Unit - II Introduction to Hadoop and MapReduce

2.1 Introduction to Hadoop

Hadoop is an open-source framework for storing and processing large-scale datasets across distributed clusters of computers.

Designed to handle **Big Data** efficiently, enabling **scalable, fault-tolerant, and cost-effective** data processing.

Why Do We Need Hadoop?

Traditional systems (RDBMS) faced two major issues as data grew into the petabyte range:

- **Storage Cost:** Storing huge amounts of data on specialized enterprise servers is extremely expensive.
- **Processing Speed:** It takes too long for one processor to scan and analyze terabytes of data sequentially.

Hadoop's Solution:

- **Storage:** Break the data into chunks and store it across many cheap computers.
- **Processing:** Send the calculation code *to* the data, rather than bringing data to the code. Process all chunks in parallel.

History

Year	Event	Details
2002	**The Origin (Nutch)**	**Doug Cutting** and **Mike Cafarella** started a project called **Apache Nutch** to build a search engine that could crawl the entire web. They struggled to store the massive amount of data.
2003	**The Google Spark**	Google published a white paper on **GFS (Google File System)**. Cutting realized this was the solution to their storage problem.

2004	**MapReduce Paper**	Google published another paper on **MapReduce**. This solved the processing problem. Cutting and Cafarella rewrote Nutch to use these Google concepts.
2006	**Hadoop is Born**	Yahoo! hired Doug Cutting. He split the storage/processing part out of Nutch and named it **Hadoop** (after his son's yellow toy elephant).
2008	**Apache Top Level**	Yahoo! released Hadoop as an open-source project to the Apache Software Foundation. Facebook, LinkedIn, and Twitter began using it.
2013	**Hadoop 2.0 (YARN)**	Major upgrade. Introduction of **YARN**, allowing Hadoop to run more than just MapReduce applications.

Applications of Hadoop

- **Fraud Detection (Finance):** Banks scan millions of credit card transactions per second. Hadoop compares live transactions against historical behavior patterns to flag anomalies.
- **Recommendation Engines (Retail/Streaming):** Analyzing user clicks, watch history, and purchase logs to suggest Products you might like.
- **Log Analysis (IT & Security):** Server logs generate terabytes of text daily. Hadoop parses these logs to find security breaches, system failures, or user behavior trends.
- **Data Warehousing (ETL):** Companies use Hadoop as a Data Lake to store raw data cheaply before processing and moving it to a more expensive data warehouse like Snowflake or Teradata.
- **Sentiment Analysis (Social Media):** Scanning millions of tweets or reviews to determine how the public feels about a brand (Positive vs. Negative sentiment).

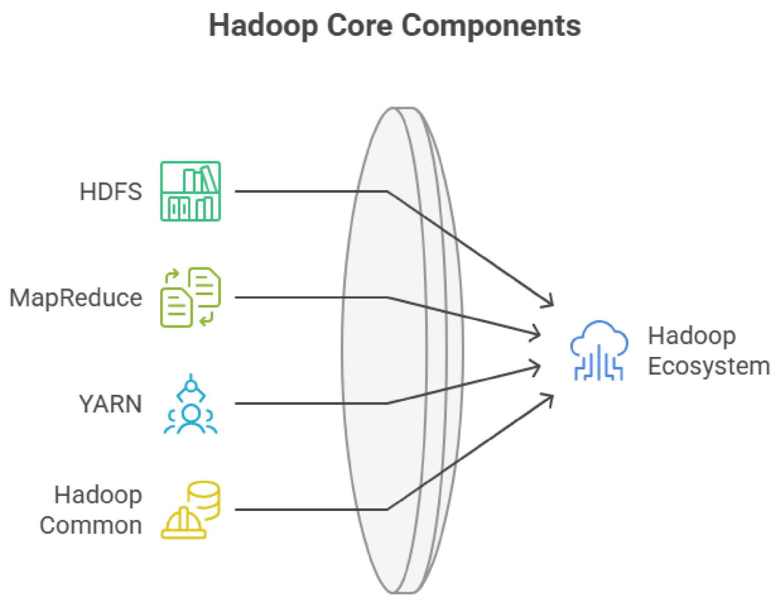
Real-World Examples

- **Facebook:** For Messages & Analytics. Facebook has one of the largest Hadoop clusters in the world. They use it to store photos, messages, and analyze user interactions to serve targeted ads.
- **Netflix:** For Content Recommendation. Netflix uses Hadoop (and its ecosystem) to analyze what you pause, rewind, or skip. This data feeds the algorithm that decides which thumbnail to show you to increase the likelihood of you clicking.
- **Uber:** For Fraud & Trip Analytics. ber uses Hadoop to store geospatial data (GPS points). They analyze this to detect fake rides (fraud) and to optimize pricing during surge hours.
- **British Airways:** For Predictive Maintenance. They analyze sensor data from aircraft engines. If an engine shows vibration patterns similar to a known failure mode, they replace the part *before* it breaks.

2.2 Hadoop and its Ecosystem : Hadoop Core Components, Features of Hadoop, Hadoop Ecosystem Components

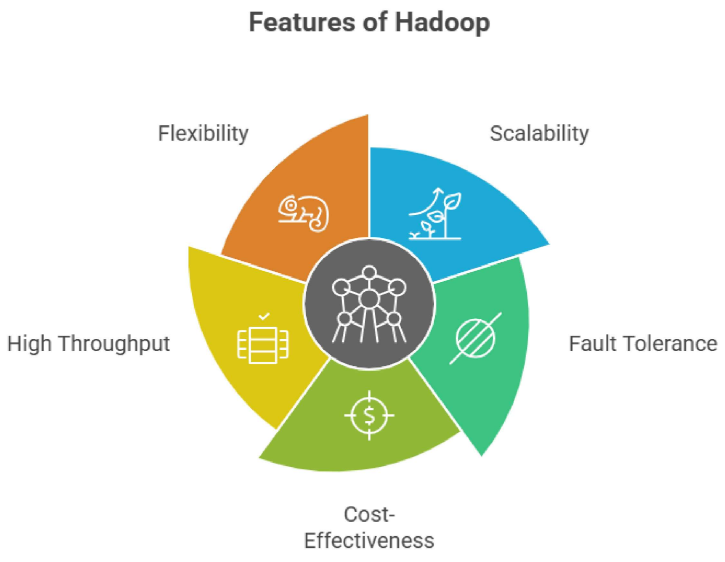
1. Hadoop Core Components

- **HDFS (Hadoop Distributed File System):** Stores large datasets across multiple nodes.
- **MapReduce:** Processes data in parallel across the cluster.
- **YARN (Yet Another Resource Negotiator):** Manages resources and job scheduling.
- **Hadoop Common:** Provides utilities, libraries, and APIs for Hadoop modules.



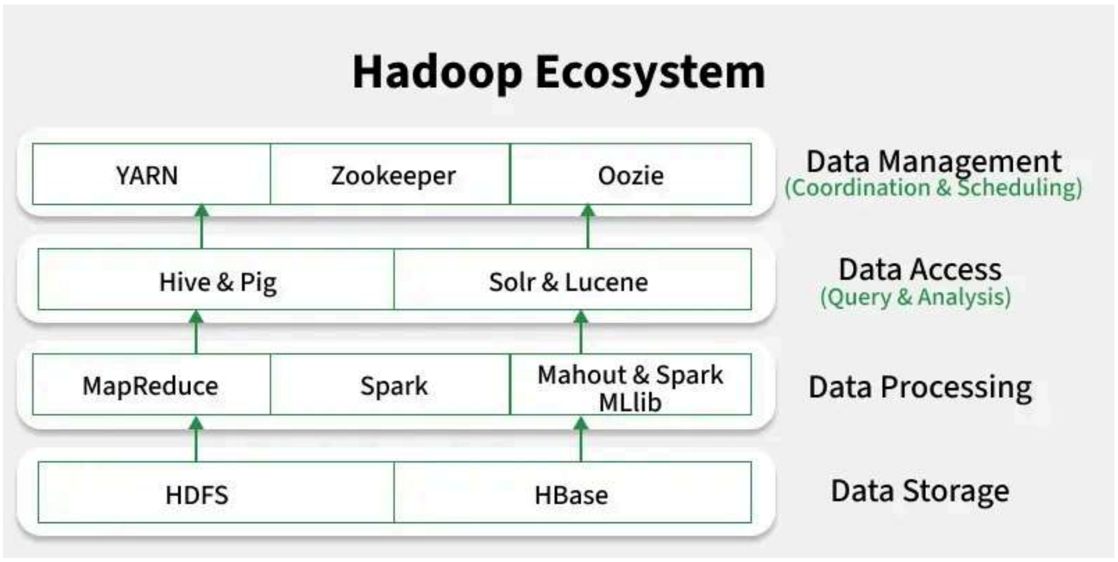
2. Features of Hadoop (Importance/Advantage)

- **Scalability:** Easily add more nodes to handle data growth.
- **Fault Tolerance:** Automatically replicates data across nodes.
- **Cost-Effective:** Runs on commodity hardware.
- **High Throughput:** Processes large datasets efficiently.
- **Flexibility:** Handles structured, semi-structured, and unstructured data.



3. Hadoop Ecosystem Components

- **HDFS (Hadoop Distributed File System):** Stores large datasets across distributed nodes.
- **YARN (Yet Another Resource Negotiator):** Manages cluster resources and job scheduling.
- **MapReduce:** Programming model for batch data processing.
- **Spark:** Fast, in-memory data processing engine.
- **Hive & Pig:** High-level tools for querying and analyzing large datasets.
- **HBase:** NoSQL database for real-time read/write access.
- **Mahout & Spark MLlib:** Libraries for scalable machine learning.
- **Solr & Lucene:** Tools for full-text search and indexing.
- **Zookeeper:** Manages coordination and configuration across the cluster.
- **Oozie:** Workflow scheduler for managing Hadoop jobs.

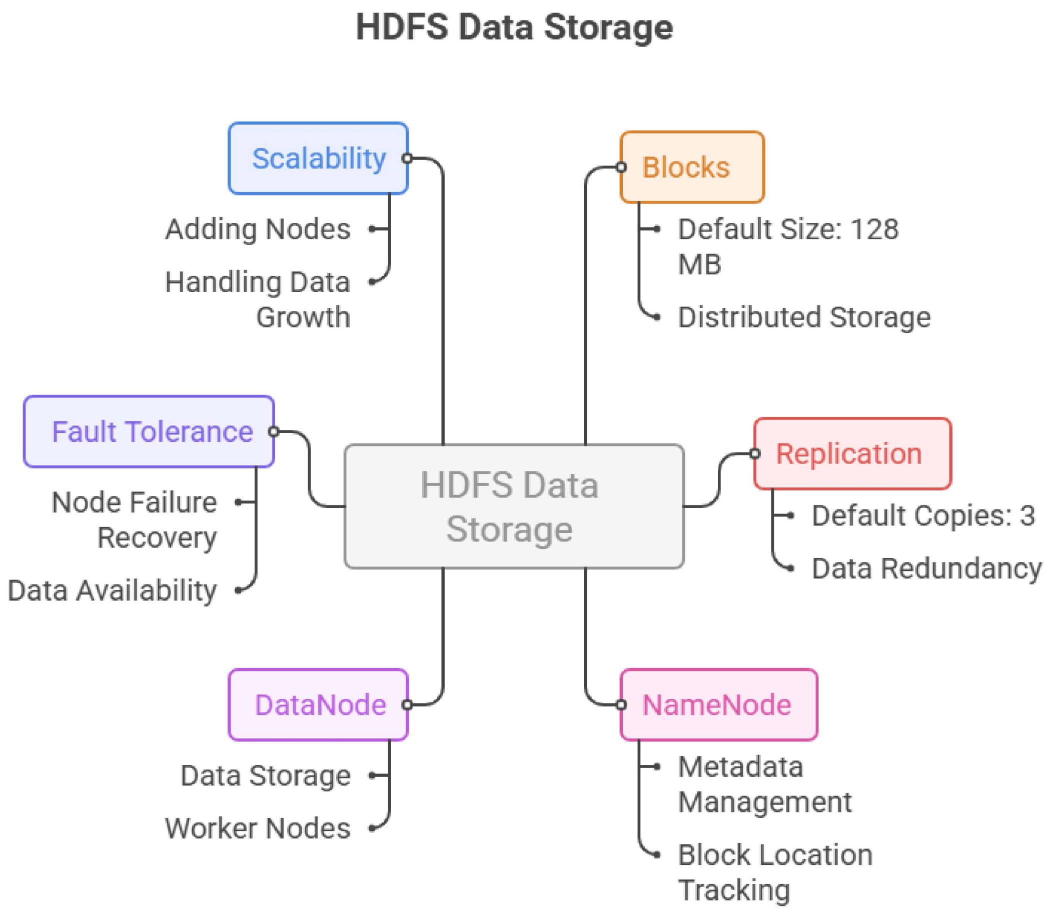


2.3 Hadoop Distributed File System : HDFS data storage, HDFS Commands for interacting with files in HDFS

HDFS (Hadoop Distributed File System) is a unique design that provides storage for extremely large files with streaming data access pattern, and it runs on commodity hardware.

HDFS data storage

- **Blocks:** Files are split into large blocks (default 128 MB) and stored across cluster nodes.
- **Replication:** Each block is replicated (default 3 copies) for fault tolerance.
- **NameNode:** Master node that manages metadata (file names, block locations).
- **DataNode:** Worker nodes that store actual data blocks.
- **Fault Tolerance:** If a node fails, data is still available from other replicas.
- **Scalability:** Can easily add more nodes to store growing data.



HDFS Commands for interacting with files in HDFS

Command	Description
hdfs dfs -ls /path	List files in a directory
hdfs dfs -mkdir /path	Create a directory
hdfs dfs -put localfile /path	Upload file to HDFS
hdfs dfs -get /path localfile	Download file from HDFS
hdfs dfs -cat /path/file	Display file contents
hdfs dfs -rm /path/file	Delete a file
hdfs dfs -rmdir /path	Delete a directory
hdfs dfs -copyFromLocal localfile /path	Copy local file to HDFS
hdfs dfs -copyToLocal /path/file localfile	Copy HDFS file to local system

HDFC NameNode

The **NameNode** is the **Master server** of the HDFS. Manages the file system namespace and controls access to files.

E.g.: **Librarian**:

1. **Stores Metadata:** It keeps the directory tree (file names, permissions) in RAM.
2. **Maps Blocks:** It knows exactly which DataNodes (workers) hold which chunks of data.
3. **Monitors Health:** It receives Heartbeats from DataNodes to ensure they are alive. If one dies, the NameNode orders replicas to be made elsewhere.

It does **not** store the actual file data, only the information about where the data is.

2.4 MapReduce Framework and Programming Model : Hadoop MapReduce Framework, MapReduce Programming Model

1. Hadoop MapReduce Framework

Hadoop MapReduce is a **distributed programming framework** for processing large datasets in parallel across a Hadoop cluster.

Components:

- **JobTracker:** Manages and schedules MapReduce jobs (in older Hadoop versions).
- **TaskTracker:** Executes tasks assigned by JobTracker.
- **YARN:** Modern Hadoop uses YARN for resource management and job scheduling.

Features:

- Scalable and fault-tolerant.
- Processes data close to where it is stored (HDFS).
- Handles large datasets efficiently with parallel processing.

2. MapReduce Programming Model

Breaks tasks into two main functions:

- **Map:** Processes input data and converts it into intermediate key-value pairs.
- **Reduce:** Aggregates intermediate results to produce the final output.

Flow:

- Input data → Map function → Shuffle & Sort → Reduce function → Output.

2.5 Hadoop Yarn : Hadoop 2 Execution Model

YARN is the **resource management layer** of Hadoop 2.x.

Separates **resource management** from **job scheduling/processing**, improving cluster utilization and scalability.

Hadoop 2 Execution Model

Components:

- **ResourceManager (RM):** Master daemon that manages cluster resources.
- **NodeManager (NM):** Runs on each node and manages resources/containers.
- **ApplicationMaster (AM):** Manages execution of a single application/job.

Execution Flow:

- Client submits a job → RM allocates resources → AM negotiates with NM → Tasks run in containers.

Advantages over Hadoop 1:

- Supports **multiple processing frameworks** (MapReduce, Spark, Tez, etc.).
- Better **scalability and cluster utilization**.
- Handles **dynamic resource allocation** efficiently.

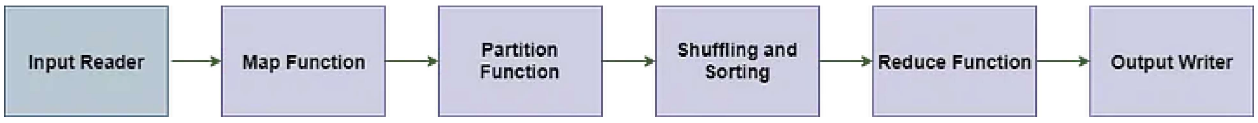
2.6 MapReduce : Map Tasks, Key-Value Pair, Grouping by Key, Partitioning, Combiners, Reduce Tasks, Details of MapReduce Processing Steps

1. **Map Tasks:** Process input data and generate **key-value pairs**.
2. **Key-Value Pair:** Fundamental data unit; key is used for grouping, value carries the actual data.
3. **Grouping by Key:** All values with the same key are grouped together for processing.
4. **Partitioning:** Divides key-value pairs among reducers to ensure balanced load.

5. **Combiners:** Optional mini-reducers applied after map tasks to reduce data transfer.
6. **Reduce Tasks:** Aggregate and process grouped key-value pairs to produce final output.

Details of MapReduce Processing Steps

1. **Input Splitting:** Large data split into chunks, each processed by a map task.
2. **Mapping:** Map function processes input and emits key-value pairs.
3. **Partition function:** The partition function assigns the output of each Map function to the appropriate reducer.
4. **Shuffling & Sorting:** Key-value pairs are **grouped by key** and sorted for reducers.
5. **Reducing:** Reduce function processes grouped data to generate final output.
6. **Output Writing:** Final results written to HDFS.



Addressing challenges of processing and analyzing large scale data:

Hadoop solves Big Data challenges using a **Divide and Conquer** approach:

1. **Storage (Volume):** It breaks huge files into small blocks and distributes them across thousands of cheap computers (**HDFS**), so you never run out of space.
2. **Speed (Velocity):** It processes data in parallel on all computers at once (**MapReduce**) and moves the "code to the data" (**Data Locality**) to save time.

3. **Variety:** It allows you to save any file type (Text, Video, JSON) immediately without setting up tables first (**Schema-on-Read**).
4. **Reliability:** It keeps **3 copies** of every file block. If one computer crashes, the data is safe on another (**Fault Tolerance**).
5. **Cost:** It runs on standard, inexpensive hardware (**Commodity Hardware**) instead of costly supercomputers.

Questions:

1. State any two features of Hadoop
2. State the importance of Hadoop
3. Explain the role of Name Node in HDFS.
4. Explain key advantages of Hadoop, List advantages of Hadoop.
5. Describe Hadoop distributed file system, Describe HDFS in detail, Describe HDFS
6. Define Hadoop.
7. List the features of Hadoop.
8. Explain Hadoop in detail.
9. How Hadoop addresses the challenges of processing and analyzing large scale data ?
10. How does Mapreduce work in Hadoop ? What role does it play in distributed data processing ?
11. Give the key advantages of Hadoop. Also compare RDBMS & Hadoop.