HDFS (Hadoop Distributed File System)

HDFS is a distributed, fault-tolerant file system designed to store and process very large datasets across clusters of commodity hardware.

Architecture

- **NameNode (Master):** Maintains filesystem metadata—directory structure, file permissions, and block locations.
- **DataNodes (Workers):** Store actual data blocks and serve read/write requests.

Core Concepts

- **Distributed Storage:** Files are split into large blocks and distributed across multiple machines.
- **Fault Tolerance:** Each block is replicated across different nodes to ensure data availability during failures.
- **Write:** Client to Name Node (metadata) to Data Nodes (pipeline write with replication).
- **Read:** Client Name Node (block locations) nearest Data Node (data stream).

MapReduce:

MapReduce breaks data processing into two primary phases:

Map Phase

- Input data is split into chunks.
- The *Mapper* processes each record and outputs intermediate **key–value pairs**.

Reduce Phase

- Intermediate data is **shuffled and sorted** by key.
- The *Reducer* aggregates values for each key to produce final output.

Architecture Components

- **Client:** Submits the job**.**
- **YARN Resource Manager:** Allocates cluster resources.
- **Node Manager:** Manages resources on each node.
- **Application Master:** Coordinates task execution.
- **Mapper / Reducer Tasks:** Perform the actual computation.

Execution Flow

1. **Input data stored in HDFS**
2. **Input splits created**
3. **Mapper tasks process splits**
4. **Shuffle & Sort phase (automatic)**
5. **Reducer tasks aggregate results**

Output written back to HDFS

Example (Word Count Logic)

- **Map:**
  Input: "big data big analytics"
  Output:
  (big,1), (data,1), (big,1), (analytics,1)

- **Reduce:**
  Input: (big,[1,1])
  Output: (big,2)

Key Features:

- **Parallel Processing:** Massive scalability across nodes
- **Fault Tolerant:** Failed tasks are automatically retried

Apache Spark

- Apache Spark is a distributed data processing engine designed for fast, scalable, in-memory analytics. It is widely used for big data processing, real-time analytics, and machine learning.

Core Concepts

- **In-Memory Processing**
  - Data is cached in memory (RAM) instead of disk.
  - Enables **10–100x faster** processing compared to disk-based MapReduce.
- **Unified Analytics Engine**
  Spark supports multiple workloads on the same engine:
  - **Batch processing**
  - **Streaming**
  - **SQL & BI**
  - **Machine Learning**
- **Spark Architecture**
  - **Driver**
    - Takes Input
    - Orchestrates the application
    - Coordinates tasks
  - **Cluster Manager:** Holds Meta data
    - YARN, Kubernetes
    - Allocates resources
  - **Executors**
    - Run tasks on worker nodes
    - Store data in memory and disk

Dataframe:

It is a distributed, immutable, schema-based table (rows and named columns) designed for high-performance data processing.

- Data is partitioned across a cluster.
- Columns have names and data types.

RDD (Resilient Distributed Dataset):

An RDD is Spark's foundational, immutable distributed data structure that provides fault tolerance through lineage and supports parallel computation via transformations and actions.

Data Governance:
Data Masking and **Encryption** are foundational controls within data governance used to protect sensitive information while enabling compliant data usage across analytics, engineering, and AI workloads.