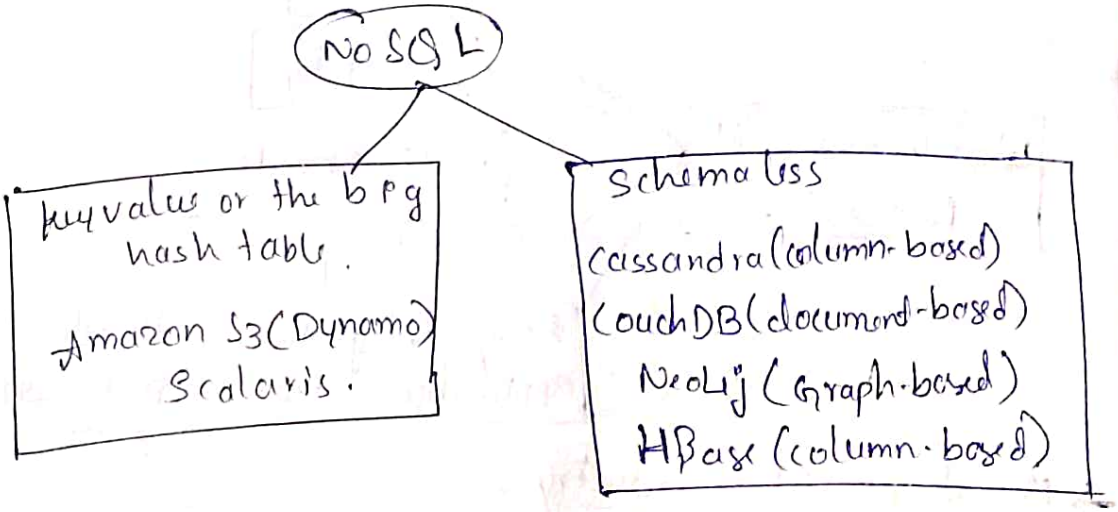
Sample key-value pair in key-value Database:

key	value
Firstname	Simmonds
Last name	David

② Document:- data in collections constituted of documents. Ex:- MongoDB, Apache CouchDB, etc.

③ column:- Each storage block has data from only one column. Ex:- cassandra, HBase etc.

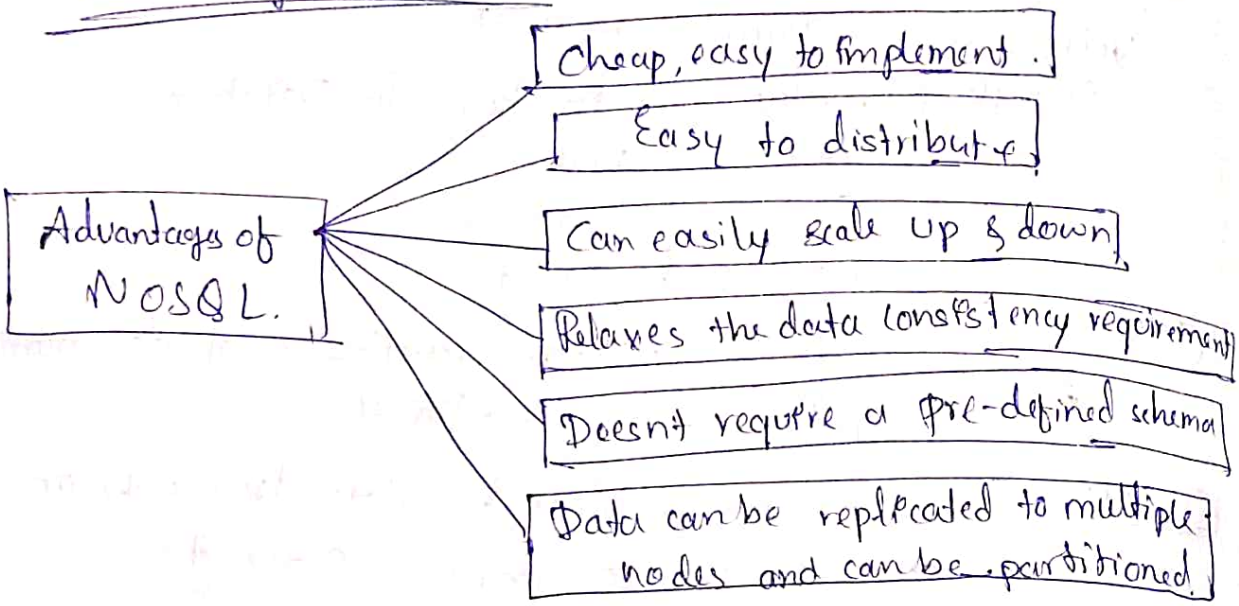
④ Graph:- (network db), stores data in nodes. Ex:- Neo4j, HyperGraphDB, etc..



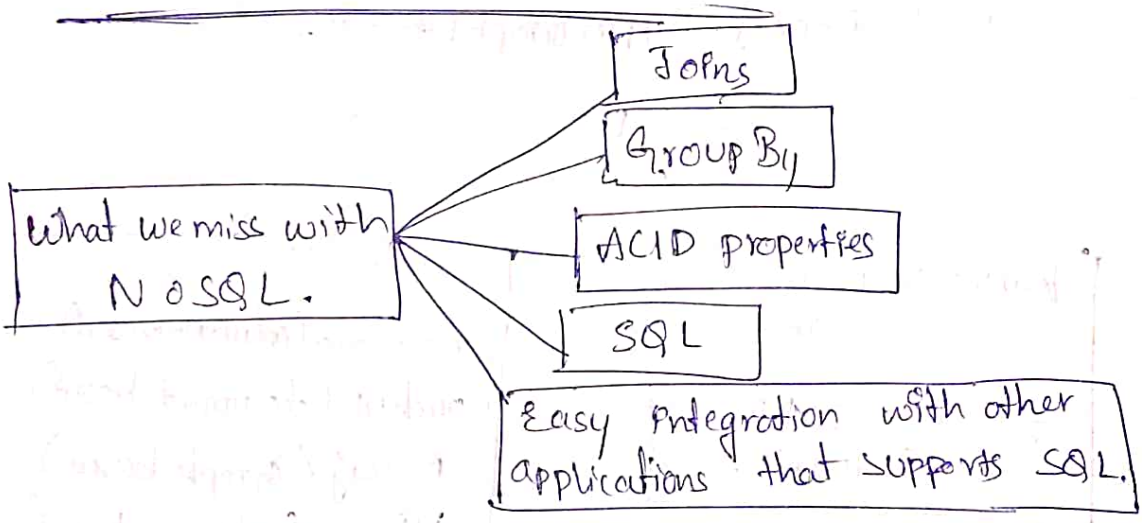
→ Why NoSQL:-

* It has scale out architecture instead of the monolithic architecture of relational databases.
* It can house large volume of structured, semi-structured, & unstructured data.
* Dynamic schema: NoSQL db allows insertion of data without a predefined schema.
* Auto-sharding: automatically spreads data across an arbitrary number of servers.
* Replication: It offers good support for replication which in turn guarantees high availability, fault tolerance, & disaster recovery.

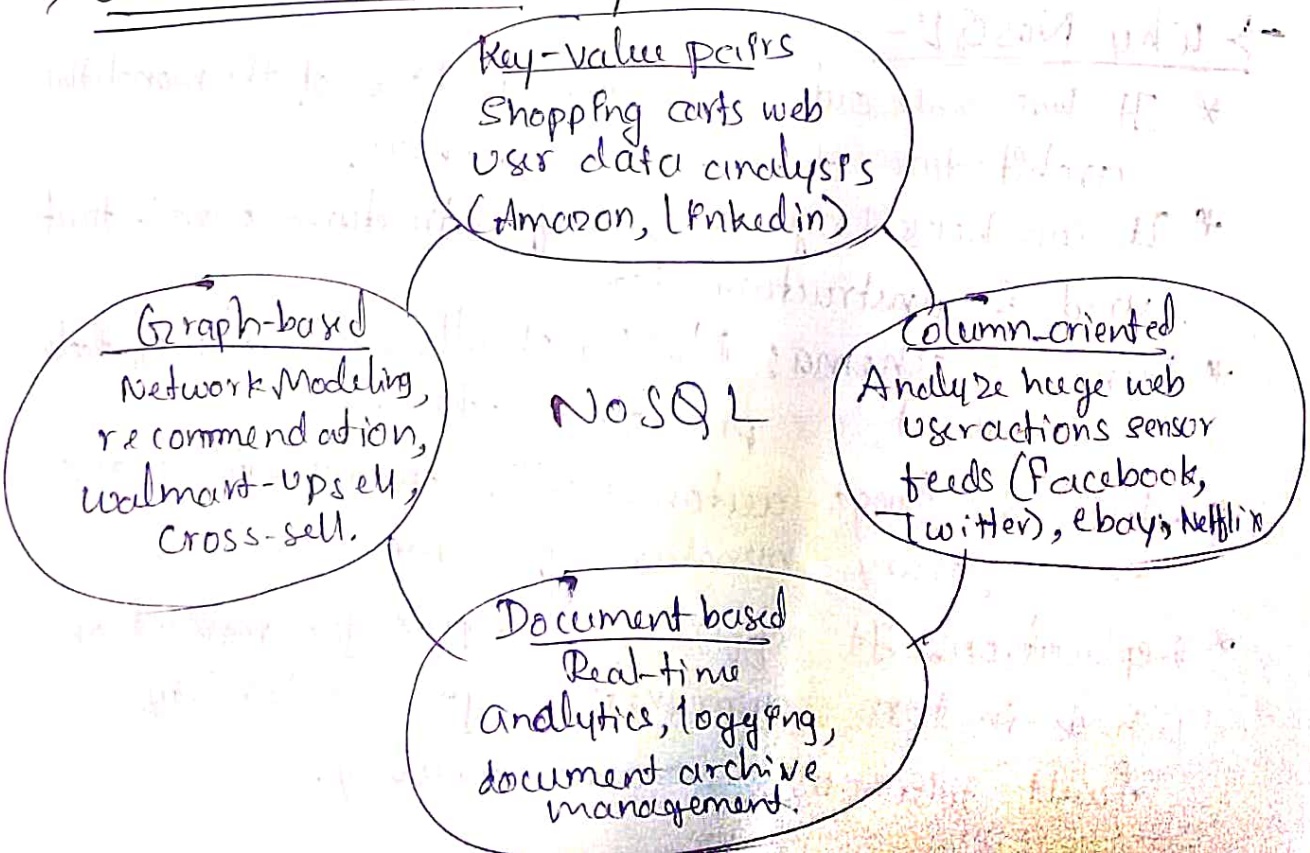
→ Advantages of NoSQL:-



→ What we miss with NoSQL:-



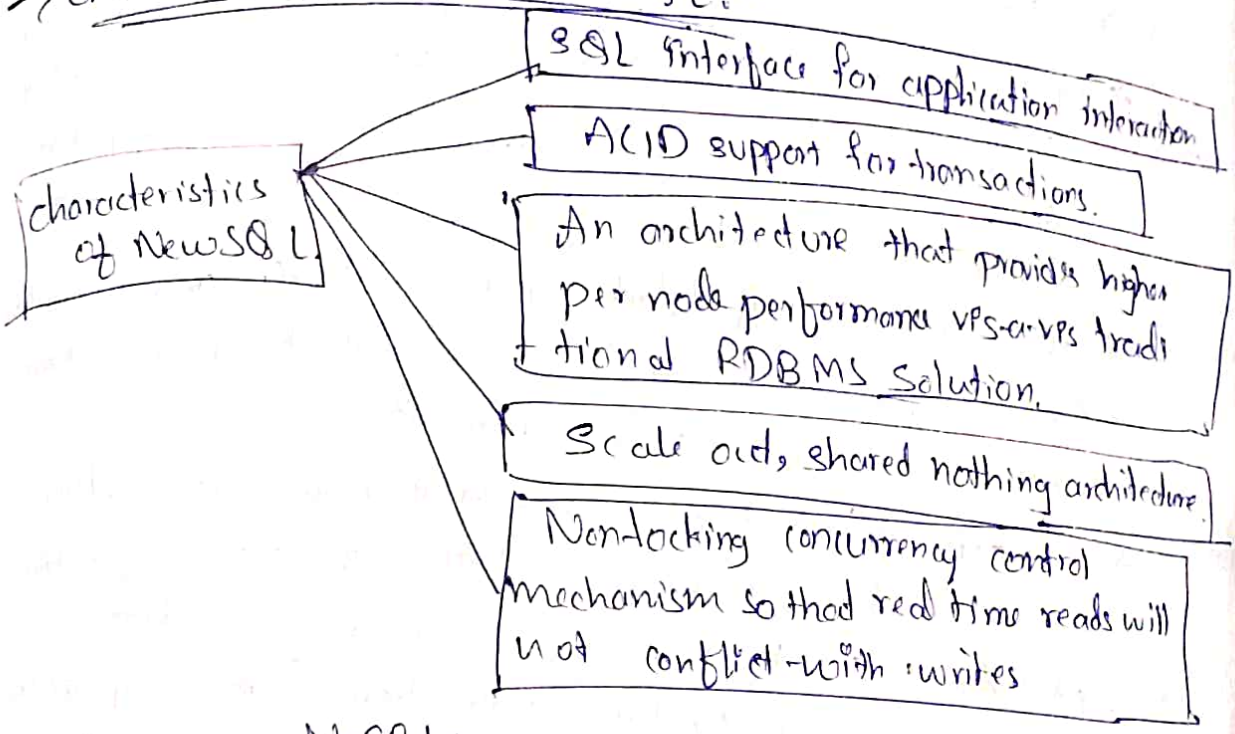
→ Use of NoSQL in Industry:-



→ NoSQL vendors:-

company	Product	Most widely Used by
Amazon	DynamoDB	LinkedIn, Mozilla
Facebook	Cassandra	Netflix, Twitter, eBay
Google	Big Table.	Adobe Photoshop

→ Characteristics of NewSQL:-



→ SQL versus NoSQL:-

SQL	NoSQL
* Relational db	* Non-relational, distributed db.
* Relational model.	* Model-less approach
* Pre-defined schema	* Dynamic schema for unstructured data.
* Table based db.	* Document-based / graph-based / wide column / key value pair db
* Vertically scalable (by increasing system resources).	* Horizontally scalable (by creating a cluster of commodity machines)
* Uses SQL	* Uses UnQL (unstructured QL)
* Not preferred for large datasets	* Largely preferred for large datasets
* Not best for hierarchical data	* best fit for hierarchical data.
* Emphasis on ACID properties	* Follows Brewer's CAP theorem
* Oracle, DB2, MySQL, MSSQL, PostgreSQL, etc..	* MongoDB, HBase, Cassandra, Redis, Neo4j, CouchDB, etc..

=> Hadoop:

* Hadoop is an opersource project of the Apache foundation. → framework written in Java.
* inspired by Google MapReduce & File system. → Hadoop distributed File System. MapReduce.

→ Features:-

* optimized to handle massive quantities of structured, semi-structured & unstructured data, using commodity hardware, that is, relatively inexpensive computers.
* Hadoop has a shared nothing architecture.
* It replicates its data across multiple computes so that if one goes down, the data can still be processed from another machine that stores its replica.
* Hadoop is for high throughput rather than low latency. It is a bath operation handling massive quantities of data, ∴ the response time is not immediate.
* It complements On-Line Transaction Processing (OLTP) & on-line Analytical Processing (OLAP). How ever its not a replacement of RDBMS
* It's not good when work cannot be parallelized, or when there are dependencies within the data.
* It's not good for processing small files. It works best with huge data files & datasets.

→ Advantages of Hadoop:-

* stores data in its native format
* No loss of information as there is no translation/ transformation to any specific schema.
* Scalability - proven to scale by companies like Facebook & Yahoo.
* Delivers new insights
* Higher Availability - Fault tolerance through replication of data / failover across computer nodes.

* Reduced cost - lower cost / terabyte of storage & processing.
* H/w can be added or swapped in or out of a cluster.

→ Versions of Hadoop:-
1. Hadoop 1.0.
2. Hadoop 2.0.

→ Hadoop 1.0:-
2 parts
* Data storage framework:- general purpose file sys. called HDFS (Hadoop Distributed File System).
* Data Processing Framework:- simple functional Programming model initially popularized by Google as MapReduce.
It essentially uses 2 functions: MAP & and REDUCE to Process the data.

→ Hadoop 2.0:-
* Hadoop 2.0, HDFS continues to be the data storage framework.
* new & separate resource management framework called Yet Another Resource Negotiator (YARN) has been added.

→ Overview of Hadoop Ecosystem:-
→ components of Data Ingestion:-
* Sqoop
* Flume
→ components of Data Processing:-
* MapReduce
* Spark.
→ components for Data Analysis:-
* Pig
* Hive
* Impala.

→ HDFS:- distributed storage unit of Hadoop.
→ HBase:- Stores data in HDFS.

→ Difference b/w HBase & Hadoop/ HDFS:-

* HDFS	* HBase.
* Filesystem	* Hadoop database.
* WORM (write once & read multiple times).	* real time random read & write.
* based on Google File System	* based on Google Bigtable
* Supports only full table Scan or partition table Scan.	* supports random small range scan or table scan.
* Performance of Hive on HDFS is relatively very good.	* Performance of Hive on HBase becomes 4-5 times slower.
* access to data is via MapReduce job only in HDFS	* HBase access is via Java APIs, Rest, Avro, Thrift APIs.
* not support dynamic storage owing to its rigit structure.	* supports dynamic storage.
* High latency operations	* low latency operations.
* most suitable for batch analytics	* Is for real-time analytics.

→ Hadoop Ecosystem & components for Data Ingestion:-

① Sqoop :- Stands for SQL to Hadoop. Its functions are:-
* importing data from RDBMS (MySQL, Oracle, DB2) to Hadoop file system (HDFS, HBase, Hive).
* Exporting data from Hadoop File system (HDFS, HBase, Hive) to RDBMS (MySQL, Oracle, DB2).

② Flume:- Its an important log aggregator (aggregates logs from different machines & places them in HDFS) component in the Hadoop ecosystem.

→ Hadoop Ecosystem components for Data Processing:-

① Map Reduce:- It is a programming paradigm that allows distributed & parallel processing of huge datasets. It is based on Google MapReduce.

① Spark:- It is both programming & computing model.
It's a open source big data processing framework.

→ Libraries:-
* Spark SQL
* Spark Streaming
* MLlib
* GraphX

→ Hadoop Ecosystem Components for Data Analysis:-

① Pig:- high-level scripting language used with Hadoop.
It serves as an alternative to MapReduce. 2 parts-
* PigLatin:- SQL-like scripting language.
* Pig runtime:- runtime environment.

② Hive:- It's a data warehouse also project built on top of Hadoop. 3 main tasks performed by Hive are summerization, querying & analysis

→ Hadoop Distributions:-

Core aspects:-
* Hadoop common
* Hadoop Distributed File System (HDFS)
* Hadoop YARN (Yet Another Resource Negotiator)
* Hadoop MapReduce

→ Hadoop Versus SQL:-

Hadoop	SQL
* Scale out.	* Scale up
* Key-value pairs	* Relational table
* Fonctional Programming	* Declarative Queries
* offline batch Processing	* Online transaction processing.

→ Integrated Hadoop Systems:-
* EMC Greenplum
* Oracle Big Data Appliance
* Microsoft Big Data Solution.
* IBM Infosphere
* HP Bigdata solutions

-> Cloud Based Hadoop Solutions:-

* Amazon web servius
* Google Big Query.

=> Introduction to Hadoop:-

-> Introducing Hadoop:-

-> Data: The Treasure Trove.

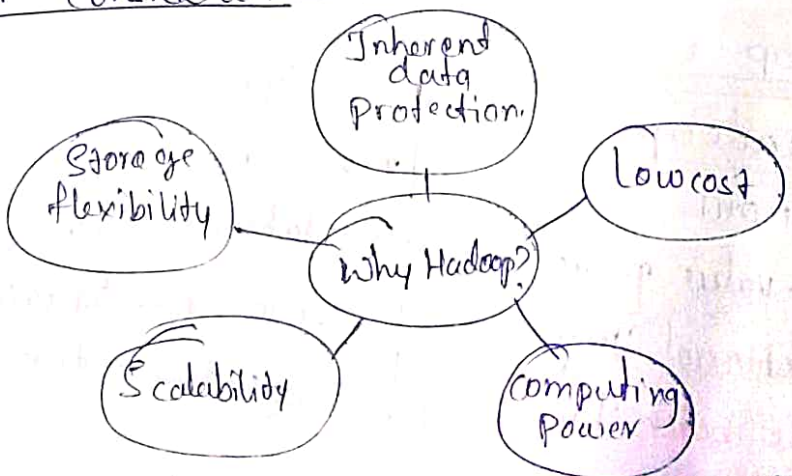
* provides business advantages such as generating product recommendations, inventing new products, analyzing the market, & many more..
* Provides few early key indicators that can turn the fortune of business
* provids room for precise analysis. If we have more data for analysis, then we have greater precision of analysis.

-> Why Hadoop?:-

-The key consideration is:-

Its capability to handle massive amounts of data, different categories of data fairly quickly.

Other considerations:-



-> data can be managed with Hadoop as follows:-

* Distributes the data & duplicates chunks of each data file across several nodes, ~~for example~~ 2.
* Locally available compute resource is used to process

each chunk of data in parallel.
* Hadoop framework handles failover smartly & automatically.

→ Why not RDBMS!-
* RDBMS is not suitable for storing & processing large files, images, & videos.
* It's not good in Advanced analytics involving ML.
* RDBMS costs increase with respect to increase in storage.
* It calls for huge investment as the volume of data shows an upward trend.

→ RDBMS versus HADOOP!-

RDBMS	Hadoop.
* Relational Database Management system.	* Node Based flat structure system
* Suitable for structured data	* Suitable for structured, unstructured data. Supports variety of data formats.
* OLTP - Processing	* Analytical, Big data Processing
* When data needs consistent relationship.	* Big Data Processing, which does not require any consistent relationship b/w data.
* Needs expensive hardware or high-end processors to store huge volume of data.	* In a Hadoop Cluster, a node require only ~~consistent relationship~~ a processor, a network card, & few hard drives
* Cost around $10,000 to $14,000 per terabytes of storage	* Cost around $4,000 per terabytes of storage.

→ Distributed computing challenges:
* Hardware failure (Replication factor)
* How to process this gigantic store of Data. (MapReduce)

→ Hadoop Overview!-

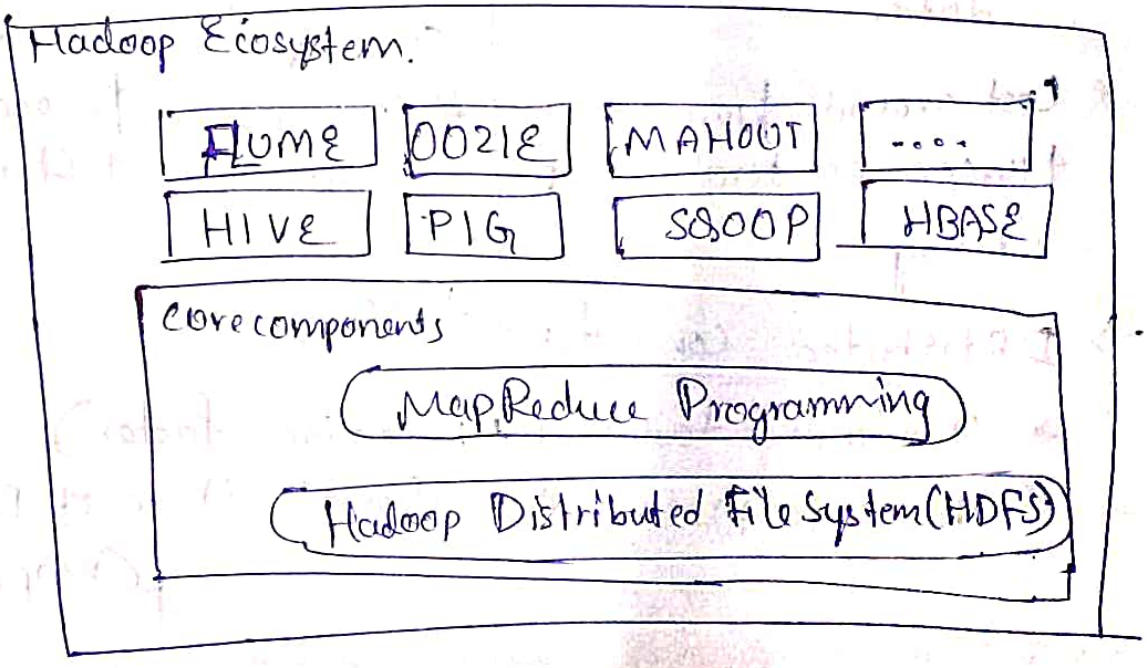
Open source software framework to store & process massive amounts of data in a distributed fashion on large clusters of commodity hardware. Basically, Hadoop accomplishes 2 tasks!-

* Massive data storage
* Faster data processing.

→ key Aspects of Hadoop!-

* Open source software :- It is free to download, use & contribute to.
* Framework :- Means everything that you will need to develop & execute & application is provided- programs, tools, etc.
* Distributed :- Divides & stores data across multiple computers, computation/processing is done in parallel across multiple connected nodes.
* Massive storage!- stored colossal amounts of data across nodes of low-cost commodity hardware.
* Faster processing!- large amounts of data is processed in parallel, yielding quick response.

→ Hadoop Components!-



→ Hadoop Core components:-

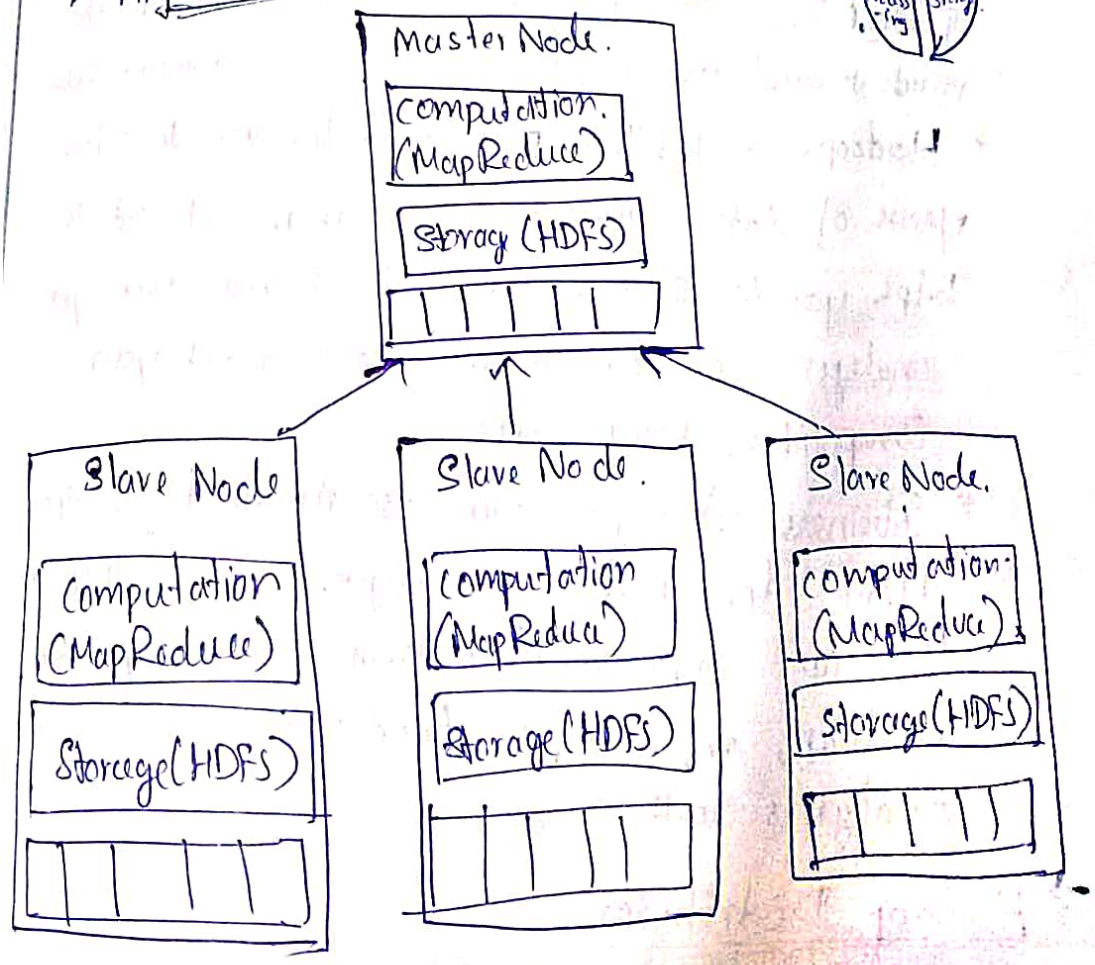
1. HDFS:-
   * Storage component
   * Distributes data across several nodes
   * Natively redundant.

2. MapReduce:-
   * Computational Framework.
   * Splits a task across multiple nodes
   * Processes data in parallel.

→ Hadoop Conceptual layer:-

* It is conceptually divided into Data Storage Layer which stores huge volumes of data and Data Processing Layer which process data in parallel to extract richer and meaningful insights from data.

→ High-Level Architecture of Hadoop:-



→ Use Case of Hadoop:-

→ Click stream Data:-

* helps to understand purchasing behavior of customers
* Clickstream analysis helps online marketers to optimize their product web page, promotional content, etc, to improve their business.

Clickstream Data Analysis Using Hadoop - Key Benefits		
Joins clickstream data with CRM & sales data.	stores years of data without much incremental cost.	Hive or Pig Script to analyze data.

Fig:- Clickstream Data Analysis

3 Key benifits:-

* Hadoop helps to join clickstream data with other Data sources such as Customer Relationship Management Data. This additional data often provides the much needed information to understand customer behavior.
* Hadoop's scalability property helps you to store years of data without ample increment cost. This helps you to perform temporal or year over year analyses on Clickstream data which your competitors may miss.
* Business Analysts can use Apache Pig & Apache Hive for website analysis. With these tools you can organize clickstream data by user selection, refine it, & feed it to visualization or analytics tools.

→ Hadoop Distributors:-

Cloudera	Hortonworks	MAPR	Apache Hadoop
* CDH 4.0	* HDP 1.0	* M3	* Hadoop 1.0
* CDH 5.0	* HDP 2.0	* M5	* Hadoop 2.0
		* M8	

HDFS (Hadoop Distributed File system):-

→ Key points:-

* storage component of Hadoop
* Distributed File System.
* Modeled after Google File system
* Optimized for high throughput
* You can replicate a file for a configured number of times, which is tolerant in terms of both sw & hw.
* Re-replicates data blocks automatically on nodes that have failed.
* You can realize the power of HDFS when you perform read or write on large files.
* Sits on top of native file s/m such as ext3 & ext4,

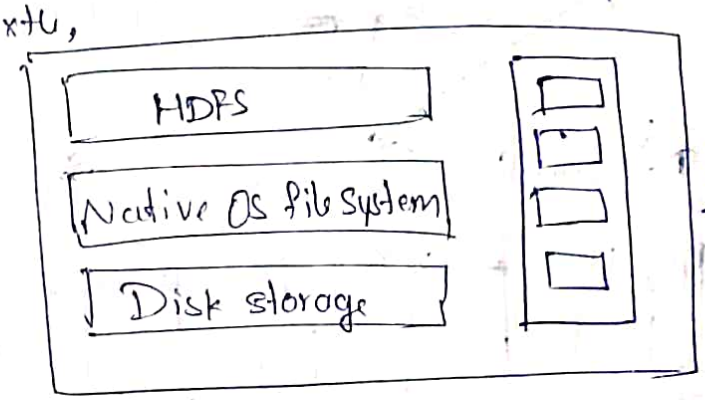


Fig:- HDFS

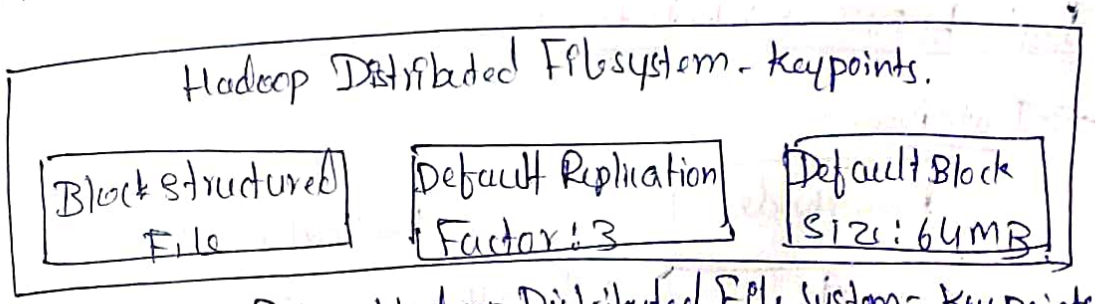


Fig:- Hadoop Distributed File System - Key points

→ HDFS Daemons:-

→ Namenode:-

* HDFS breaks a large file into smaller pieces called blocks.
* Namenode uses a rack ID to identify DataNodes in the rack.
* A rack is a collection of DataNodes within the cluster.

* Namenode keeps tracks of blocks of a file as it is placed on various Data Nodes.
* Namenode manages file related operations such as read, write, create & delete.
* Its main Job is managing the File System Namespace.
* A filesystem namespace is collection of files in the cluster.
* Namenode stores HDFS namespace.
* File system namespace includes mapping of blocks to file, file properties & is stored in a file called FsImage.

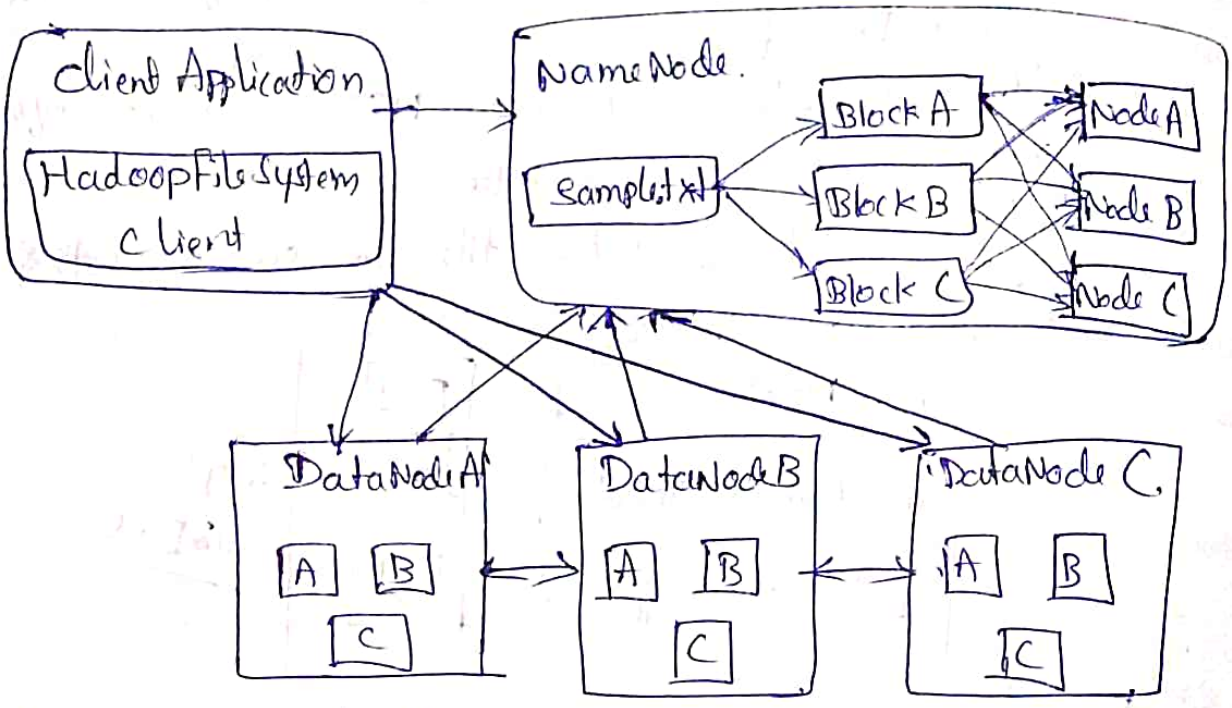


Fig:- HDFS Architecture

→ DataNode:-

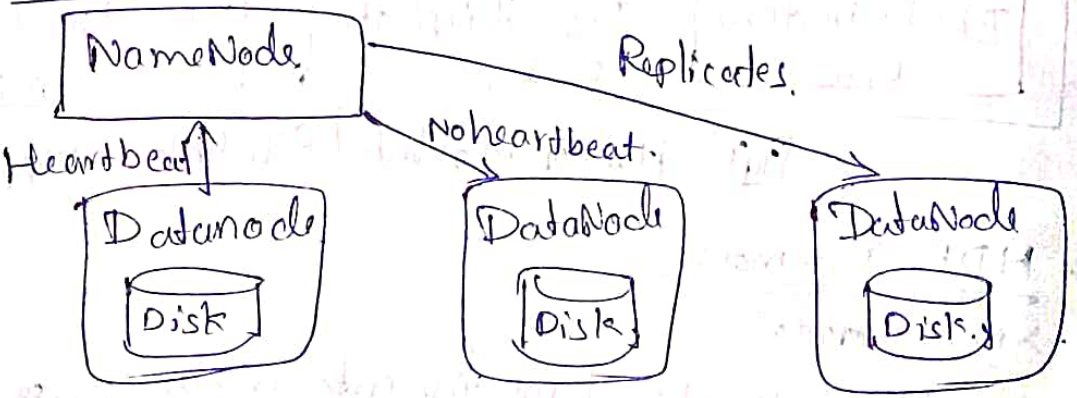


Fig:- NameNode & DataNode Communication.

* There are multiple Data Nodes per cluster.
* During pipeline read & write DataNodes communicate with each other.

* A DataNode also continuously sends "heart beat" message to NameNode to ensure the connectivity blw them.
* In case there is no heartbeat from Data node, the NameNode replicates that DataNode within the cluster & keeps on running as if nothing had happened.

→ Secondary NameNode :-

* Secondary Namenode takes a snapshot of HDFS metadata at intervals specified in the Hadoop configuration.
* Since memory requirement of Secondary NameNode are the same as NameNode, it is better to run NameNode & secondary NameNode on different machines.
* In case of NameNode failure, secondary NameNode can be configured manually to bring up the cluster.
* It doesn't record any realtime changes that happen to HDFS metadata.

→ Anatomy of File read :-

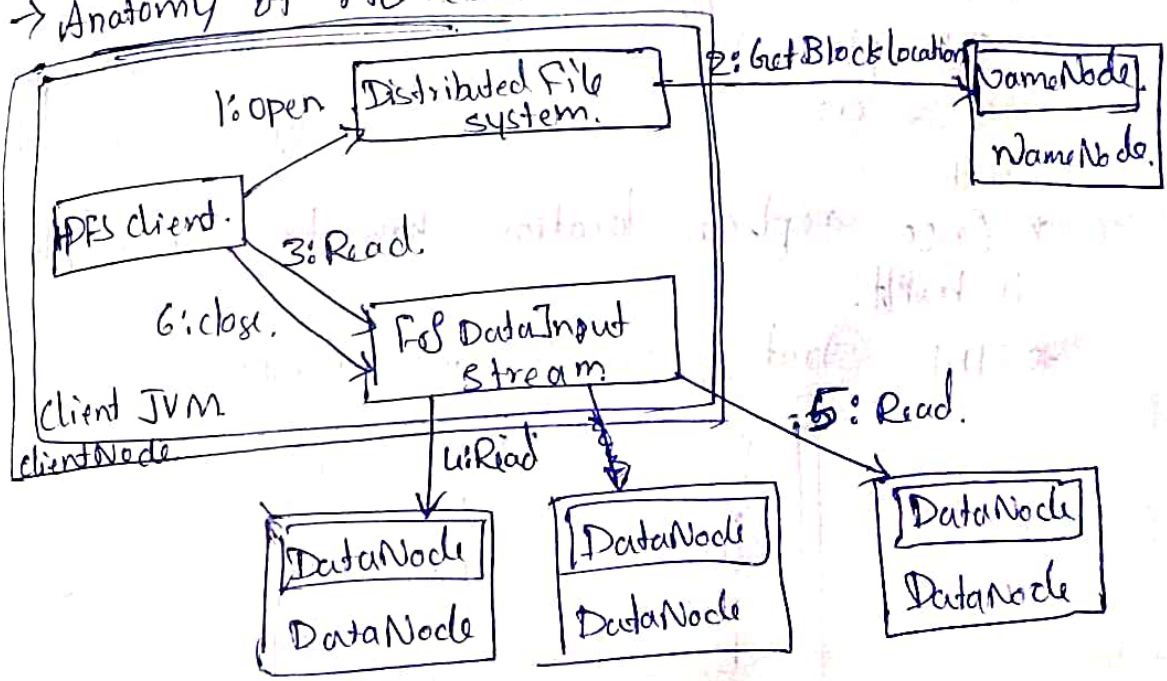
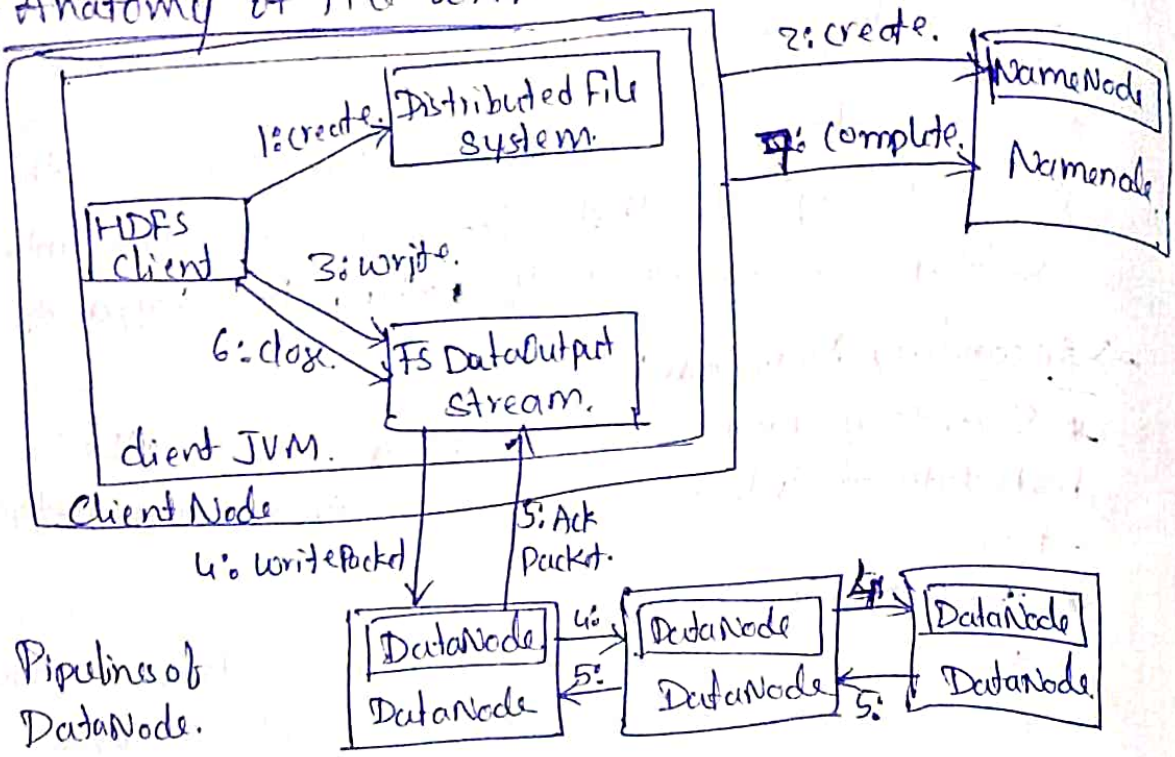


Fig :- File Read

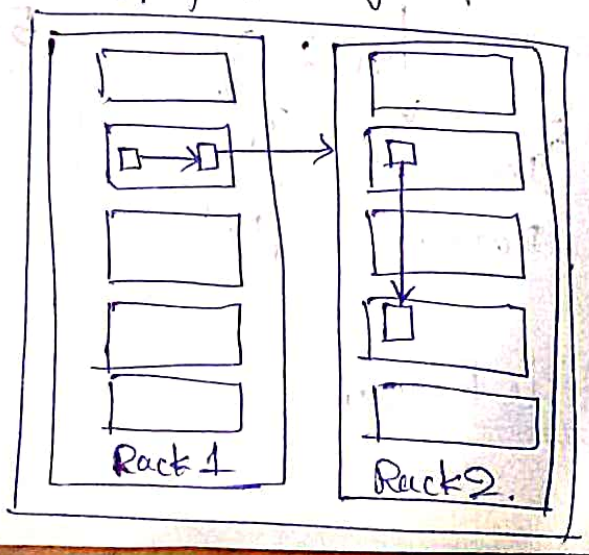
→ Anatomy of File write:-



→ Replica Placement Strategy:-

→ Hadoop Default Replica Placement Strategy:-

* As per this strategy, first replica is placed on the same node as the client.
* Then it places second replica on a node that is present on different rack.
* It places the third replica on the same rack as second, but on a different node in the rack.
* Once replica locations have been set, a pipeline is built.
* This strategy provides good reliability.



→ Special features of HDFS:-

(5) Data Replication:- There is absolutely no need for a client application to track all blocks. It directs the client to the nearest replica to ensure high performance

(6) Data Pipeline:- A client application write a block to the first DataNode in the pipline. Then this DataNode takes over & forward the data to the next node in the pipeline. This process continues for all the data blocks, & subsequently all the replicas are written to the disk.

→ Processing data with Hadoop:-

Map Reduce Framework

Phases	Daemons:
Map: converts I/p into key value pair Reduce: Combines o/p ~~to~~ of mappers & produces a reduced resultset.	JobTracker: Master, schedules task. TaskTracker: Slave, executes task.

* MapReduce Programming is a ~~s~~/w framework., & helps to process massive amounts of data in parallel.
* In MapReduce Programming, the i/p dataset is split into independent chunks.
* Map tasks process these independent chunks completely in a parallel manner.
* The o/p produced by the map tasks serves as intermediate data & is stored on the local disk of that server.
* The o/p of the mappers are automatically shuffled and sorted by the framework.
* MapReduce Framework ~~redu~~ sorts the o/p based on keys.
* This sorted o/p becomes the i/p to reduced tasks
* Reduced task provides reduced o/p by combining the o/p of various mappers.

* Job I/p & o/p are stored in a Files/m.
* MapReduce framework also takes care of the other tasks such as scheduling, monitoring, re-executing failed tasks, etc.

→ MapReduce Daemons:-

① JobTracker:- * Provides connectivity blw Hadoop & your application:
* when you submit code to cluster, Job Tracker creates the execution plan by deciding which task to assign to which node.
* It also monitors all the running tasks.
* when a task fails, it automatically re-schedules the task to a different node after a predefined number of retries.
* Jobtracker is a master daemon responsible for executing Overall MapReduce Job.
* There is a single Job Tracker per Hadoop cluster.

② TaskTracker:- * responsible for executing individual tasks that is assigned by the Jobtracker.
* There is a single TaskTracker per slave & spawns multiple Java Virtual Machines (JVMs) to handle multiple map or reduce tasks in parallel.
* TaskTracker continuously sends heartbeat massage to JobTracker.
* when Job Tracker fails to receive a heartbeat from a Task Tracker, the Job Tracker assumes that the TaskTracker has failed & resubmits the task to another available node in the cluster.
* Once the client submits a job to the JobTracker, it partitions & assigns diverse MapReduce tasks for each TaskTracker in the cluster.

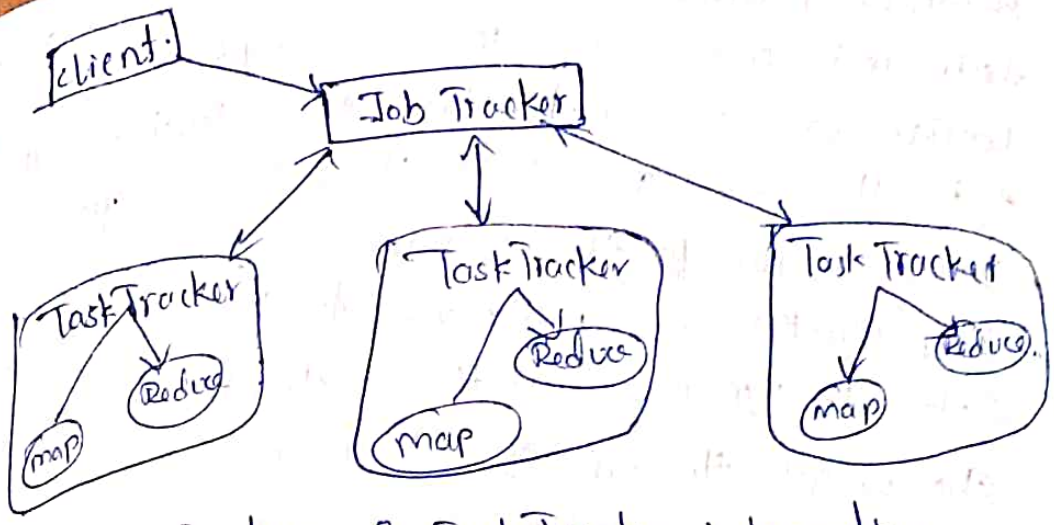


Fig:- JobTracker & Task Tracker interaction.

→ How does MapReduce work?

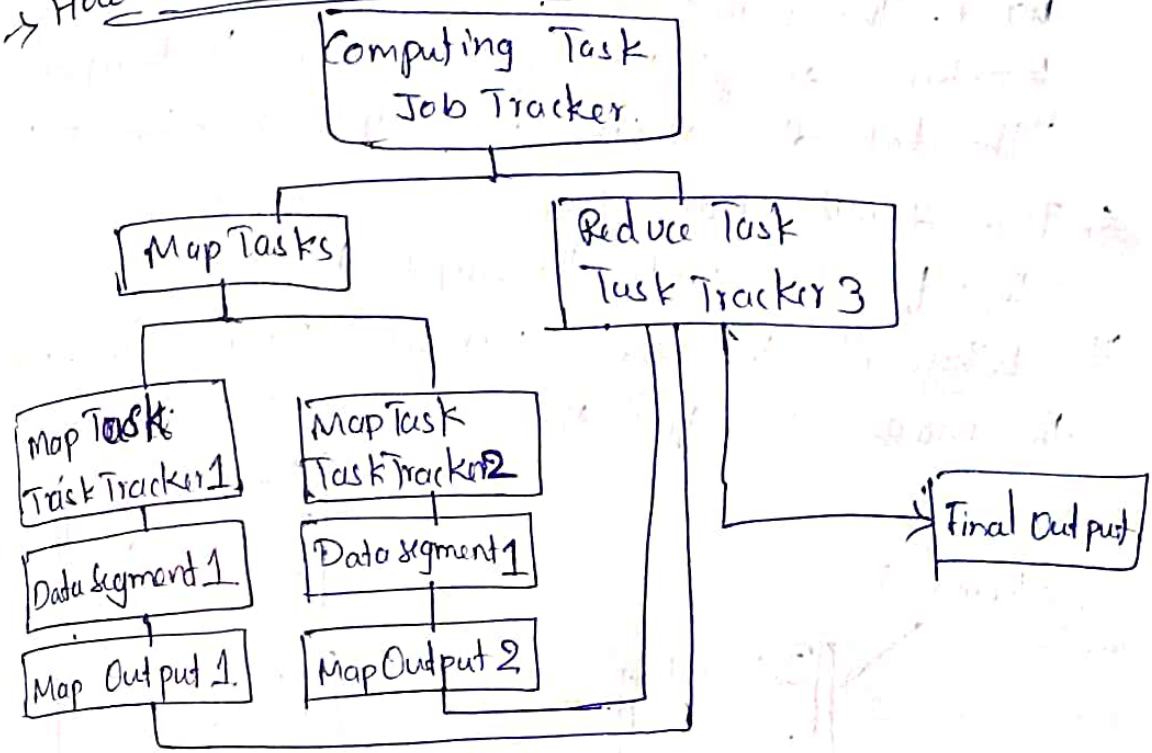


Fig:- MapReduce Programming workflow.

→ The following steps describes how MapReduce performs its tasks:

1. First, the i/p dataset is split into multiple pieces of data (several small subsets).
2. Next, the framework creates a master & several workers processes & executes the worker processes remotely.

3. Several Map tasks work simultaneously & read pieces of data that were assigned to each map task. The map worker uses the map function to extract only those data that are present on their server & generates key/value pair for the extracted data.

4. Map worker uses partitioner function to divide the data into regions. Partitioner decides which reducer should get the o/p of the specified mapper.

5. When the map workers complete their work, the master instructs the reduce workers to begin their work. The reduce workers in turn contact the map workers to get the key/value data for their partition. The data thus received is shuffled & sorted as per keys.

6. Then it calls reduce function for every unique key. This function writes the output to the file.

7. When all the reduce workers complete their work, the master transfers the control to the user program.

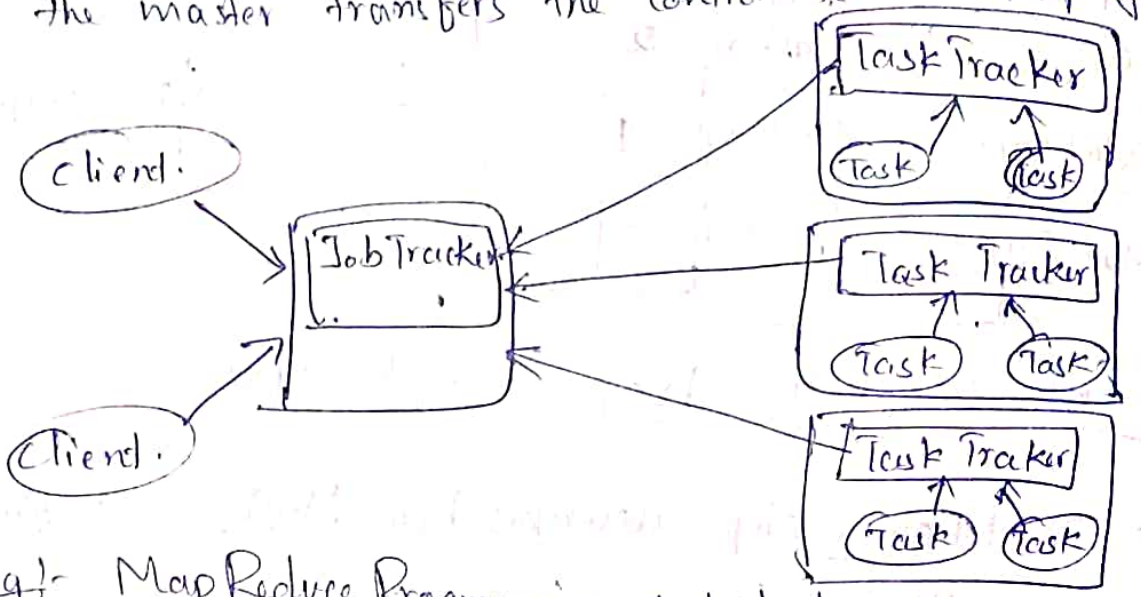


Fig:- MapReduce Programming Architecture.

→ Managing Resources & Applications with Hadoop YARN (Yet Another Resource Negotiator)

* Apache Hadoop YARN is a sub-project of Hadoop 2.x
* Hadoop 2.x is YARN based Architecture.
* It is a general processing platform.

* YARN is not constrained to MAPReduce only.
* You can run multiple applications in Hadoop 2.x in which all applications share a common resource management.
* Now Hadoop can be used for various types of processing such as Batch, Interactive, Online, Streaming, Graph & others.

→ Limitations of Hadoop 1.0 Architecture:-

* Single Namenode is responsible for managing entire namespace for Hadoop Cluster
* It has a restricted processing Model which is suitable for batch-oriented MapReduce jobs.
* Hadoop MapReduce is not suitable for interactive analysis.
* Hadoop 1.0 is not suitable for machine learning algorithms, graphs, & other memory intensive algorithms
* MapReduce is responsible for cluster resource management & data processing.

→ HDFS Limitations:-

* NameNode saves all its file metadata in main memory.
* There is a limit on the no of objects that one can have in memory on a single namenode.
* The NameNode can quickly become overwhelmed with load on the system increasing.

→ Hadoop 2: HDFS:-

HDFS 2 consists of 2 components:-

(a) namespace:- Service takes care of file-related operations, such as creating files, modifying files & directories
(b) blocks storage service:- handles data node cluster management, replication.

→ HDFS 2 Features:-

1. Horizontal scalability (independent NameNodes)
2. High availability

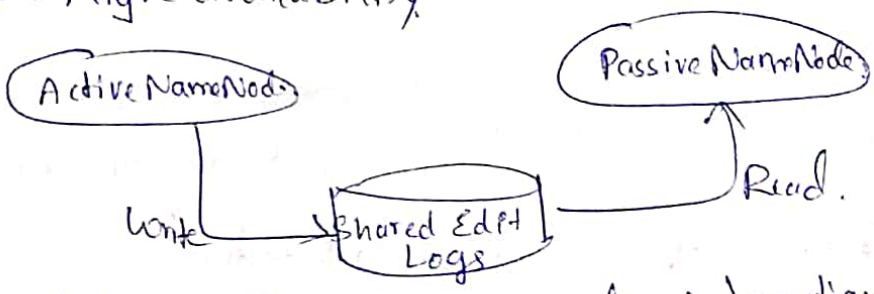


Fig:- Active & Passive NameNode interaction.

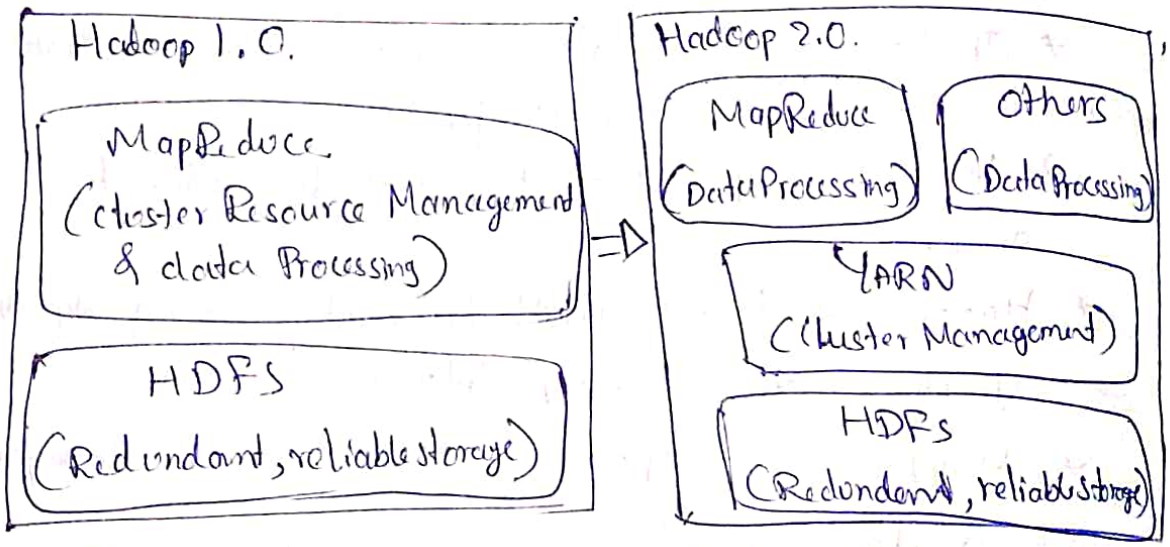


Fig:- Hadoop 1.X versus Hadoop 2.X

→ Hadoop 2 YARN: Taking Hadoop beyond Batch:-

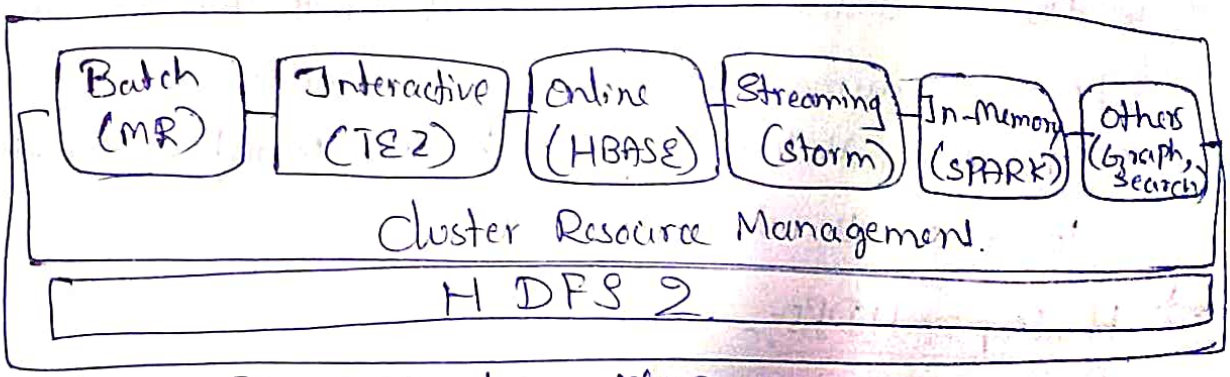


Fig:- Hadoop YARN

→ Fundamental Idea:-

* Spliting the JobTracker responsibility of resource management & Job scheduling/Monitoring into separate daemons.
* Daemons that are part of YARN Architecture are cluster

below:

(1) Global Resource Manager:- responsible to distribute resources among various applications in the s/m.

* 2 components:-
  (a) scheduler
  (b) Application Manager
    * Accepting job submissions
    * Negotiating resources for executing the application specific ApplicationMaster
    * Restarting the ApplicationMaster in case of failure.

(2) Node Manager:- responsible too launching application containers for application execution. Monitors resource usage → memory, CPU, disk, N/w etc., then reports the usage of resources to Global Resource Manager.

(3) Per-application ApplicationMaster:- responsible to negotiate required resources for execution from the Resource Manager. works along with Node Manager for executing & monitoring component tasks.

→ YARN architecture:-

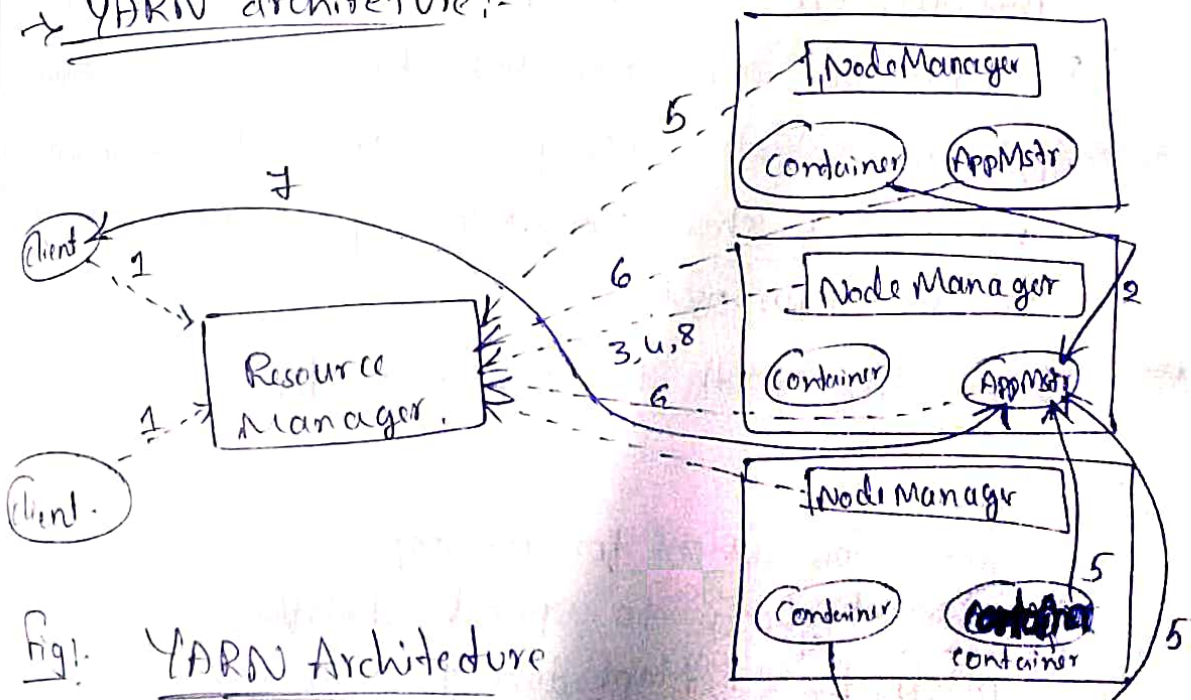


fig: YARN Architecture

1. A client program submits the application which includes the necessary specifications to launch the application-specific ApplicationMaster itself.

2. The ResourceManager launches the ApplicationMaster by assigning some container.

3. The ApplicationMaster, on boot-up, registers with the ResourceManager. This helps the client program to query the ResourceManager directly for the details.

4. During the normal course, ApplicationMaster negotiates appropriate resource containers via the resource-request protocol.

5. On successful container allocations, the ApplicationMaster launches the container by providing the container launch specification to the NodeManager.

6. The NodeManager executes the application code & provides necessary information such as progress, status, etc. to its ApplicationMaster via an application-specific protocol.

7. During the application execution, the client that submitted the job directly communicates with the applicationManager to get status, progress updates, etc. via an application-specific protocol.

8. Once the application has been processed completely, ApplicationMaster deregisters with the ResourceManager & shut down, allowing its own container to be repurposed.

⟹ Interacting with Hadoop Ecosystem :-

→ Pig :-

* data-flow system for Hadoop.
* uses PigLatin to specify dataflow
* alternative to MapReduce Programming.
* abstracts some details & allows you to focus on data processing.

① PigLatin : The data Processing Language

② Compiler : To translate PigLatin to MapReduce Program-ming.

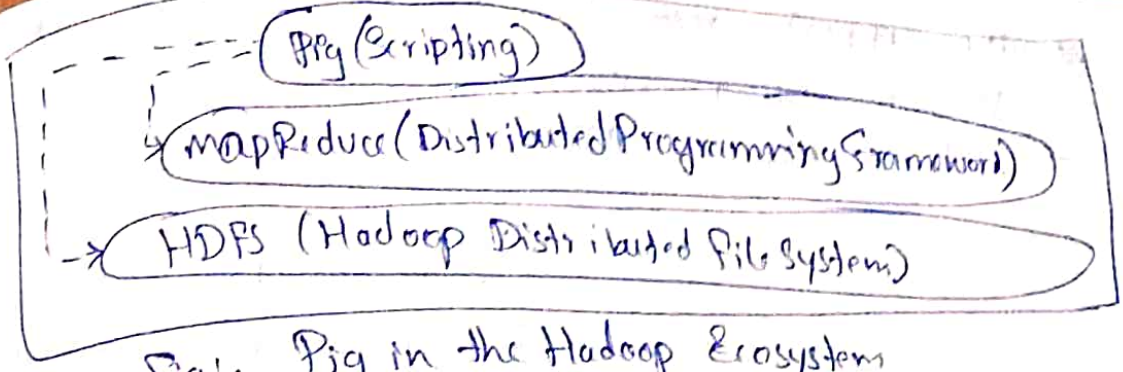


Fig:- Pig in the Hadoop Ecosystem

→ Hive:-

* Hive is a Data Warehousing layer on top of Hadoop.
* Analysis & queries can be done using an SQL-like language.
* can be used to do adhoc queries, summarization, & data analysis.

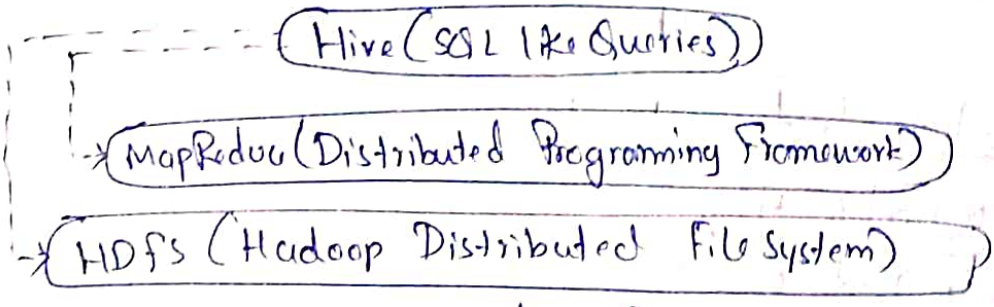


Fig:- Hive in the Hadoop Ecosystem.

→ Sqoop:-

* tool which helps to transfer data between Hadoop & Relational Database.
* you can import data from RDBMS to HDFS & vice-versa.

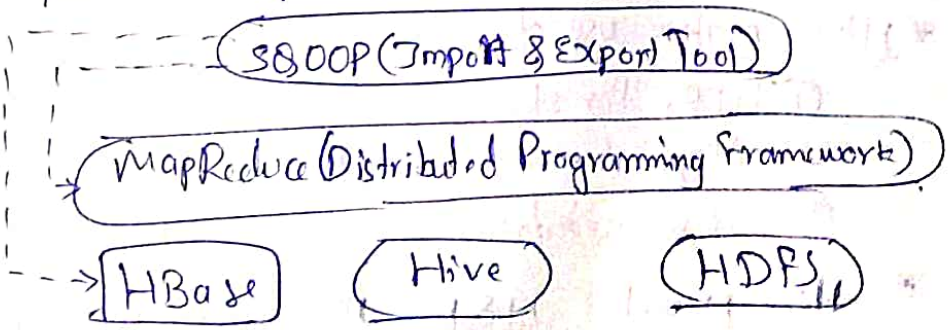


Fig:- Sqoop in the Hadoop Ecosystem.

→ HBase:-

* It's a NOSQL database for Hadoop.
* Column-oriented NOSQL db.
* Used to store billions of rows & millions of columns.
* Provides random read/write operation.

* also supports record level updates which is not possible using HDFS.
* sits on top of HDFS

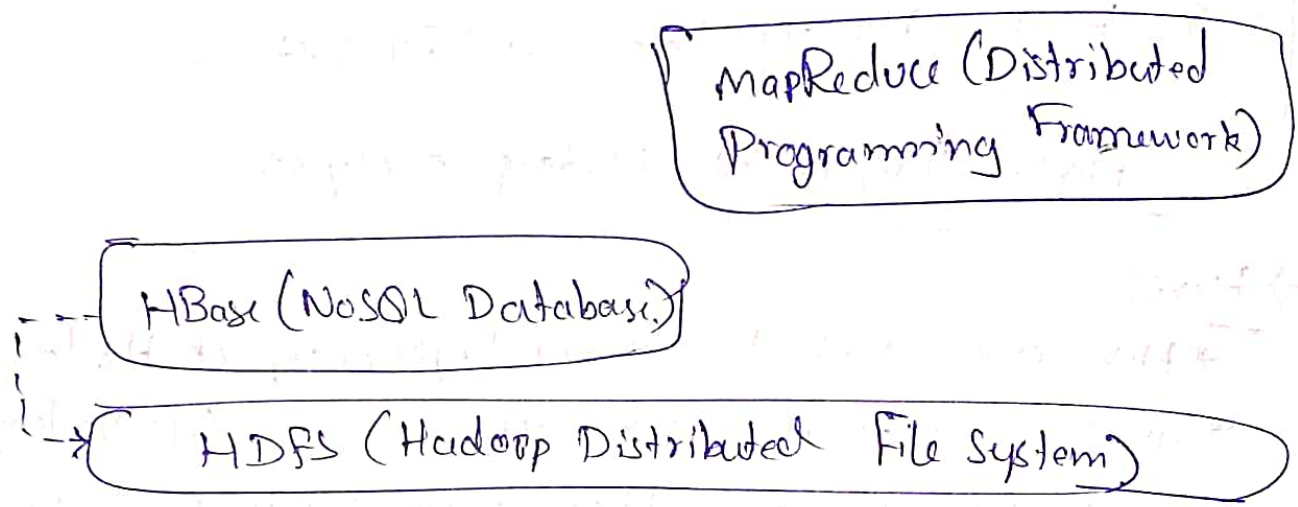


Fig:- HBase in the Hadoop Ecosystem