__/__/____

Principles of Big Data Management (CSE-417)

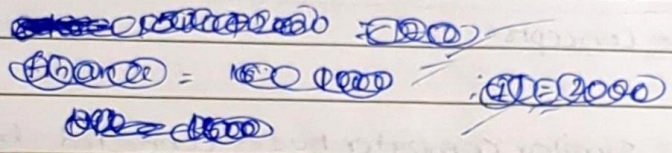
Unit-I

Big Data characteristics:

Volume, Velocity, Variety, Veracity, Value
" " " " "
size speed of Variou Quality usefulness
MB, TB Data types of Data
PB Emails, tweets of Data
Structured, unstructured, semi-structured
RDBMS : Multimedia = web pages, *csv

* Big Data:

It is a branch/field dedicated to the analysis, processing and storage of large ~~volumes~~ collections of data that frequently originate from disparate sources.



Analysis: It is a process of examining data inorder to find the relationships, patterns, insights and trends among the data

Analytics: It includes development of analysis methods, scientific methods and Automated tools used to manage the complete data life cycle.

Goal of Data Analysis is to support better decision making

Different types of Data Analytics:

- Descriptive : Events that have already occured Eg: sales volume over past 12mon ths
- Diagnostic : aims to Determine cause of phenomenon in past (reason behind)
- predictive :- predict outcome (based on patterns found in historical & present)
- prescriptive :- Actions should be taken after results of predictive

Descriptive Ex: Monthly ~~Sales~~ Commission earned by Sales Agent

Diagnostic: Ex: Item 2 sales less than item 3 why?

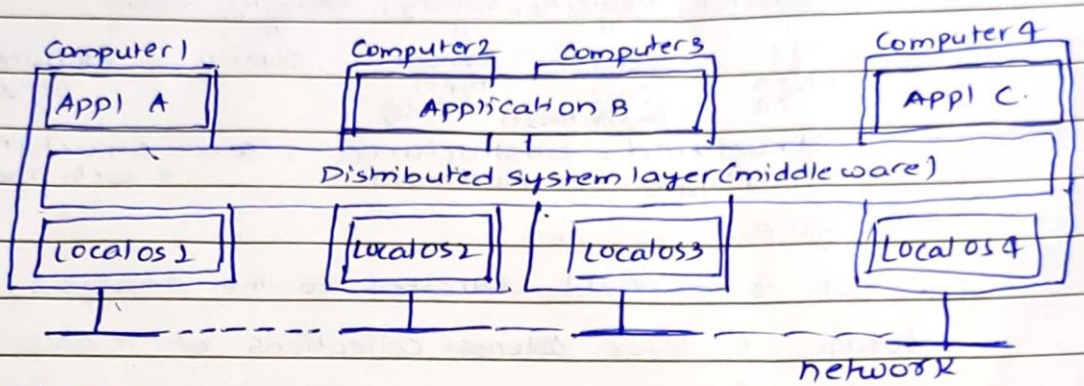
Predictive: Patient survival rate if Drug B is administered instead of Drug

prescriptive: Among 3 drugs which one provide best results

__/__/____

Distributed System (DS):

It is a collection of independent computers that appears to its users as a single coherent system
- computing power & storage is shared.



Advantages:
- scalability - Reliability - Availability - Communication

Big Data Storage Concepts:

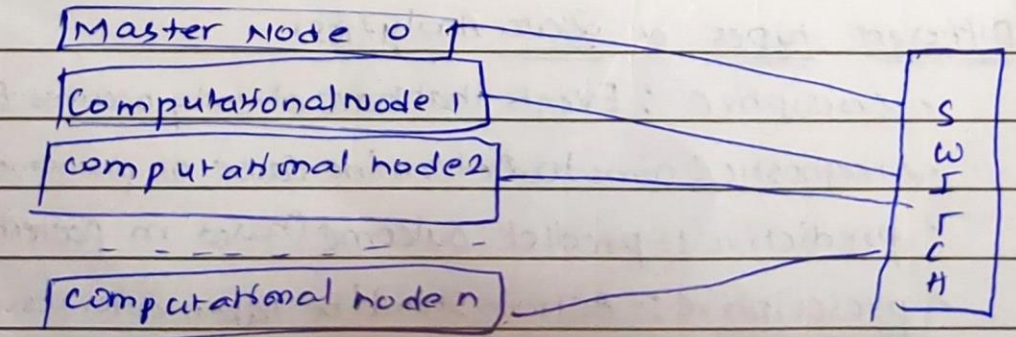
Cluster systems:

- Collection of similar computer nodes connected by high speed LAN
- Nodes run on similar OS.

There will be 2 nodes

1) Master Node ⇩ It control storage allocation and Job scheduling
2) Computational node.

- Computing power and storage are shared among users.



__/__/____

File system:

- A File is named collection of related info that is recorded on secondary storage (Hard drives, DVD's)
- A File consists of data or Program

two parts

i) collection of files
For storing data & programs

ii) Directory structure
provides info abt all files in the system

- Fs is responsible for, organization, storage, naming, retrieval, sharing & protection of files.

Centralized File system:

FS which stores files & directories in a single computer system is known as CFS

Disadvantage: - storage is limited

Distributed Filesystem (DFS):

- Fs that manages the storage across a network of machines is called DFS
- It is a client/server based application that allows client to access and process data stored on the server
- A DFS has to provide single system image to the clients even though data is stored in multiple computer systems

Advantages:

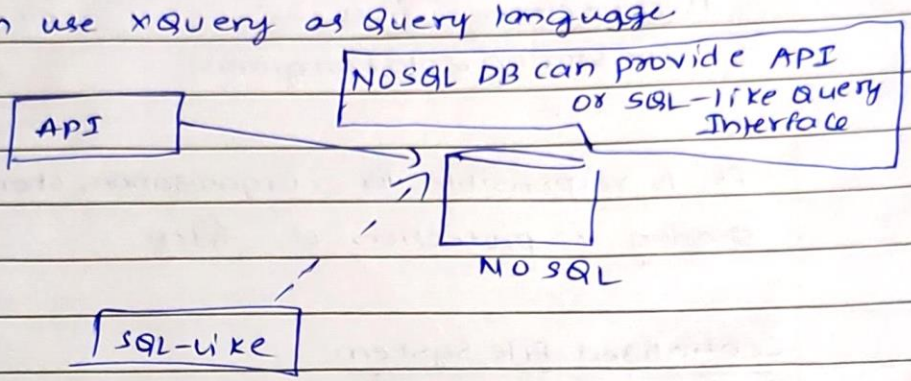
- Network Transparency: client doesn't know abt location of files. clients can access files from any computer connected in network.
- Reliability: Keeping multiple copies for file.
- performance: parallel access of data is possible
- Scalable: Can add ~~multiple storage~~ computer systems in network

__/__/____

NOSQL (Not only SQL)

~~non~~ - It is a non-relational Database that is highly scalable, specifically designed to sent structured & unstructured data.

A NOSQL DB is optimized to store XML files will often use XQuery as Query language



Sharding:

It is process of horizontally partitioning a large dataset into smaller, more manageable datasets called "shards". These shards are distributed across multiple nodes where a node is a server.

- Each shard is stored on a seperate node.
- And each node is responsible for only the data stored on it.
- And All shards represent complete dataset.

id	name	DOB
1	Bob	6may
2	Helen	23nov
3	John	15 Feb
4	Jane	4July

Node A

id	name	DOB
1	Bob	6may
2	Helen	23 nov

Shard A

server (or) machine

Node B

id	name	DOB
3	John	15Feb
4	Jane	4July

Shard B

Explain Diagram

Ex:

Advantages: In case of a node failure, only data stored on that node is affected.

__/__/____

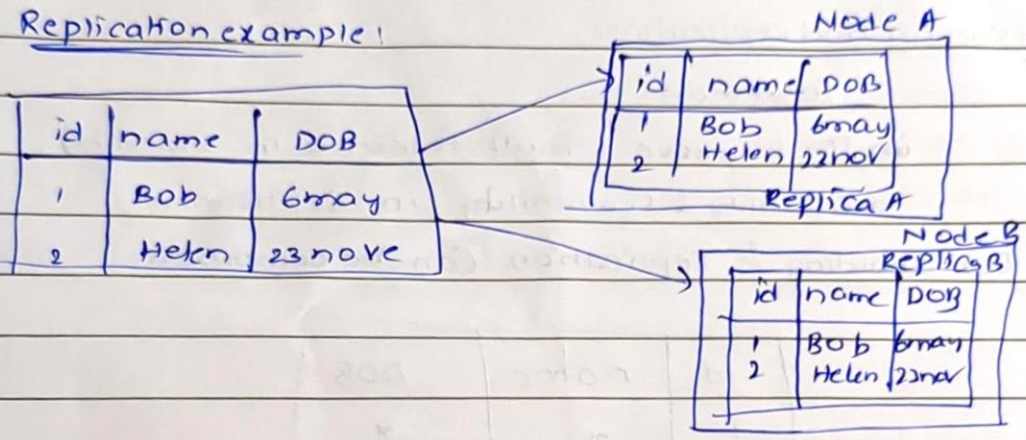
Replication: It stores multiple copies of a dataset, known as replicas, on multiple nodes.

- It provides scalability and availability, due to the fact that the same data is replicated on various nodes.
- Fault tolerence is also achelved since data redundacy ensures that data is not lost when an individual nodes fails.

There are two methods that are used for replication

(1) Master-slave (2) peer-peer.

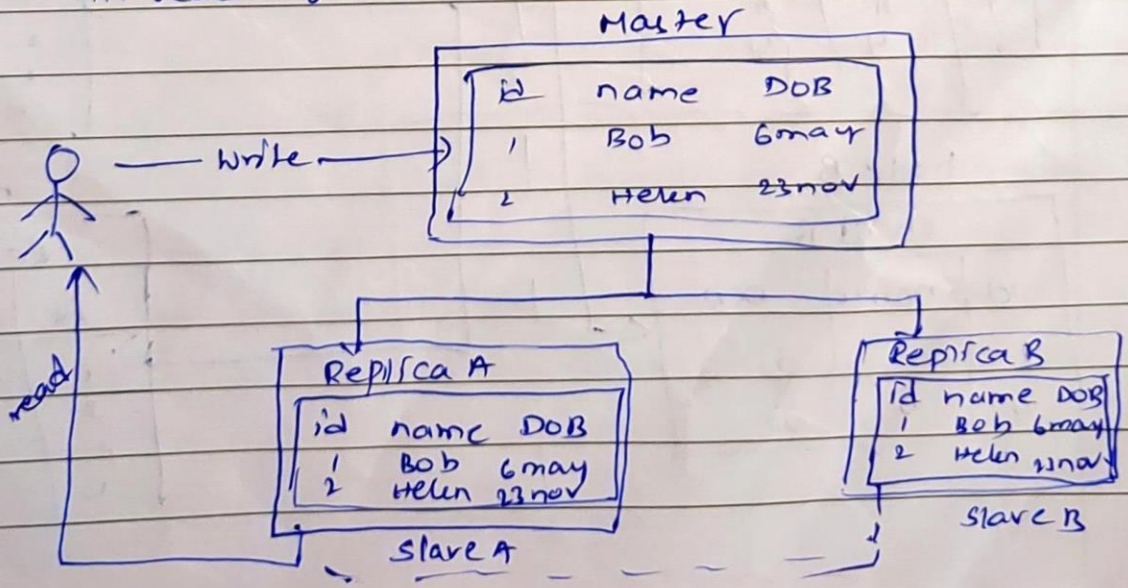
Replication example:



(1) Master-slave:

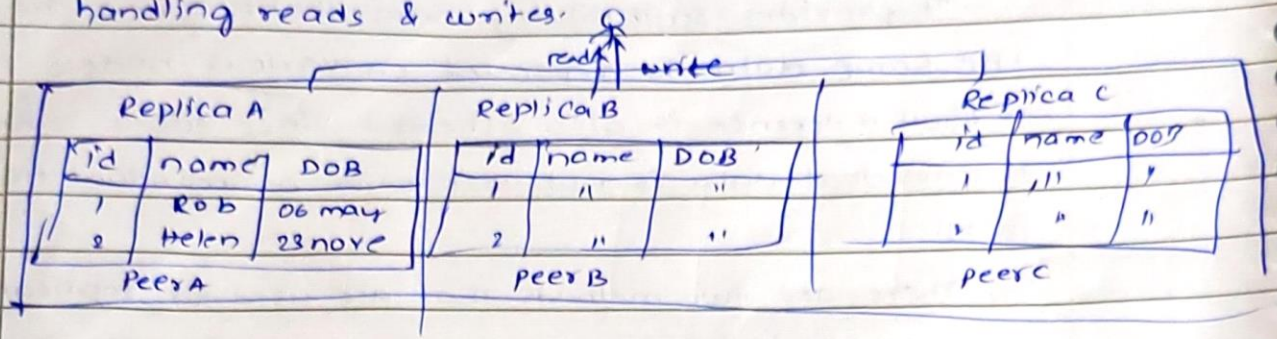
Here nodes are arranged in master slave configuration and all data is written in master node

- Then data is replicated to multiple slave nodes.
- All write requests including insert, update and delete, occur in Master node
- All read requests are fullfiled in Slave node.



__/__/____

Peer-peer: All nodes operates at same level (no-master & slave)
Each node is a peer and equally capable of handling reads & writes.

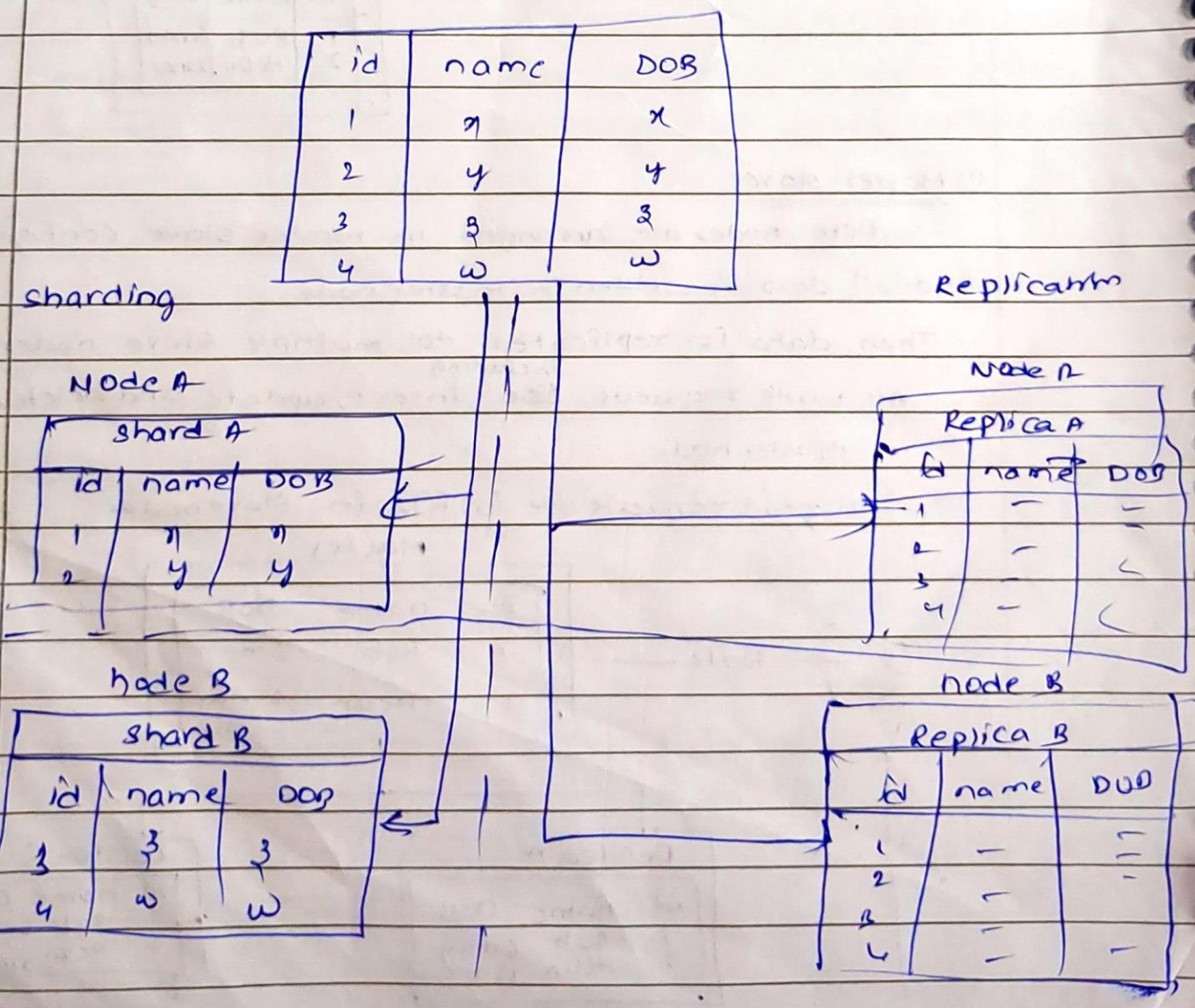


Sharding and Replication

~~To remove~~

To improve fault tolerence in sharding, avaliability & scalability in Replicaton.

Both sharding & Replicaton can be combined.



__/__/__

CAP Theorem: / Brewer's Theorem:-

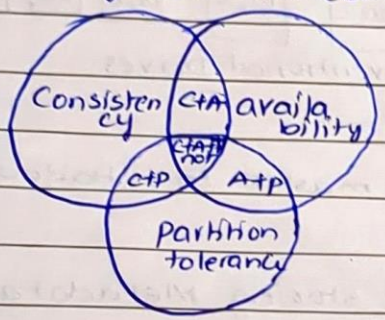
Consistency, Availability, and Partition Tolerance

Consistency: A read ~~~~ from any node results in same data across multiple nodes. diagram

Availability: A r/w request will always be "acknowledged" in the form of a success or failue.

Partition tolerance: The database system can tolerate communication outages that split cluster into multiple silos and can still serve read/write requests.

If any communication outages and still perfos ~~to~~ r/w



cap theorem states that a distributed database system can only provide two of three properties.

BASE:- It is a database design principle based on the CAP theorem

BASE = basically available, soft state, eventual consistency

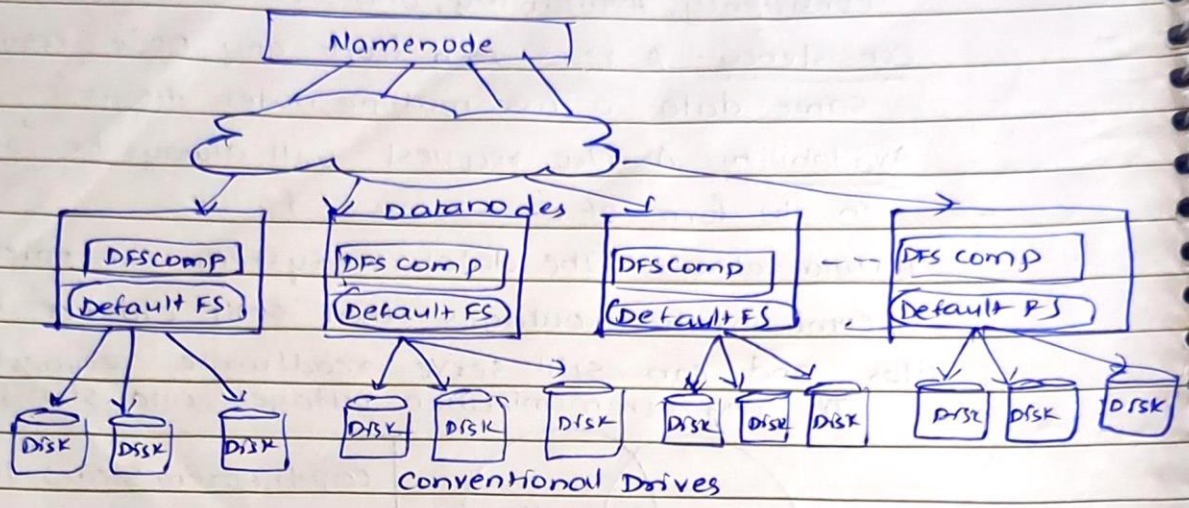
Bastically Available: This property refers to fact the The database will acknowledge client's request in the form of success or failure notification.

Soft state: This property refers to the fact that the state of database can change overtime, even without any explicit user interventon. This happens due to the effects of backgroundprocess and updates to data

Eventual Consistency:- ~~This property refers to the fact that~~ is the state in which reads by different clients, immediately following a write to the database, may not return consistent results

__/__/____

Hadoop Distributed filesystem:

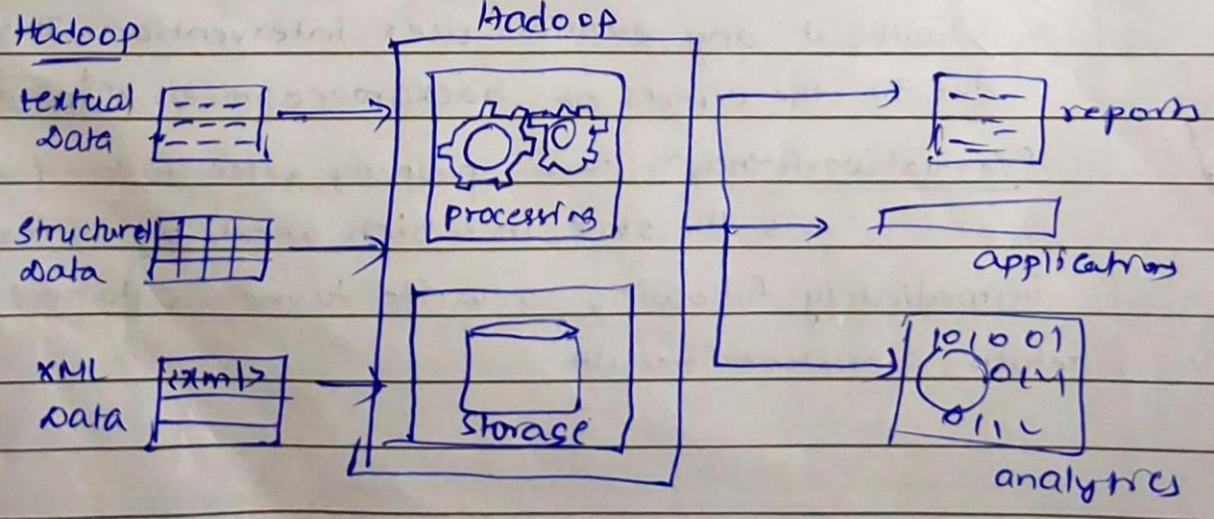


Name Node (Master)

- It works as a master in a Hadoop cluster that guides Datanodes
- Mainly used for storing Metadata (location of file, size of file etc.)

Datanode (Slave)

- works as a slave mainly used for storing Data in a hadoop cluster
- Runs HDFS client program
- Executes Map & Reduce tasks



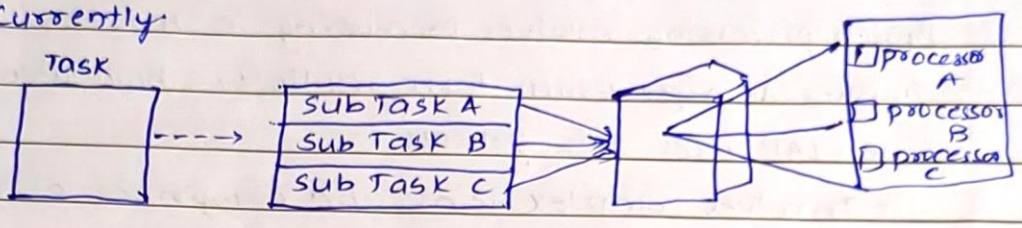
__/__/____

Unit - II

Big Data processing Concepts:

Parallel Data Processing:

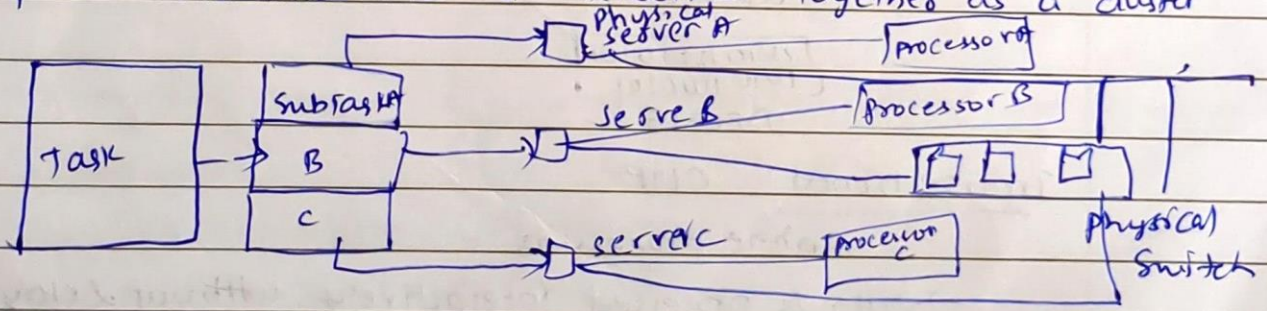
- It involves simultaneous execution of multiple tasks that collectively comprise a larger task.
- The goal is to reduce the execution time by diving single larger task into multiple smaller tasks that run concurrently.



A task can be divided into 3 subtasks that are executed in parallel on 3 diff processors within the same machine.

Distributed Data processing:

- It is also closely related to parallel data processing in that the same principle of "Divide & Conquer" is applied
- However, Distributed is always achieved through physically seperate machines that are networked together as a cluster



Hadoop:

It is a open source framework for large scale data storage.

→ It is versatile framework that provides both processing and storage capabilities.

__/__/____

Processing Workloads:

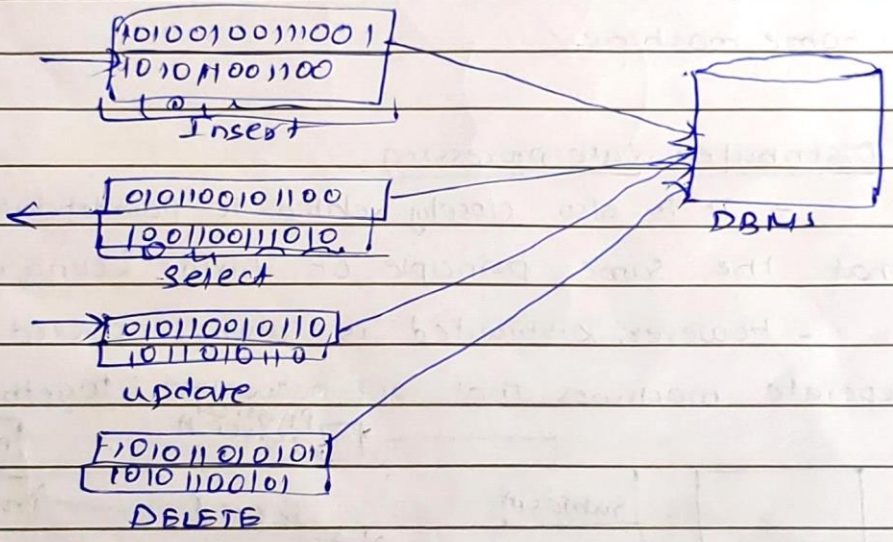
It is the amount & nature of data that is processed within a certain amount of time.

→ Two types

(1) Batch / offline processing (2) Transactional

Batch processing involves processing in batches and usually imposes delays which turn results in high latency responses.

- OLAP uses this process.
- Involves complex joins and comprises of group of row queries.



Transactional: OLTP

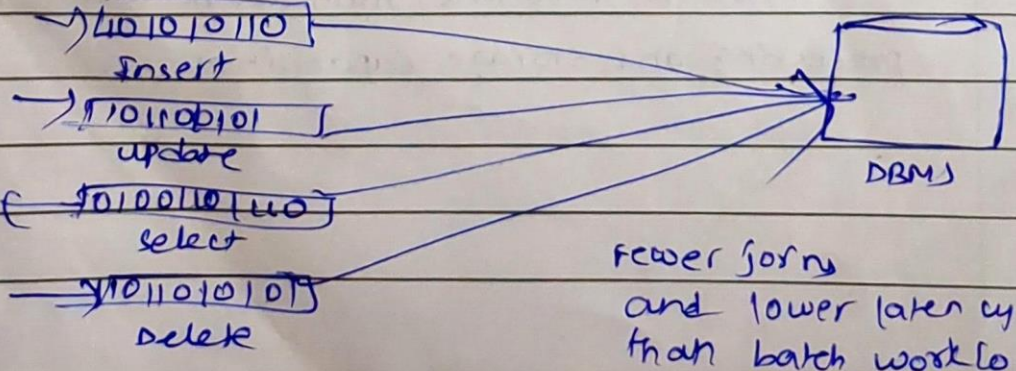
online processing

→ Data is processed interactively without delay.

→ low latency response.

→ small amt of data with random read & w

→ Fewer joins.



Fewer joins and lower latency responses than batch workloads

__/__/__

→ Majority of Big Data processing occurs in Batch Mode.
- Predictive, prescriptive in Batch processing.

Batch processing with MapReduce:

- MapReduce is a widely used Batch processing framework.
- It is highly scalable, Reliable and it is based on divide & conquer rule principle; which provides built in fault tolerance and redundancy.
- MapReduce can process schema less Datasets.

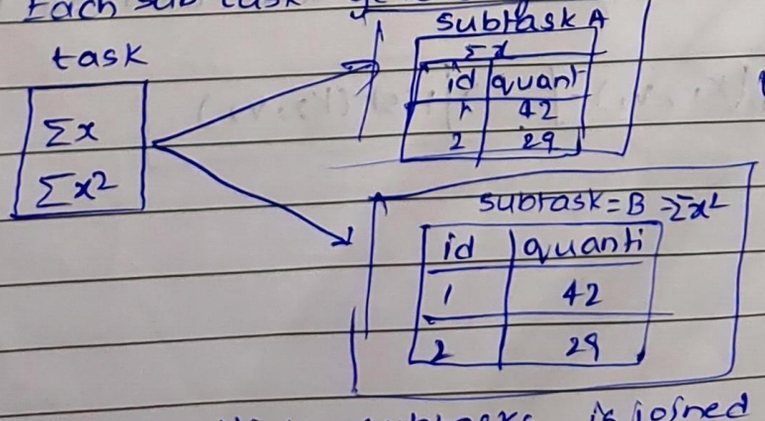
~~Map and Reduce Tasks~~:

MapReduce:

→ programming model for Data processing.
→ It is based on divide and Conquer principle
→ It divides big problem into smaller problem
→ A dataset is broken into multiple part and operations are performed on each part independently in parallel.

The divide & conquer is generally achieved using one of following approaches:

(1) Task parallelism: (Dividing Tasks)
- Dividing a task into subtasks and running each sub-task on a seperate node.
- Each sub-task generally executes a different algorithm.



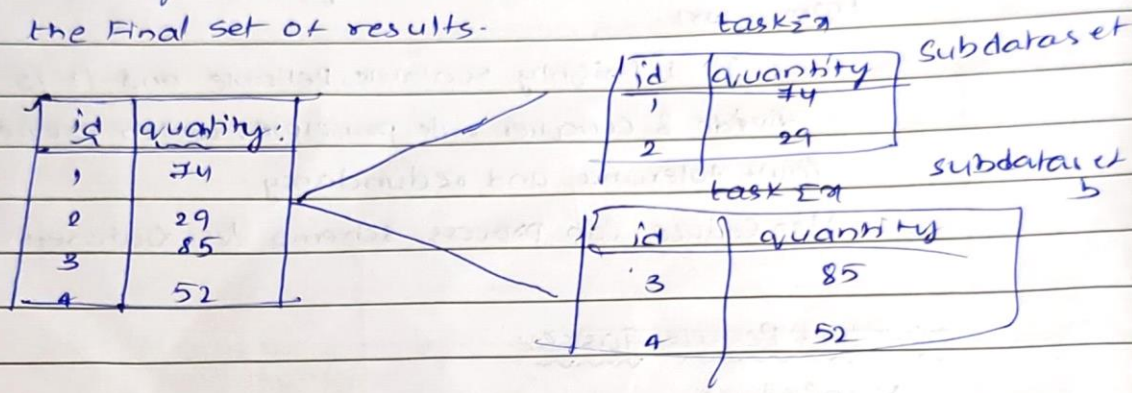
task div in to sub tasks and run on 2 different nodes on same Data.

- Finally multiple subtasks is joined together to obtain

__/__/____

Data parallelism: (Dividing dataset)

- Dividing dataset into multiple datasets and processing each subdataset in parallel
- All are processed using same algo.
- Finally subdatasets are joined together to obtain the final set of results.



id	quantity
1	74
2	29
3	85
4	52

Map and Reduce Tasks:

Map tasks: (optional)

- map
- combine
- partition

Reduce tasks

- shuffle & sort
- Reduce

Map:

→ The first stage is Map, here dataset file is divided into multiple smaller splits

→ Converted into key-value pair

Key - ordinal position of record; value - Actual record

→ Then these key-value pairs for each split are then sent to a map func.

→ map function executes user-defined logic.

$$[(K_1, V_1) \longrightarrow \text{list}(K_2, V_2)]$$

__/__/____

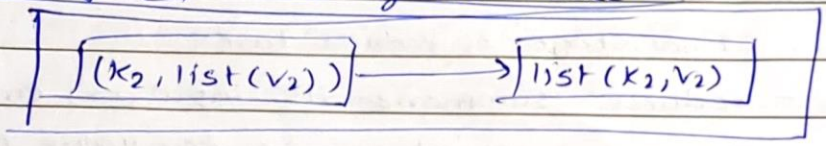
Combine:

Generally, output of the map func^n is handled directly by reduce func^n

→ However, Map and Reduce run on different nodes, This requires moving data btw Mappers and Reducers. This requires lot of bandwidth and processing latency

→ In large datasets, the time taken to move the data btw Mapper and Reduce will be excess.

↝ Due to this reason MapReduce engine provides an optional combine func^n. that "summarises mapped output before it gets processed by the reducer".



Partition:

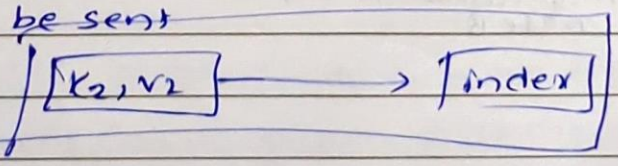
→ In this stage, if more than one reducer is involved a partitioner divides the output from the mapper or combiner into partitioner between reducer instances.

→ No. of partitions will be equal to number of reducers

→ It helps for distribution of key-value pairs.

→ It is last stage of map task

→ It returns index to which particular partition should be sent



Shuffle & sort:

- First stage of reducer task.
- During this stage output from all partitioners is copied across network to reducer nodes, this is known as shuffling
- MapReduce engine Automatically groups and sorts the K-V pairs according to keys, so that the output contains a sorted list of all input keys and their values.

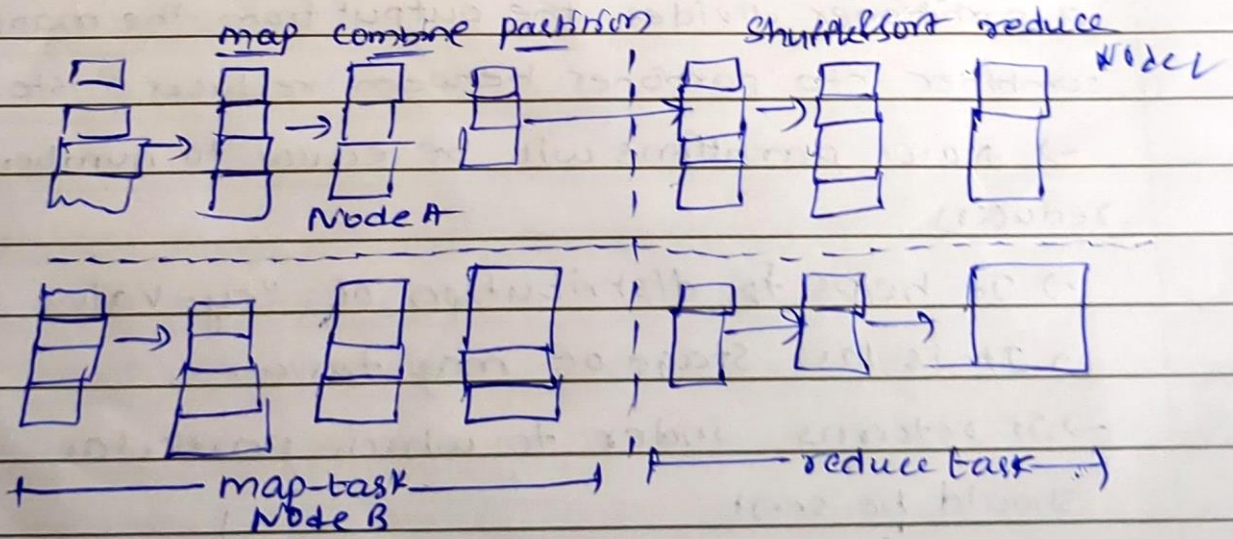
$(k_2, v_2) \longrightarrow k_2, \text{list}(v_2)$

Reduce:

Final stage of Reducer task.

- reducer - summarize its input (or) emit the output.
- The output of reducer i.e; key-value pairs is return out as a seperate file.

$(k_2, \text{list}(v_2)) \longrightarrow \text{list}(k_2, v_2)$



__/__/____

Unit-4

A [customer] has placed many [orders]

Customer
a

Cid	Cname	Cage	Caddr	Csal
101	xyz	30	HYD	25K
102	PQR	45	BAN	30K
103	ABC	32	DEL	48K
104	DEF	27	MUM	52K

order
b

Oid	Odate	Oamt	Cid
201	10Jun	5300	101
202	21 Apr	2400	102
203	3 Aug	3841	103
204	5sep	2728	104

→ hdfs dfs -get /hivejoins/customer.csv
→ hdfs dfs -get /hivejoins/order.csv

hive

```
hive > create table customer_join (Cid INT, Cname STRING,
        Cage INT, Caddr STRING, Csal float)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
hive > LOAD DATA LOCAL INPATH 'customer.csv' INTO TABLE
        customer_join;
hive > create table order_join (Oid INT, Odate STRING, Oamt Float
        , Cid INT)
  ROW FORMAT DELIMITED FIELDS TERMINATED ',';
hive > LOAD DATA LOCAL INPATH 'order.csv' INTO TABLE
        order_join;
hive > SELECT a.cid, a.cname, a.cage, ~~a.caddr, a.csal~~ b.Odate
      from customer_join a LEFTOUTERJOIN order_join b.
      ON (a.cid = b.cid)

hive > SELECT a.cid, a.cname, a.cage, b.Odate FROM
      customer_join a JOIN order_join b. ON (a.cid = b.cid);
```

sum of squares

mapper :

```
function mapper(input_list):
    output_list = []
for index, number in enumerate(input_list)
    square = number * number
output_list.append((index+1, square))
return output_list
```

reducer

```
function reducer(input_list):
    sum_of_squares = 0
for key, value in input_list
    sum_of_square += value
return sum_of_squares.
```