## ⭐⭐ (5 Vs of Big Data)

1. Volume: The size and amount of Big Data that companies manage, process and analyse.
2. Variety: The diversity & range of data types which includes structured, unstructured, semi-structured and raw data.
3. Velocity: The speed at which companies receive, store and manage data. Eg:- Specific number of social media posts or search queries received within a day, or other unit of time.
4. Veracity: Truth or accuracy of data and info assets.
5. Value: Value of Big data usually comes from insight discovery and pattern recognition that lead to more effective operations, stronger costumer relationships and other clear and quantifiable business benifits.

## (Advantages)

1. Enables smart thinking.
2. ~~Cut costs~~ Cost Reduction
3. Fraud Detection
4. Rise in productivity
5. ~~Advance costumer~~ Enhance Costumer Support
6. Enhance speed and agility

## (Disadvantage)

1. Talent and skill gap
2. Security Hazard
3. Adherence
4. High Cost
5. Data Quality
6. Rapid change

FREEMIND

Date

Page

# (Approaches to Manage Big Data)

1. Use a centralised interface / dashboard to monitor and ensure the availability of all big data resources.
2. Maintaining and updating database to get better outcomes.
3. Monitoring Big Data analytics, reporting and other similar solutions and implementing them.
4. Efficient design and implementation of Data Cycle process.
5. Control access and security of Big data repositories.
6. Using data visualization to reduce volume and improve BD operations.

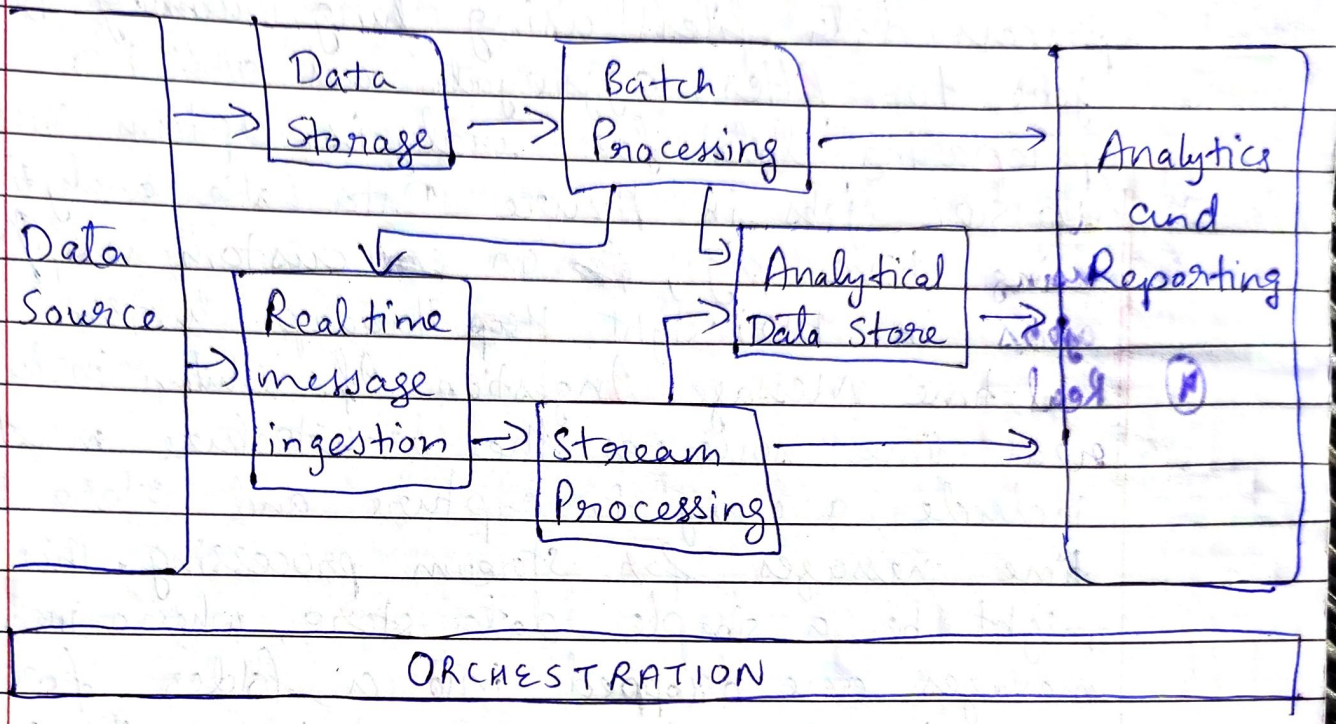
FREEMIND

Date

Page 93

# [Big Data]

## (Architecture)




A big data architecture is designed to handle ingestion, processing and analysis of data that is too large or complex for traditional database system.

Most Big Data architectures include these components:

① Data Sources: All big data solutions start with one or more data sources. Ex:- Relational Databases, Static files like web server log files, real time data sources like IOT devices.

② Data Store: For batch processing operations, data is typically stored in a distributed file store that can hold high volume of large files in various formats. It is often called a Data Lake. Ex:- Azure Data Lake, or Blob containers in Azure storage.

FREEMIND

Date

Page

③ Batch Processing: Often a big data solution must process data files using long running batch jobs to filter, aggregate and otherwise preparing data for analysis. Options include U-SQL jobs in Azure Data Lake analytics using hive pig, or custom mapreduce jobs in HDinsight Hadoop clusters.

④ Real-time Message Ingestion: If solution includes real time sources the architecture must include a way to capture and store real time messages for stream processing. This might be a simple data store, where incoming messages are dropped into a folder for processing. Ex:- Azure Event Hubs, Azure IOT Hubs and Kafka.

⑤ Stream Processing: After capturing real time messages and resources, the architecture must include a way to capture and store real time messages for stream processing. This might be a simple data store. The solution must process them by filtering, aggregating and otherwise preparing data for analysis. The processed stream data is then written to an output stream. Eg:- Azure Stream provides a stream processes based on SQL queries that operate on unbounded streams.

FREEMIND

Date

Page

# [BIG DATA]

(6) Analytical Data Store: It is used to serve these queries that can be a repository used to store BI and data analytics workloads. It's a centralised platform that manages large amount of data from various sources making it easily accessible for analysis, reporting and visualization.

(7) Analytics & Reporting: Analytics refers to the systematic computational analysis of data or statistics. It involves applying various technologies. such as statistical analysis, data mining, predictive modelling and machine learning to interpret and make sense of data.
Reporting is the process of organising data into summaries to monitor how different areas of businesses are performing. It typically involves the creation of static or dynamic reports, mostly in the form of tables, charts and dashboards.

(8) Orchestration: Refers to the automated management, coordination and arrangement of a complex data processing task accross different systems and environment. It involves the integration and scheduling of various data pipelines, ensuring that data flows smoothly from connection to processing, storage and analysis without manual intervention. Orchestration tools manage the dependencies between tasks, handle failures and optimise resource usage.

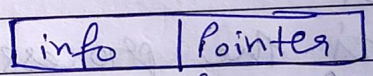
Date
Page 11

# [BD]

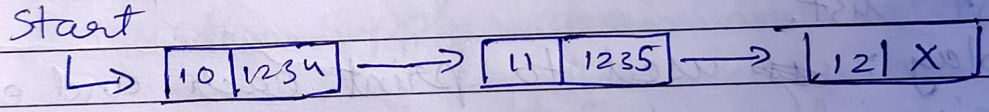
## Unit 2 : Data Structures

### 1. Linked List

- Linear data structure in which elements are not stored at contiguous memory location.
- It's also a dynamic data structure. The number of nodes in a list is not fixed, and can grow and shrink on demand.
- Each element is called a node which has 2 parts: ① info parts which store info ② Pointer which stores the address of the next element.



Node




Advantages---

1. Dynamic DS: can grow & shrink on demand during program execution.
2. Efficient Memory Utilization: Memory is not pre-allocated but assigned when required and can be removed when it's no longer needed.
3. Insertion and Deletion is easy and efficient: It provides flexibility in inserting a data item at a specified position and deletion of data item from a given position.
4. Many complex applications can be easily carried out with linked list

FREEMIND

Date

Page

## Operations...

Basic operations to be performed on the linked list are:

1. Creation: used to create a linked list & to link with another node.
2. Insertion: used to insert a new node in linked list: at the beginning, end or at a specified position.
3. Deletion: same as previous point but vice-versa.
4. Traversing: It is a process of going through all the nodes of a linked list from one end to another.
5. Concatenation: It is the process of joining the second linked list to the end of the first list.
6. Display: used to print each and every node's information.

## Types...

1. Singly
2. Doubly
3. Circular
4. Circular Doubly

FREEMIND

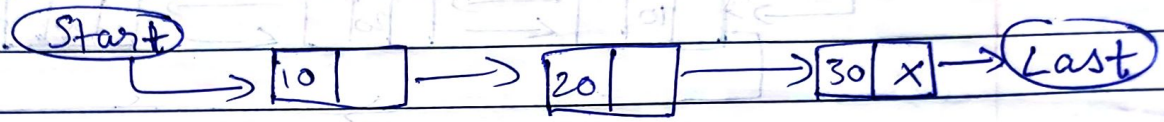
Date

Page 27

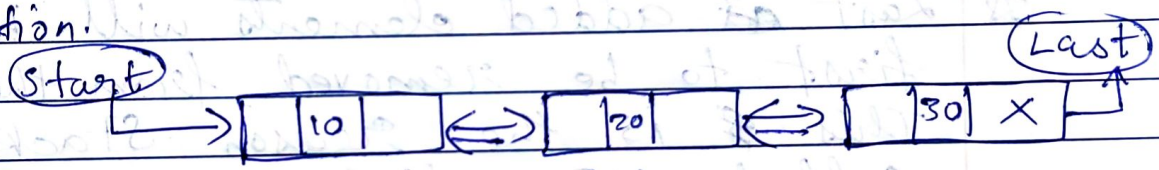
# [BD]

## Types of Linked List

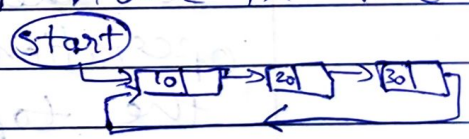
1. Singular :
In this, all nodes are linked in a singular sequential manner. It's also called Linear Linked List.




2. Doubly :
In this, all nodes are linked together by multiple links which help in accessing both the successor node and predecessor nodes within the list. This helps to traverse the list in the forward and backward direction.

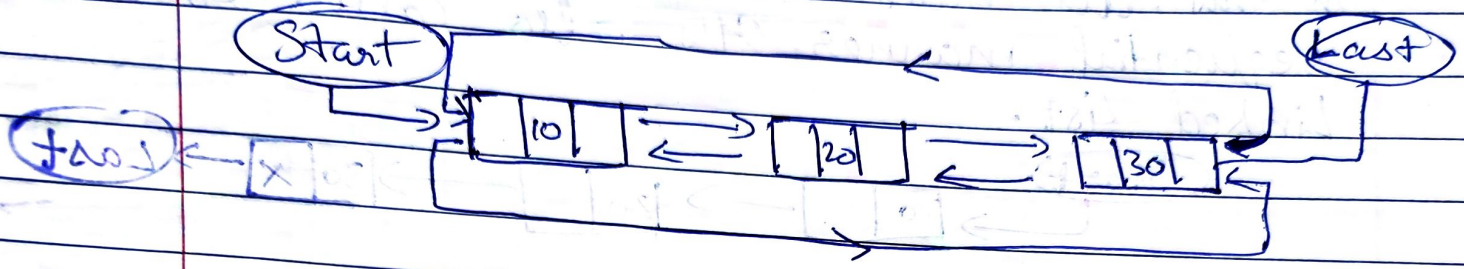



3. Circular :
In this, there is no beginning or end. A singly linked list can be converted to a circular linked list by simply sorting the address of the very first node in the link field of last node.




4. Circular Doubly :
Complex Data structure that combines the features of both doubly and circular linked list. It has 3 fields for each node : Data field, previous pointer and next pointer. It follows a circular

structure in which.




## 2. Stack

- It's a non-primitive data structure
- Ordered list in which addition of new data item and deletion of existing data is done from only one end, i.e. Top of the Stack.
- Last added elements will be the first to be removed from the stack. This is the reason Stack is called LIFO list.

### (Operations)

There are 2 types of operations:

1. Push: process of adding a new element to the top of stack is called Push operation. Every new element added to the top is incremented by 1. In case the array is full and no new element can be added, it is called Stack Overflow.
2. Pop: process of deleting an element from the top of stack is called pop operation. After every pop operation,

the top of stack is decremented by 1. If there is no element on the stack, and the pop is performed, then, this will result in stack Underflow.

# [QUEUE]

- Queue is a non-primitive linear data structure
- homogeneous collection of elements in which new elements are added at the rear end, and the existing elements are deleted from the front end.
- first added element will be first to be removed (FIFO principle)
- every insert operation at rear is incremented by 1. $r = r + 1$.
- every delete operation at front is incremented by 1. $f = f + 1$.

## [Circular Queue]

- In this, insertion of a new element is done at the very first location of queue if the last location of queue is full.

FREEMIND

Date

Page

* It overcomes the problem of unutilised space in linear queues implemented as arrays.

Conditions...

1. Front will always be pointing to 1st element.
2. If front = rear, the queue is empty.
3. Each time, a new element is inserted in queue, the rear point is incremented by 1.
4. Each time, an existing element is deleted from queue, the front point is incremented by 1.

## [SET]

* data structure that stores the collection of distinct elements.
* fundamental data structure used to store and manipulate a group of objects where each object is unique.
* signature property = it doesn't allow duplicate elements.

(Types)

① Unordered:

* An unordered associative container implemented using a hash table, where keys are hashed into indices of a hash table so that the insertion is always randomised.
* all operations take constant time, on an average, which can go upto linear time O(n) in the worst case.

(2) (Ordered)

* Common set data structure we are familiar with.
* generally implemented using balance binary search tree.
* Supports O(log(n)) lookups, insertion and deletion operations.

Date
Page

# [RD]

## [MAP]

- Map data structure, also called Dictionary Associative Array or Hash Map, is a data structure that stores a collection of key value pairs where each key is associated with a single value.

Eg:- < 1 a >
(Key, value)
< 2 b >
< 3 c >

- It provides an efficient way to store & retrieve data based on a unique identifier (key).

(Properties)

1. Key Value Association.
2. Dynamic Size: Maps can grow & shrink dynamically as you add or remove elements.
3. Efficient lookup: you can quickly find the value associated with a specific key using methods like get() or [], with an average time complexity of O(1). O(1) for hash based implementation and O(log(n)) for tree based implementations.
4. Duplicate Key Handling: exceptions like multimap in C++ allow duplicate key handling.

## [Wrapper Class]

- Class whose objects wrap/contain primitive data types. When we create an object to a wrapper class, it contains a field, and in this field, we can store primitive data types. In other words, we can wrap a



primitive value into a wrapper class object.

| (Primitive Data Type) | (Wrapper Class) |
|---|---|
| char | Character |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| boolean | Boolean |

Ex:- ① int data = 20;
Integer i = new Integer(data);
# Now i is 20

② # we can directly pass the value:
Integer j = new Integer(20);

FREEMIND

Date

Page 91

[BD]

# Unit 3: Hadoop

→ open source framework that allows us to store and process large datasets in a ~~dist~~ parallel and distributed manner.

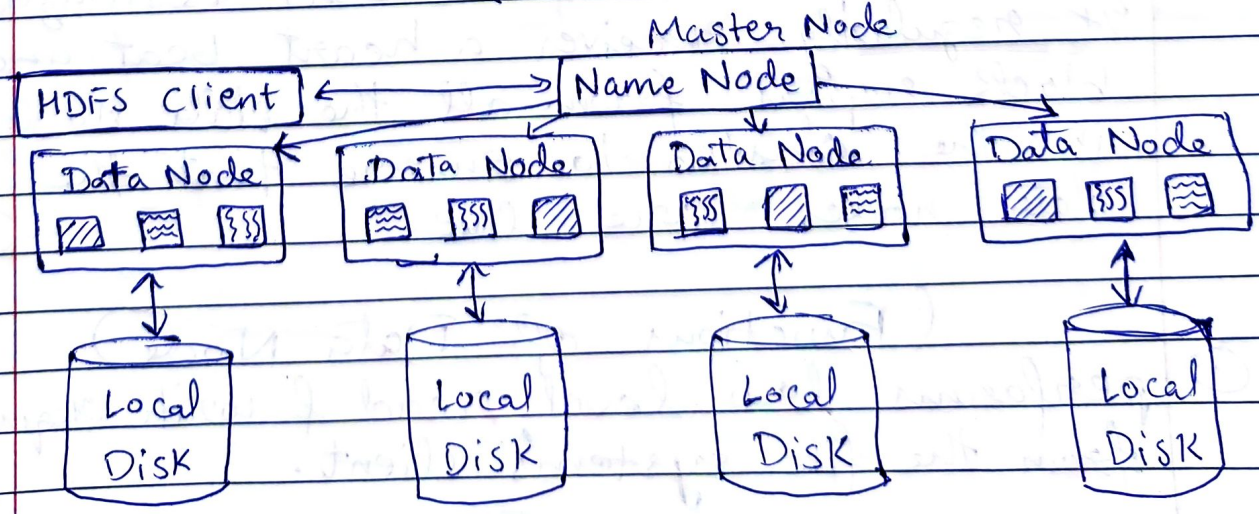
→ invented by Dough Cutting & Mike Cafarella.

⇒

(characteristics)

1. can scale from a single server to 1000s of machines, each offering local computation & storage.
2. data is automatically replicated ~~across~~ across different nodes, ensuring no data is lost even if one node fails.
3. Can process various types of data (structured, semi-structured, unstructured).
4. as an open source framework, it reduces the need for expensive ~~hardware~~. hardware.
5. has 2 main components:
   i) HDFS
   ii) Map Reduce

## {HDFS}



FREEMIND

Date

Page

→ Hadoop comes with a distributed file system called HDFS, in which, data is distributed over several machines & replicated to ensure their durability to failure & high availability to parallel application.

→ It is cost effective as it uses commodity hardware; it involves the concept of data nodes, blocks and name nodes.

(Functions of Name Node)

① records the meta data of all the files stored in the cluster. Eg:- location of blocks stored, size of files, permissions, hierarchy, etc. are all included in this.

② 2 types of files associated with Meta Data / Master Node:-

i) FsImage: contains the complete state of the file system name space since the start of the name node.

ii) Edit Log: contains all the recent modifications made to the file system with respect to the recent FsImage. It regularly receives a heart beat and block report from all the data nodes in the cluster to ensure that the data nodes are live.

(Functions of Data Node)

① performs low level read & write request from the file system's client.

BDC(Big Data).

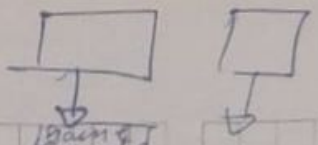
Building Blocks of HDFS.

1.) Namenode.
2.) Data node.
3.) Secondary namenode.
4.) job Tracker.
5.) Job tasker.

Junction of Name node: ☆ track server host not store user data Per Com

- Server hosting the namenode ~~dat~~ doesn't store user data and perform any computations.
- Secondary name node → is that assitant which monitors the state of cluster.





DT.

- each cluster has an secondary namenode. its communicate with the namenode to take snapshot of HDFS meta data at regular interval of time. which is define by cluster configuration
- snapshot help in minimizing the down time & loose of data

## * Job Tracker :-

the role of job tracker is true link between your app and hadoop.

- it determines the following execution plan :-
  a) file to the processed
  b) assigning of nodes to the different task
  c) Monitoring all the task when they are running.
- if task fails job tracker will automatically relaunch renewed task

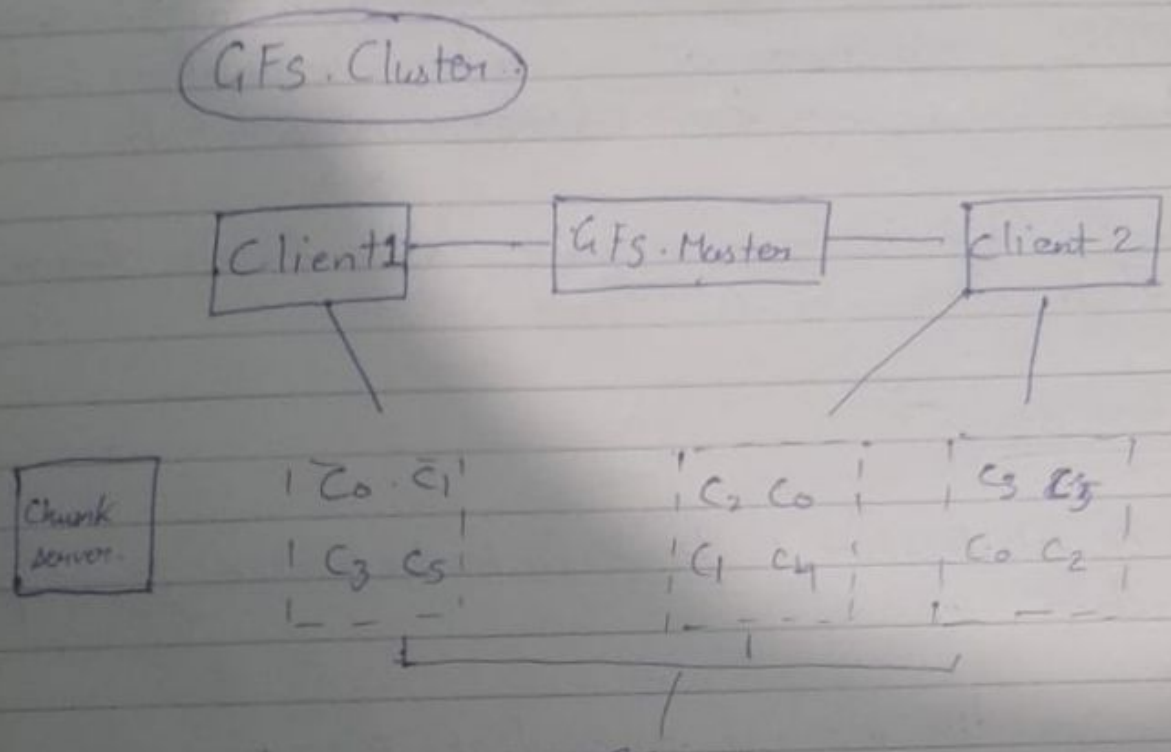
- task tracker works constantly with to communicate with job tracker.

- if a task trackers fails receive the heartbeat from job tracker within specified amount of time.

- it will assume that the task tracker has crashed and will resubmitted the crossponding task to other node in the cluster.

DT.

# Google file system.




GFS Master :-

Client, server and Masters are Linux machines and each runs a server process at the user level which is known as 'user level server processes'

- meta data is managed by GFS master and communicate with clients and Chunk server
- Chunk server : can be define as small block of data from system files

DT. □□ □□ □□□□

Client interacts with chunk server for data transfer.

GFs stores 3 replicas by default. which can be obtained at any levels defined by users

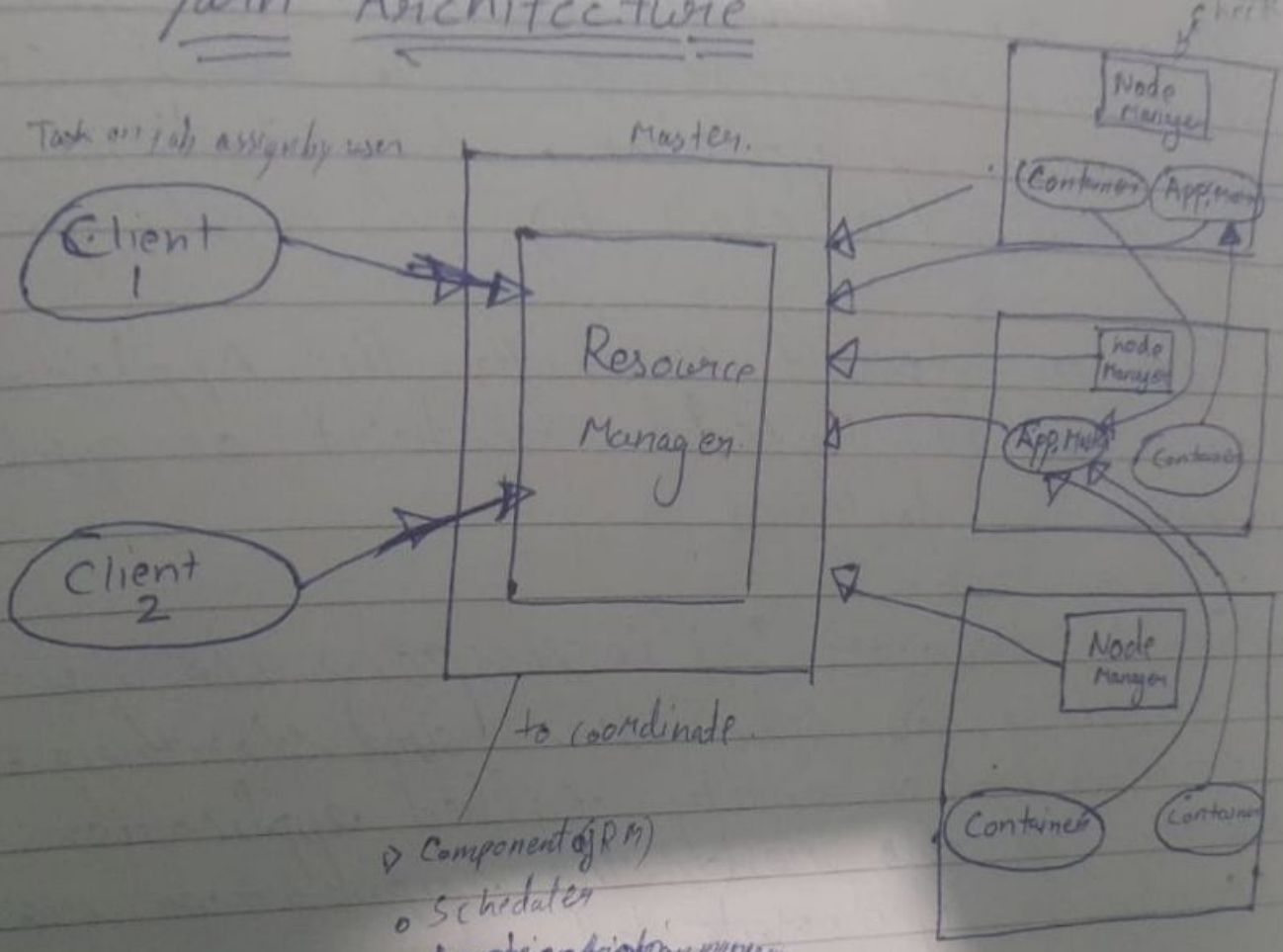
Applications contain some specific Fs Apis which are executed by code written by client and it communicate with GFs master and Chunk server is established.

- No ~~caching~~ caching of file data is performed by clients server ——x—— because of streamed ~~world~~ work load. Caching doesn't benefit clients.
- server has atleast a buffer cache to maintain a record for frequently requested files locally.

Big Data.

24/10/2024
Thursday, Oct 10

# Yarn Architecture




- Component of (RM)
  - Scheduler
  - Application manager
    - assign we assign

# Resource Manager

Ultimate authority that arbitrates resources among all the applications.
It has 2 major components:

① Scheduler: it is responsible for allocating resources such as disk, CPU and network running application, subject to restrictions imposed by queues & capacity.
* It doesn't monitor the application, nor does it initiate restarts on application or hardware failures.

② Application Manager: it is responsible for accepting job submissions & starting a container for an entity called the Application Master. It also restarts the Application Master container if it fails.

③ Node Manager: it is an agent that runs on every machine in the cluster. It is responsible for launching containers on their machine & managing the use of resources by the container. It reports the usage back to the Scheduler of the resource manager.

④ Containers: A single node houses a set of CPU, RAM, & other CPU hungry resources. The lifecycle of containers is managed by Container Launch Contacts & resources are made available to applications for certain purposes on a given host.

FREEMIND

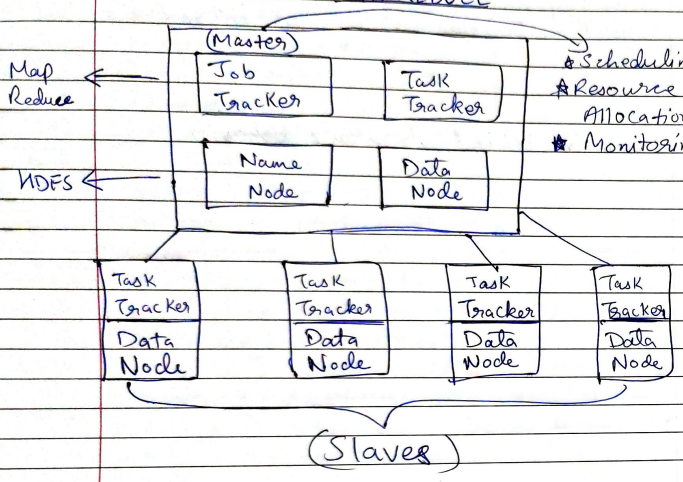
Date

Page

(5) Application Master: it maintains a registry of running applications & monitors their execution.
Whenever a job is submitted to the framework, an Application Master is elected for it, which is in charge of allocating resources from Resource Manager to Node Manager, which then monitors & executes the Task.

[BD]

Unit-4 : MAPREDUCE




formula: PD (Propagation Delay) = Distance / velocity

i) Map Reduce performs the processing of large datasets in a distributive & parallel manner.

ii) It consists of 2 distinct tasks : Map & Reduce.

iii) 2 essential labels of MapReduce are : Job Tracker & Task Tracker.

PROCESS

Input → Splitting → Mapping → Shuffling → Reducing → Output (Final)

# [BD]

## {Map Reduce Programming}

(Driver Node)

This is the main entry point of a Map Reduce Job. It specifies configuration, sets up the job & defines which network & reducer classes to use. It also sets an input & output paths in HDFS & coordinates the job execution.

"Code"

on whatsapp....

(Mapper Code)

A mapper is the first stage in Map Reduce. It processes each input data split, producing Key-value pairs. It is highly parallelisable, meaning multiple mappers can run simultaneously on different data chunks. The key-value pairs generated by mappers are sent to reducers for further processing.

"Code" on whatsapp....

Page

(Reducer Code)
The Reducer takes the key-value pairs generated by mappers and groups them by key and performs an operation on each group (like aggregation or summing values).
The output is another set of key value pairs that represent the final result.

"Code" on whatsapp...

## {COMBINER}

It is an optional mini-reducer that Hadoop runs between the mapper & reducer phases.
It's role is to reduce the volume of data that needs to be transferred across the network to the reducers.
It aggregates / combines the mapper output, locally on each node before sending it to the reducer, which helps in improving efficiency & reducing network traffic.

Eg:- If the manual output is :

apple 1
banana 1
apple 1

The combiner will aggregate it to:

apple 2
banana 1

This data is then transferred to reducer.

## {Partitioner}



It is responsible for deciding how to distribute the data among different reducers.
It determines which reducer will process each mapper output based on the key, ensuring that records with the same key go to the same reducer.

Eg:- If we are processing Sales data of multiple cities, the partitioner can be set up so that all the records for a single city goes to the same reducer. This way Delhi records go to one reducer while Mumbai records go to another, facilitating parallel processing and aggregation.

## {Sorting}

It is closely tied to the shuffling phase and occurs automatically as part of it.
In this step, the data arriving at each reducers is sorted by Key in ascending order.
Sorting makes it easier for the reducer to aggregate / process data since all records for each key are together and ordered.

Ex:-
apple 1
mango 1
banana 1
apple 1
banana 1
apple 1

=>

Sorting:-
mango 1
banana 1, banana 1
apple 1, apple 1, apple 1

## {Shuffling}

It is the phase where the mapper's output is transferred to the appropriate reducers based on the partitioner's decisions. It's responsible for ensuring that all records with the same key end up on the same reducer, regardless of where they were originally generated.

Eg:- In a word count program, if the word 'apple' appears on multiple nodes, shuffling ensures that all the occurances of 'apple' are sent to the same reducer for counting.