

"Unit III: Big Data Processing"

Unit III	BIG DATA PROCESSING	(06 Hrs)
Big Data Analytics- Ecosystem and Technologies , Introduction to Google file system, Hadoop Architecture, Hadoop Storage: HDFS, Common Hadoop Shell commands, Anatomy of File Write and Read, NameNode, Secondary NameNode, and DataNode, Hadoop MapReduce paradigm, Map Reduce tasks, Job, Task trackers - Cluster Setup – SSH & Hadoop Configuration, Introduction to NOSQL, Textual ETL processing.		

- Q1)** a) Explain Big data Ecosystem with suitable diagram. [7]
b) Explain anatomy of File read and write in HDFS. [7]
c) Write and explain any two Hadoop shell commands. [4]

OR

- Q2)** a) Explain Map Reduce with proper diagram for word count example. [7]
b) Explain Google file system. [7]
c) Explain ETL processing. [4]

- Q1)** a) Explain Google file system and its advantages. [10]
b) Explain Hadoop distributed file system [8]

OR

- Q2)** a) Why map reduce is required in Hadoop? Explain the stages involved in map reduce task with a suitable example? [9]
b) Describe the various types of NoSQL Databases with example and also compare them. [9]

- Q1) a)** Explain all the steps for writing a file in HDFS with neat diagram. **[8]**
- b)** Describe the various types of NoSQL Databases with example and also compare them. **[10]**

OR

- Q2) a)** Why map reduce is required in Hadoop? Explain the stages involved in map reduce task with a suitable example? **[9]**
- b)** What is Hadoop Distributed system? What is the advantage of heart bit message in Hadoop. **[9]**

4) Assume suitable data, if necessary.

- Q1)** a) i) Explain Google File system and its advantages. [5]
ii) Explain ETL in Big data. [5]
b) Explain Hadoop distributed file system. [8]

OR

- Q2)** a) i) Write 5 Hadoop Shell commands. [5]
ii) Explain Role Job tracker and Task Tracker in Hadoop Architecture. [5]
b) Explain Map Reduce with proper diagram for word count example. [8]

- Q1)** a) Explain the process of reading and writing a file in HDFS with neat diagram. [8]
- b) List and explain any four Hadoop shell commands with syntax. [4]
- c) Differentiate between SQL and NoSQL databases with example. What is the need to develop big data applications using NoSQL databases? [6]

OR

- Q2)** a) What is the need of map reduce in Big Data? Explain the stages involved in map reduce task with a suitable example? [9]
- b) Explain Hadoop ecosystem in detail. [9]

Big Data Analytics- Ecosystem and Technologies

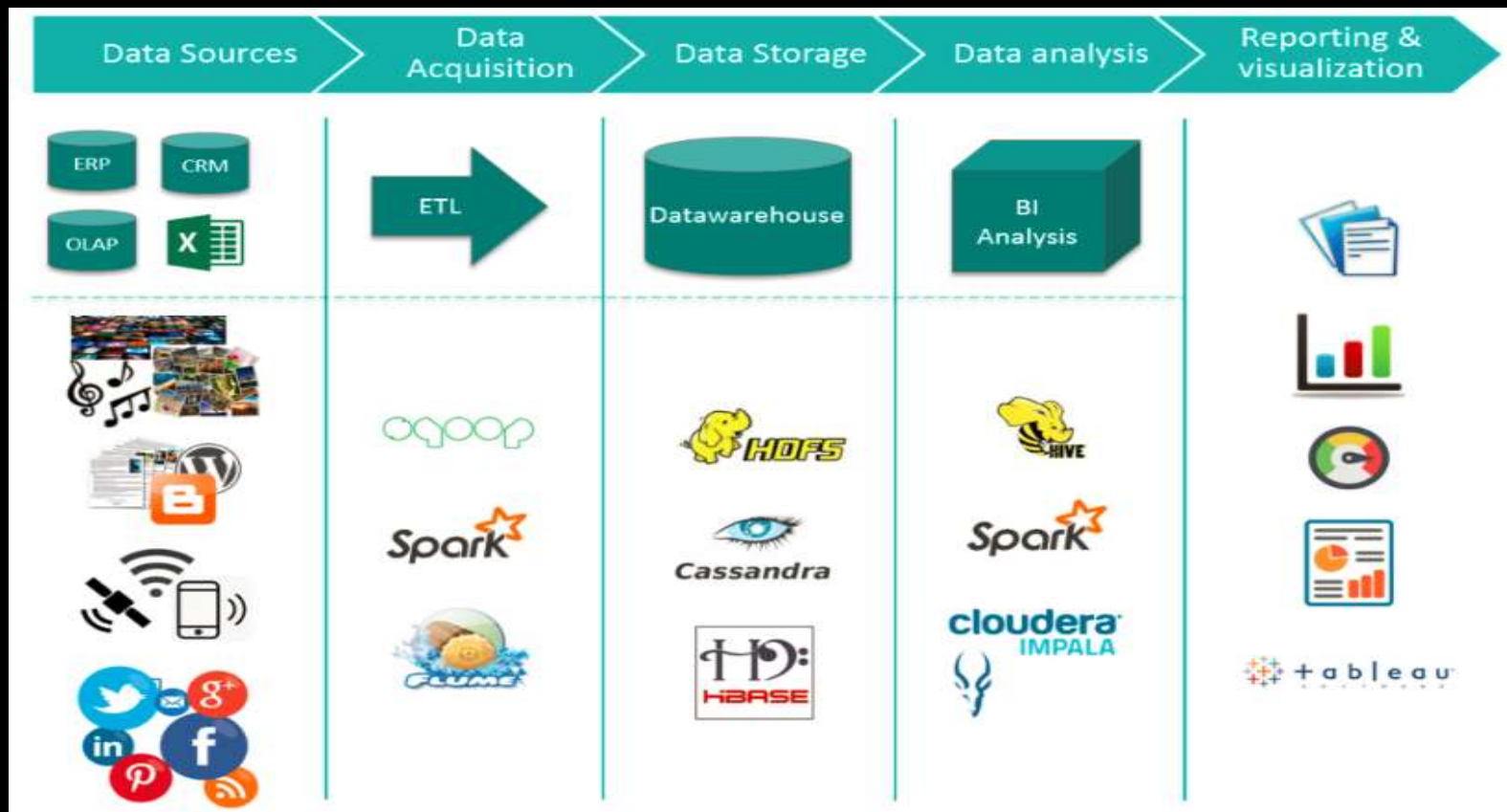
◆ What is Big Data Ecosystem?

The **Big Data Ecosystem** is a collection of **tools, frameworks, and technologies** that help in collecting, storing, processing, and analyzing large volumes of data.

Ye ecosystem ek **structure** provide karta hai jisme different tools alag-alag roles play karte hain, jaise:

- Data ko collect karna (ingestion)
- Store karna (storage)
- Process karna (processing)
- Analyze karna (analytics)
- Aur results dikhana (visualization)

□ Diagram



1. Data Sources (Input Layer)

This is where data originates. It includes:

•Structured Data Sources:

- **ERP, CRM** (Enterprise/Customer systems)
- **OLAP** (Online Analytical Processing)
- **Excel spreadsheets**

•Unstructured & Semi-structured Sources:

- Multimedia (Images, audio, video)
- Blogs (WordPress, Blogger)
- Logs (Web server logs, system logs)
- Sensors (IoT devices)
- Social Media (Facebook, Twitter, LinkedIn, YouTube, Pinterest, etc.)

These are all raw data formats, and they need processing before analysis.

2. Data Acquisition (ETL - Extract, Transform, Load)

This layer is responsible for collecting and transferring data from sources into the system:

- **ETL Tools:** Used to extract data, transform it into usable formats, and load it into storage.
 - **Sqoop** – Transfers data between Hadoop and relational databases
 - **Flume** – Collects, aggregates, and transports large amounts of log data
 - **Apache Spark** – Also used in data acquisition for real-time streaming and batch processing

3. Data Storage (Distributed & Scalable Storage Systems)

This layer stores the huge volumes of processed data.

- **Data Warehouse** – Central repository for structured data
- **HDFS (Hadoop Distributed File System)** – Stores large files across multiple machines
- **Cassandra** – NoSQL database designed for scalability
- **HBase** – A non-relational, distributed database built on HDFS

These storage systems provide both structured and unstructured data storage.

4. Data Analysis (Processing & Query Layer)

This is where actual insights are generated through data processing and querying tools.

- **Hive** – SQL-like querying tool for Hadoop
- **Spark** – Fast, general-purpose cluster computing system for big data
- **Cloudera Impala** – Real-time querying engine

These tools perform data mining, filtering, aggregation, and pattern detection.

5. Reporting & Visualization (Output Layer)

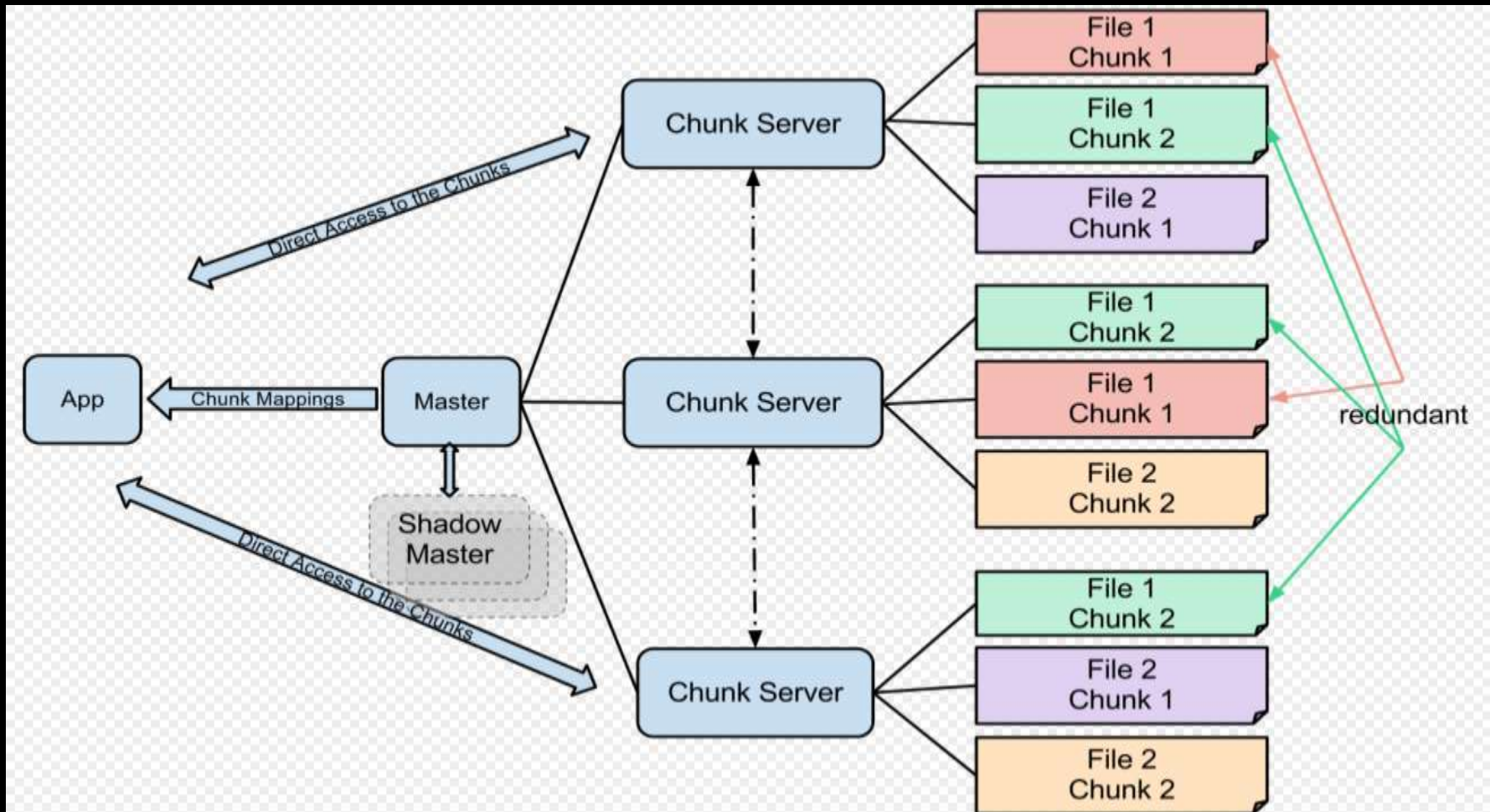
This is the final step where data is presented in human-readable form:

- Dashboards
- Charts, graphs, KPIs
- Tools like **Tableau** and other BI platforms to visualize patterns, trends, and summaries

Introduction to Google file system

□ Introduction to Google File System (GFS)

Google File System (GFS) ek **distributed file system** hai jo specially design kiya gaya tha Google ke large-scale data processing ke liye. Ye system handle karta hai **billion files aur petabytes of data** ko across multiple machines.



- **App (Application):**

- Ye user ya client ka software hai jo data ko access karta hai.

- **Master:**

- Ye ek main controller hai jo batata hai ki file kaunsa chunk server mein hai.
- Sirf metadata store karta hai (file name, chunk location, etc.).

- **Shadow Master:**

- Backup master, failover ke liye hota hai in case main master down ho jaaye.

- **Chunk Server:**

- Ye real data chunks store karte hain (file ka tukda).
- Har chunk ka multiple copies bana ke alag-alag servers pe rakha jaata hai for **redundancy**.

- **Chunks and Redundancy:**

- File ko chunks mein tod diya jaata hai.
- Har chunk ka 2-3 copies hota hai different chunk servers pe (shown as colored boxes).
- Agar ek server fail ho jaaye to data still available rahe.

- **App ↔ Chunk Server:**

- Master sirf location batata hai, app directly chunk server se data read/write karta hai (fast performance ke liye).

◆ Working of GFS (How It Works):

1. File is divided into **chunks** (small parts).
2. Each chunk is stored in different **Chunk Servers**.
3. The **Master** knows where each chunk is stored.
4. Client (App) asks Master for chunk location.
5. Then client directly accesses the **Chunk Server** to read or write data.
6. Each chunk has multiple copies on different servers (called **redundancy**) so that if one fails, others can be used.

Advantages of Google File System (GFS):

1. Fault Tolerance (Data loss nahi hota):

1. Agar ek server fail ho jaye to data fir bhi safe rehta hai kyunki har chunk ke multiple copies hoti hain (replication).

2. High Performance (Tez kaam karta hai):

1. Client directly chunk server se data leta hai, Master sirf location batata hai. Isse speed fast hoti hai.

3. Scalability (Asani se grow kar sakta hai):

1. System mein naye servers easily add kiye ja sakte hain without any problem.

4. Efficient for Large Files (Bade files ke liye best):

1. GFS specially design kiya gaya hai bade-bade files ko handle karne ke liye jaise videos, logs, big documents, etc.

5. Automatic Recovery (System khud theek hota hai):

1. Agar koi chunk ya server down ho jaye, GFS automatic naya copy bana deta hai.

✗ Disadvantages of Google File System (GFS):

1. Not for Small Files (Chhoti files ke liye suitable nahi):

1. GFS mainly large files ke liye banaya gaya hai. Small files pe use karna inefficient ho sakta hai.

2. Single Point of Failure (Master fail ho gaya to problem):

1. Master server agar down ho gaya to puri file system ruk sakti hai, although Shadow Master help karta hai.

3. Complex Implementation (Banana aur manage karna tough hai):

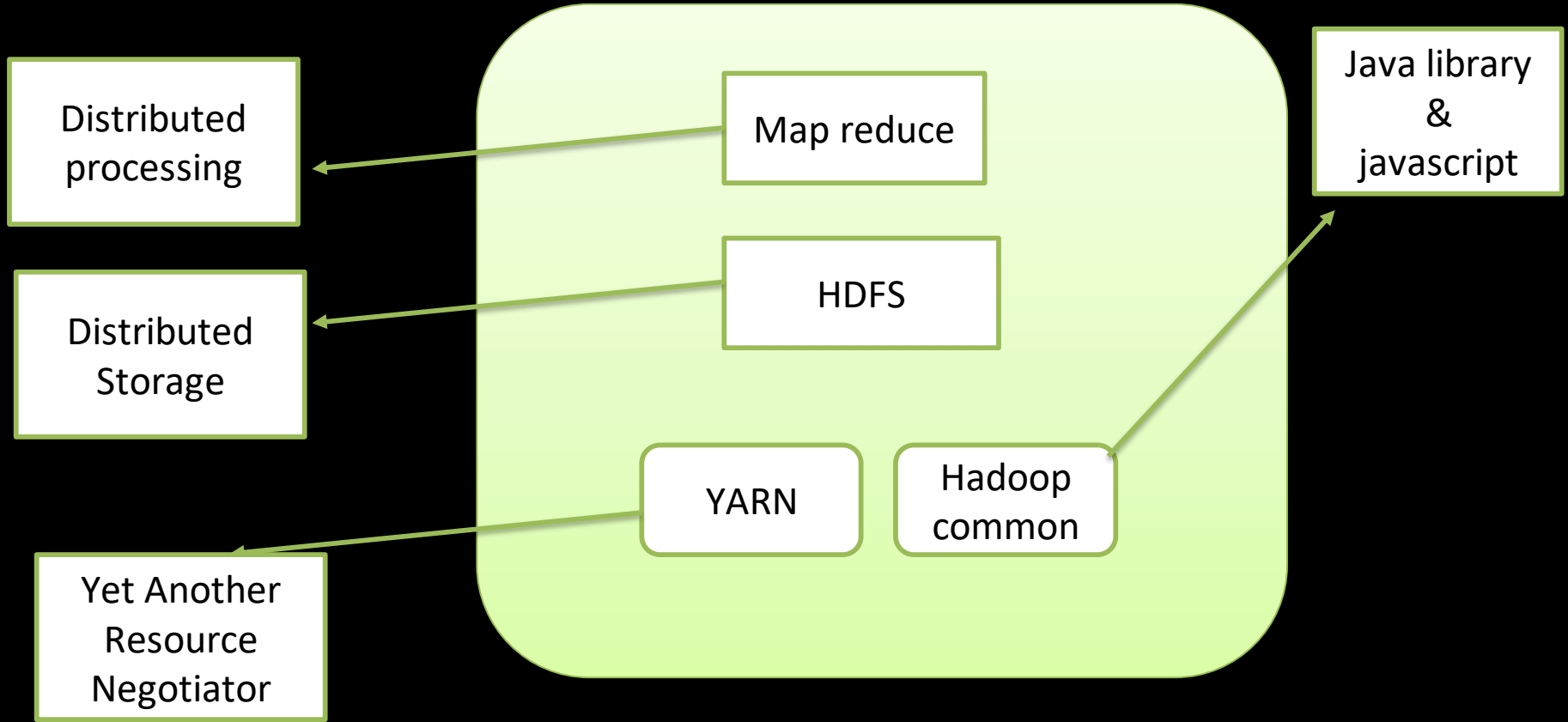
1. GFS ka design aur working complex hai, normal systems ke comparison mein.

4. High Network Usage (Zyada network use karta hai):

1. Chunk replication aur multiple server communication ki wajah se network load badh jaata hai.

Hadoop Architecture

Hadoop ek **open-source framework** hai jo **large data ko store** aur **process** karne ke liye use hota hai, especially jab data bahut zyada (Big Data) ho. Hadoop ka architecture **4 major parts** mein divide hota hai:



□ Hadoop Architecture –

Hadoop architecture 4 main parts mein divided hoti hai. Ye sab milke **Big Data** ko efficiently **store** aur **process** karte hain.

◆ 1. HDFS (Hadoop Distributed File System)

- Ye Hadoop ka **storage system** hai.
- Large files ko chhote blocks mein tod ke **different computers (DataNodes)** pe store karta hai.
- Isse kehte hain **distributed storage**.

Example: Agar 1 GB file hai, to usse 128 MB ke blocks mein tod ke multiple servers pe store karega.

◆ 2. MapReduce

- Ye Hadoop ka **processing engine** hai.
- Data ko parallel process karta hai across multiple nodes.
- 2 steps hoti hain:
 - **Map** → Data ko key-value pair mein todta hai.
 - **Reduce** → Similar keys ko combine karke final result banata hai.

Use: Fast data processing, jaise log analysis, word count, etc.

◆ 3. YARN (Yet Another Resource Negotiator)

- Ye Hadoop ka **Resource Manager** hai.
- Job scheduling karta hai (kaunsa task pehle chalega).
- System ke resources (RAM, CPU) ka dhyan rakhta hai.

Work: Job ko assign karta hai available machines pe.

◆ 4. Hadoop Common

- Isme **Java libraries and tools** hoti hain jo Hadoop ke baaki components ko support karti hain.
- Ye ek **shared utility layer** hai sabke liye.

Component

Kaam (Function)

HDFS

Data ko tod ke alag-alag servers pe store karta hai (Storage)

MapReduce

Data ko fast process karta hai (Processing)

YARN

Job scheduling aur resource manage karta hai

Hadoop Common

Java libraries aur tools provide karta hai

💡 Extra Components:

- **JobTracker (Master)** – Job scheduling aur monitoring karta hai (MapReduce ke liye).
- **TaskTracker (Slave)** – Actual task run karta hai on each DataNode.

Layer	Component	Role
Storage	NameNode	Stores metadata
	DataNode	Stores actual data blocks
Processing	JobTracker	Schedules and monitors jobs
	TaskTracker	Executes tasks

◆ 1. HDFS (Hadoop Distributed File System)

◆ 1. HDFS (Hadoop Distributed File System)

- Ye Hadoop ka **main storage system** hai.
- Big files ko **small blocks (default 128 MB)** mein todta hai.
- Har block ki **multiple copies (replication)** banata hai for safety.
- Works on **Master-Slave model**.

Component

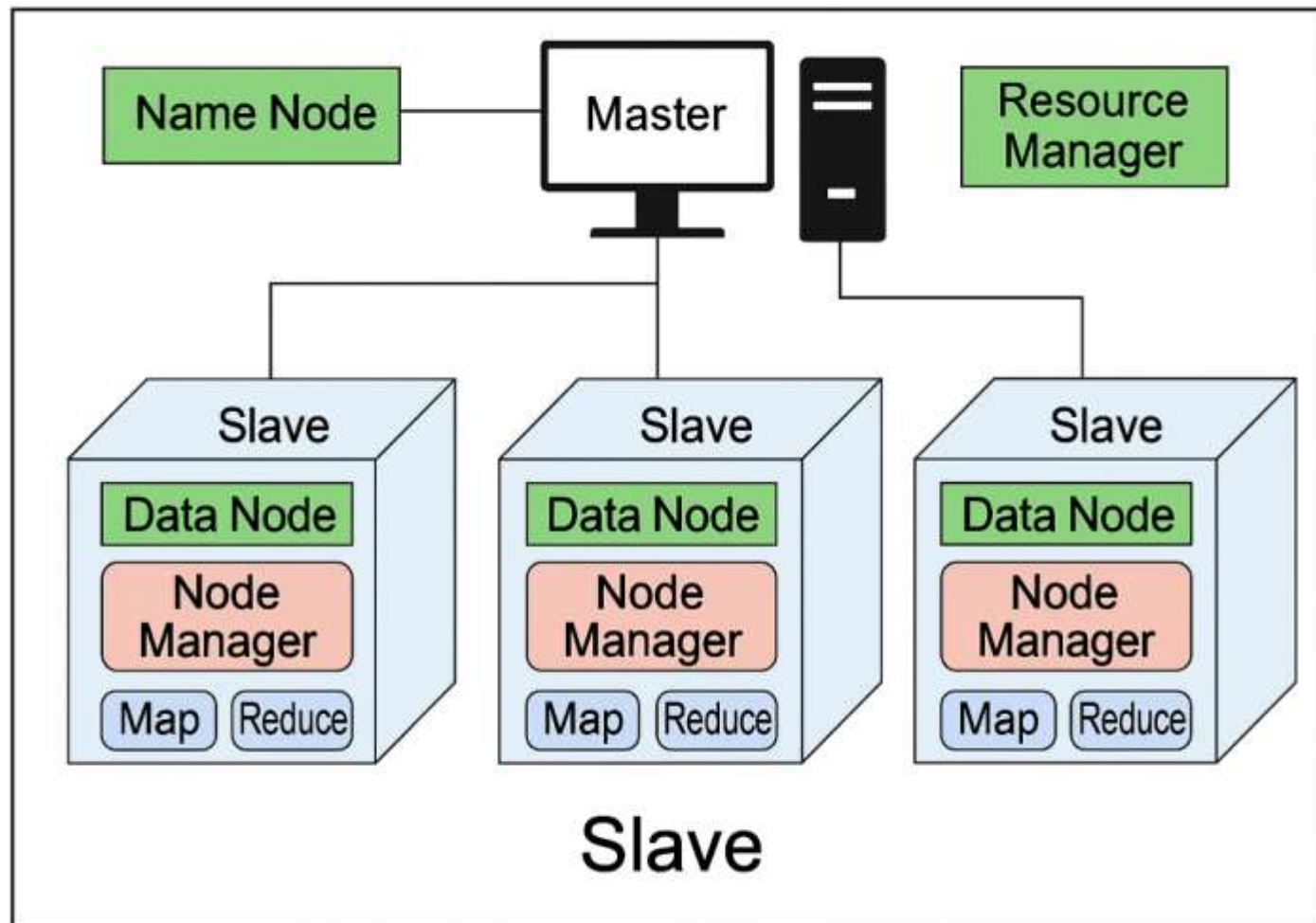
Role (Kaam)

NameNode

Master server – metadata store karta hai (block location, file name etc.)

DataNode

Slave servers – actual data blocks store karte hain



📁 Hadoop Distributed File System (HDFS) Architecture –

◆ Main Parts of Diagram:

□ 1. Master Node

- Master node ke andar do important parts hote hain:
 - **NameNode** → File system ka master, har file/block ka record rakhta hai.
 - **Resource Manager** → Job scheduling aur cluster resource manage karta hai (YARN ka part).

📁 2. Slave Nodes (Multiple Machines)

- Ye actual data aur job ko handle karte hain.
- Har Slave node mein hota hai:
 - **DataNode** → Data store karta hai (HDFS ka part).
 - **Node Manager** → Resources ko manage karta hai aur job execute karta hai (YARN ka part).
 - **Map & Reduce Tasks** → Data ko process karte hain (MapReduce engine).

Working Flow :

1. Jab ek file Hadoop mein daali jaati hai, to **NameNode** us file ko small blocks mein todta hai.
2. Ye blocks **DataNodes** pe store hote hain (multiple copies for safety).
3. Jab data process karna hota hai:
 1. **Resource Manager** decide karta hai ki kaunsa job kahan chalega.
 2. **Node Managers** uss job ko **Map & Reduce** tasks ke through run karte hain.
4. Result finally collect hoke user ko milta hai.

Component	Role (Kaam)
NameNode	Metadata store karta hai (file/block info)
DataNode	Actual data blocks store karta hai
Resource Manager	Job scheduling aur resource handling karta hai
Node Manager	Local node pe job execute karne mein help karta hai
Map/Reduce	Data processing karta hai

◆ 2. Hadoop Common:

Definition:

Hadoop Common ek set hota hai basic tools aur libraries ka jo Hadoop ke sabhi components ko chalane ke liye zaroori hote hain.

Simple Words:

- Ye ek base ya foundation hai Hadoop ecosystem ka.
- Isme Java libraries hoti hain jo HDFS, MapReduce, YARN jaise tools ko support karti hain.
- Common configuration files aur utilities bhi yahin hoti hain.

Hadoop Shell commands

Hadoop provides a command-line interface (CLI) called the **Hadoop Shell** to interact with its distributed file system (HDFS) and other components. Below are some commonly used **Hadoop Shell commands** for various operations:

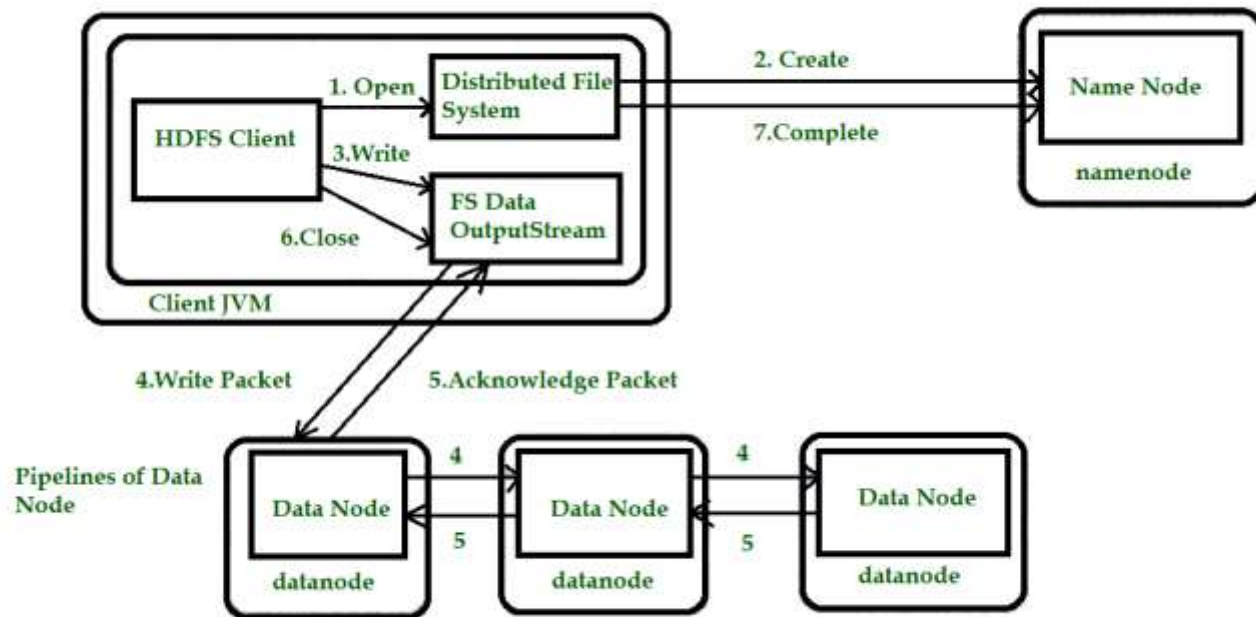
Hadoop Command	Explanation	Example
hadoop fs -ls /	Lists the files and directories in HDFS.	hadoop fs -ls / This lists all files and directories in the root of HDFS.
hadoop fs -mkdir /path	Creates a directory in HDFS.	hadoop fs -mkdir /user/hadoop/newdir Creates a directory named newdir in /user/hadoop/.
hadoop fs -put /local/path /hdfs/path	Uploads a file from the local file system to HDFS.	hadoop fs -put /home/user/data.txt /user/hadoop/ Uploads the data.txt file to /user/hadoop/ in HDFS.
hadoop fs -get /hdfs/path /local/path	Downloads a file from HDFS to the local file system.	hadoop fs -get /user/hadoop/data.txt /home/user/ Downloads the data.txt file from HDFS to the local directory.
hadoop fs -rm /path/to/file	Deletes a file or directory from HDFS.	hadoop fs -rm /user/hadoop/data.txt Deletes the data.txt file from HDFS.

Anatomy of File Write and Read

📁 Anatomy of File Write in HDFS

- **Client** – User ya application jo HDFS ke saath interact karta hai.
- **NameNode** – Master node jo file metadata manage karta hai (jaise file ka naam, blocks kaha hain).
- **DataNode** – Data store karne wale slave nodes.
- **Block** – HDFS files 128MB (by default) ke blocks mein divide hoti hain

Diagram: File Write in HDFS



Step	Explanation (Hinglish)
1. Open	HDFS Client Distributed File System ko open karta hai (jaise new file create karni ho).
2. Create	DFS, NameNode ko request bhejta hai file banane ke liye. NameNode metadata check karta hai (jaise file already exist to nahi karti).
3. Write	FSDDataOutputStream ke through file ka data likhna start hota hai.
4. Write Packet (to DataNodes)	Data blocks write kiye jaate hain pehle DataNode mein, phir second aur third mein (ye process ko kehte hain pipeline writing).
5. Acknowledge Packet	Har DataNode write ke baad ACK (acknowledgement) bhejta hai wapas client ko.
6. Close	Data write hone ke baad FSDDataOutputStream close ho jata hai.
7. Complete	Client NameNode ko batata hai ki file writing complete ho gayi. NameNode metadata update karta hai.

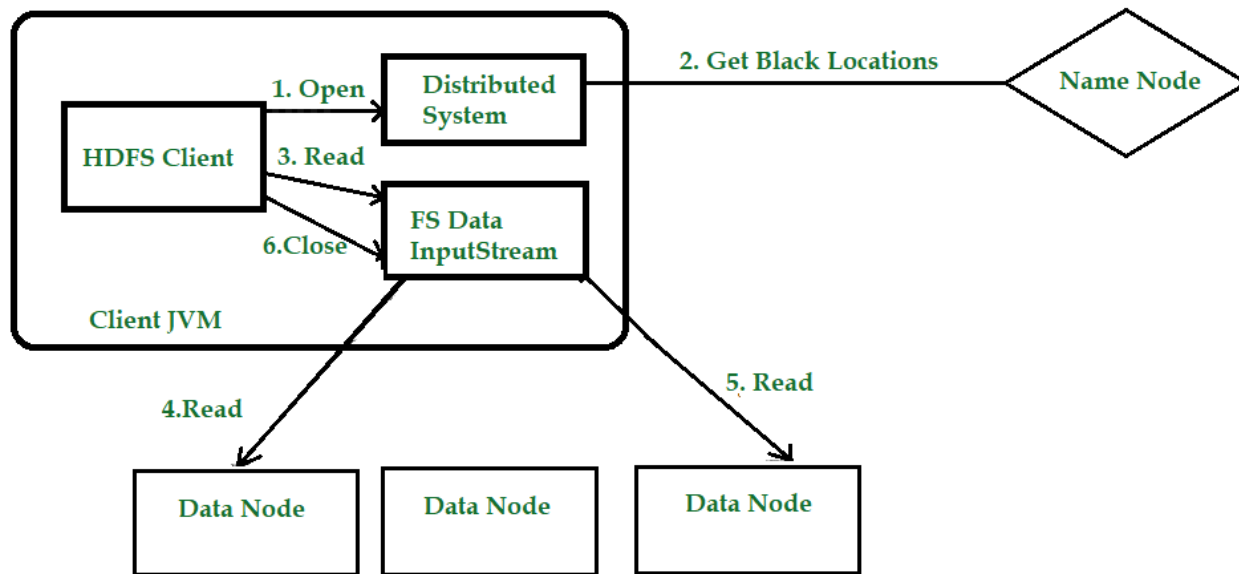
💡 Diagram Key Terms:

- **Client JVM:** Jahaan se client Hadoop system ke saath interact karta hai.
- **FSDataOutputStream:** Java stream jisse data likha jaata hai HDFS mein.
- **DataNode Pipeline:** Teen DataNodes mein ek ke baad ek block write hone ki sequence.
- **ACK Packet:** Confirmation that write successful tha.

★ Summary in Hinglish:

Jab client HDFS mein koi file write karta hai, to sabse pehle NameNode se permission mangi jaati hai. Fir data FSDataOutputStream ke through pipeline mein write hota hai DataNodes ke andar. Har DataNode write hone ke baad confirmation (ACK) bhejta hai. Jab poori file write ho jati hai, to client NameNode ko notify karta hai ki write complete ho gaya.

HDFS File Read Process



Step	Explanation (Hinglish)
1. Open	HDFS Client file open karta hai using the Distributed File System (DFS).
2. Get Block Locations	DFS NameNode se block locations ke baare mein request karta hai. NameNode client ko batata hai ki file ke blocks kaha-kaha stored hain (kis DataNode par).
3. Read	DFS block ke addresses ke saath FSDataInputStream open karta hai taaki data read kiya ja sake.
4. Read (from DataNode 1)	Client block ko first available DataNode se read karta hai.
5. Read (from alternate DataNodes)	Agar pehla DataNode busy ya down ho, to client next DataNode se data read karta hai.
6. Close	Jab file puri read ho jaati hai, FSDataInputStream close ho jata hai.

Term	Meaning
NameNode	Metadata manager – ye batata hai file ke blocks kaha stored hain.
DataNode	Real data ko store karne wale machines.
FSDataInputStream	Stream through which data is read in client program.
Client JVM	Java Virtual Machine jisme client ka program run ho raha hota hai.

Diagram Summary in Simple Words:

- Client pehle file ko open karta hai.
- NameNode se puchta hai: "Mujhe blocks kaha milenge?"
- NameNode bata deta hai: "Ye blocks in DataNodes par hain."
- Client phir un DataNodes se direct data padhta hai.
- Agar ek DataNode fail ho gaya, to dusre DataNode se read hota hai.
- Jab read complete ho jata hai, connection close hota hai.

Feature / Role	NameNode	Secondary NameNode	DataNode
Kya hai?	HDFS ka master node	Checkpointing assistant of NameNode	HDFS ka slave node
Main kaam	Metadata store karta hai (file name, block info, permissions)	FsImage aur EditLogs ka merge karta hai	Real data (file blocks) ko store karta hai
Data store karta hai?	✗ (Sirf metadata)	✗ (Sirf NameNode ke data ka backup)	✓ (Actual data blocks)
Failure ka effect	Poora cluster fail ho sakta hai	Cluster fail nahi hota, but recovery slow ho sakti hai	Sirf us data ka loss jisme wo block stored tha
Communication	Direct client se aur DataNodes se	Sirf NameNode se	Client aur NameNode dono se
Memory requirement	High	Medium	Low
Backup role	Main controller	Temporary backup/Checkpointing manager	None (just storage)
File system image	Maintains FsImage + EditLog live	Periodically new merged FsImage banata hai	✗

Q In Short:

- **NameNode** = Master, Metadata manager.
- **Secondary NameNode** = Backup + checkpoint creator (but not failover).
- **DataNode** = Stores real file data (blocks).

□□ Hadoop MapReduce Paradigm

Q MapReduce kya hota hai?

MapReduce ek programming model hai jo **large data** ko **parallel** process karta hai. Ye do main steps mein kaam karta hai:

1.Map Phase – Data ko todta hai (chunk banaata hai) aur key-value pairs create karta hai.




2.Reduce Phase – Same key wale values ko add ya summarize karta hai.

Word Count Example (Simple Explanation)

 **Goal:**

Har word kitni baar aaya hai, ye count karna.

🔄 Steps: Word Count using MapReduce

 Step	 Phase	 Explanation (Hinglish)
1	Input	Input file read hoti hai line by line.
2	Mapper	Har line ke words ko tod kar (word, 1) pair banata hai.
3	Shuffle/Sort	Same words ko ek group mein le aata hai.
4	Reducer	Sabhi 1s ko add karke word ka total count nikalta hai.
5	Output	Result file banata hai jisme (word, count) hote hain.

Input Files



```
graph TD; A[Input Files] --> B["Mapper<br/>(word, 1) key-value pairs"]; B --> C["Shuffle and Sort by Key"]; C --> D["Reducer<br/>(word, total count)"]; D --> E[HDFS Output];
```

The diagram illustrates the MapReduce workflow. It begins with 'Input Files' in a yellow box, which leads to a 'Mapper' in a light blue box. The Mapper outputs '(word, 1) key-value pairs'. This is followed by a 'Shuffle and Sort by Key' step in a light green box. The data then moves to a 'Reducer' in a light red box, which outputs '(word, total count)'. Finally, the result is stored in 'HDFS Output' in a yellow box. Arrows indicate the sequential flow from top to bottom.

Mapper

(word, 1) key-value pairs

**Shuffle and
Sort by Key**

Reducer

(word, total count)

HDFS Output

Input Example:

Line 1: hadoop is open source

Line 2: hadoop is powerful

□ Map Phase Output:

(hadoop, 1)

(is, 1)

(open, 1)

(source, 1)

(hadoop, 1)

(is, 1)

(powerful, 1)

↯ Shuffle & Sort Phase:

(hadoop, [1,1])

(is, [1,1])

(open, [1])

(source, [1])

(powerful, [1])

+ Reduce Phase Output:

(hadoop, 2)

(is, 2)

(open, 1)

(source, 1)

(powerful, 1)

MapReduce Tasks List

No.	Task Name	Explanation (Hinglish)
1	Input Splitting	Input file ko chhote-chhote splits (parts) me divide kiya jaata hai.
2	Mapping	Har input split par mapper kaam karta hai aur (key, value) pairs banata hai.
3	Shuffling	Mapper se nikle data ko sort aur group kiya jaata hai key ke according.
4	Sorting	Same keys ke data ko ek saath laane ke liye sort kiya jaata hai.
5	Reducing	Har key ke liye values ko summarize ya aggregate kiya jaata hai.
6	Output Writing	Final result ko HDFS (ya kisi aur output format) me store kiya jaata hai.

Real Life Analogy:

Socho ek school me students ke test scores hai:

- **Map:** Har student ka naam aur score nikala gaya → (name, score)
- **Shuffle/Sort:** Sabhi same students ke scores ek saath kiye gaye
- **Reduce:** Har student ke total score ya average nikala gaya
- **Output:** Final result ban gaya (like result sheet)

Job, Task trackers

Hadoop Architecture mein Job Tracker aur Task Tracker ka Role:

1.Job Tracker:

1. Ye **master node** hota hai.
2. Client se job **receive karta hai**.
3. Job ko **multiple tasks** mein divide karke alag-alag Task Trackers ko assign karta hai.
4. Task ki **progress monitor karta hai**, agar koi task fail ho jaye to usse dobara run karwata hai.

2.Task Tracker:

1. Ye **worker node** hota hai.
2. Job Tracker se tasks **receive karta hai**.
3. Apne local system par **Map ya Reduce task execute** karta hai.
4. Regularly **heartbeat bhejta hai** Job Tracker ko, apna status batane ke liye.

Dono milke Hadoop cluster mein **distributed data processing** manage karte hain.

□ 3. Cluster Setup in Hadoop

Hadoop cluster ka matlab hai **multiple machines** ko connect karna (1 master + multiple slaves) jahan distributed processing hoti hai.

Step	Description
1.	Install Java – Hadoop ke liye Java zaroori hai
2.	Install Hadoop – Hadoop binaries download aur configure karo
3.	Create hadoop user – Ek dedicated user create karo Hadoop ke liye
4.	Setup SSH – Master node se slaves ke beech passwordless SSH hona chahiye
5.	Configure XML Files – core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml ko edit karo
6.	Format NameNode – hdfs namenode -format
7.	Start Hadoop Services – Start DFS and YARN: start-dfs.sh and start-yarn.sh

🔑 4. SSH Setup (Passwordless Communication)

Kyun zaroori hai?

Master node ko slave nodes ke saath **password ke bina** connect karna hota hai taaki automatic job assign ho sake.

Master Node pe run karein:

```
ssh-keygen -t rsa
```

```
ssh-copy-id user@slave1
```

```
ssh-copy-id user@slave2
```

Ab master node bina password ke slave pe connect kar sakta hai.

⚙️ 5. Important Hadoop Configuration Files

File Name	Use (Hinglish Me)
core-site.xml	HDFS ka base URI define karta hai
hdfs-site.xml	Replication factor, namenode/data directory define karta hai
mapred-site.xml	MapReduce engine configuration (JobTracker location)
yarn-site.xml	ResourceManager aur NodeManager config define karta hai
slaves	Is file me slave machines ke hostnames diye jaate hain

★ **Summary (Ek Line Me):**

Hadoop Cluster setup me **JobTracker** aur **TaskTracker** distributed job processing ke liye kaam karte hain, jahan **SSH** aur **config files** ki madad se master-slave nodes communicate karte hain.

✦ Introduction to NoSQL:

- ◆ **NoSQL** ka full form hai "**Not Only SQL**".
- ◆ Ye databases specially **Big Data, unstructured ya semi-structured data** ko handle karne ke liye banaye gaye hain.
- ◆ NoSQL flexible schema allow karta hai, isme data ka structure fix nahi hota jaise SQL me hota hai.
- ◆ Use hota hai: **Real-time analytics, social media, IoT apps, content management**, etc.

12 34 Types of NoSQL Databases (with examples):

Type	Description (Hinglish)	Example
1. Key-Value Store	Data ko key aur uske value ke form me store karta hai.Fast lookup ke liye best.	Redis, Riak
2. Document Store	JSON, BSON ya XML format me documents store karta hai.Schema flexible hota hai.	MongoDB, CouchDB
3. Column-based Store	Rows ki jagah columns me data store hota hai.Analytics aur large-scale reads ke liye.	Apache Cassandra, HBase
4. Graph Database	Nodes aur edges ka use karke data ke relationships store karta hai.Best for social networks.	Neo4j, Amazon Neptune

vs SQL vs NoSQL Comparison:

Feature	SQL (Relational DB)	NoSQL (Non-Relational DB)
Structure	Fixed schema (tables)	Flexible schema (documents, key-value)
Scaling	Vertical (upgrade server)	Horizontal (add more servers)
Language	SQL	Varies (MongoQL, CQL, etc.)
Relationships	Strong (joins)	Weak or none
Use Case	Structured data, ERP, Banking	Big Data, IoT, Social apps
Example	MySQL, PostgreSQL, Oracle	MongoDB, Cassandra, Redis

? Why use NoSQL for Big Data applications?

Reason

 High Volume of Data

 Speed & Performance

 Schema Flexibility

 Horizontal Scalability

Explanation (Hinglish)

NoSQL easily handle karta hai huge data sizes

Fast reads/writes for real-time processing

Changing structure frequently? NoSQL me easy hai

Easily

Feature / Type	🗄️ Key-Value Store	📄 Document Store	📊 Column-Family Store	🕸️ Graph Database
Data Format	Key → Value	JSON / BSON / XML	Rows with dynamic columns	Nodes & Edges
Structure	Simple & schema-less	Semi-structured	Table-like (but column-wise)	Graph-based structure
Use Case	Caching, session mgmt	Content mgmt, user profiles	Analytics, time-series data	Social networks, relationships
Scalability	High (horizontal)	High	Very High	Medium to High
Querying	By key only	By fields inside documents	By column families	By traversing relationships
Performance	Very fast for lookup	Good for read/write	Optimized for heavy read/write	Best for relationship queries
Example Databases	Redis, Riak, DynamoDB	MongoDB, CouchDB	Apache Cassandra, HBase	Neo4j, Amazon Neptune
Schema Flexibility	No fixed schema	Flexible schema	Semi-structured	No schema
Ideal For	Simple data access	E-commerce, CMS	Logs, sensor data	Social graphs, recommendations

Textual ETL Processing

 What is ETL?

Step	Full Form	Kaam (Work)
E	Extract	Data ko alag-alag sources se nikalna
T	Transform	Data ko clean, format, ya modify karna
L	Load	Final data ko data warehouse ya Hadoop me store karna

■ Textual ETL Processing kya hota hai?

Textual ETL ka matlab hai **ETL process** jo **text-based data** pe **apply hota hai** — jaise ki:

- Log files
- Social media posts
- News articles
- Emails
- Customer feedback

□ **Why Textual ETL is Important?**

Text data is **unstructured**, isliye:

- Usko process karna complex hota hai
- Traditional ETL tools text data ko easily handle nahi karte
- Textual ETL natural language processing (NLP) ya custom logic ke saath kiya jata hai

□ Stages in Textual ETL Processing

Stage	Description
🔍 Extract	Textual data extract karte hain from files, APIs, websites, etc.
🔍 Preprocessing	Clean karna (remove stopwords, punctuation, etc.)
🔍 Transform	Useful info nikalna: keywords, sentiment, categories, etc.
🔍 Load	Processed text data ko Hadoop, NoSQL ya Data Warehouse me daalna

Example: Social Media Feedback ETL

Step	Kaam
Extract	Twitter API se tweets collect kiye
Transform	Emojis remove kiye, keywords extract kiye, sentiment analysis lagaya
Load	Final CSV ya MongoDB me save kiya analysis ke liye

□ Tools used in Textual ETL:

Category	Tools
Extraction	Apache Nifi, Flume, custom Python scripts
Transformation	NLTK, spaCy, TextBlob, Apache Spark (with PySpark)
Loading	Hadoop HDFS, MongoDB, Elasticsearch, Hive

What is Hadoop Distributed System?

Hadoop ek **open-source distributed computing framework** hai jo large datasets ko **multiple machines** par store aur process karta hai. Iska main part hai **HDFS (Hadoop Distributed File System)** jisme data ko blocks mein tod kar alag-alag nodes pe store kiya jata hai.

Is system mein:

- Data **distributed** hota hai multiple nodes pe.
- **Parallel processing** hoti hai, jisse speed badhti hai.
- Agar ek node fail ho jaye, to data doosri node se mil jata hai (**fault tolerance**).

Advantage of Heartbeat Message in Hadoop:

Heartbeat message ek regular signal hota hai jo Task Tracker (ya DataNode) master node (Job Tracker / NameNode) ko bhejta hai.

Fayde:

- 1.Node alive hai ya nahi yeh check hota hai.**
- 2.Failure detection** fast hota hai. Agar heartbeat missing ho, to master us node ko dead maan leta hai.
- 3.Resource management** mein help karta hai – master decide karta hai kis node ko kaun sa task dena hai.
- 4.Load balancing** improve hota hai – master latest status ke basis par kaam assign karta hai.
- 5.System reliability** badhti hai kyunki system apne nodes ka health track karta rehta hai.