What is Hadoop?

Hadoop is an **open-source framework** designed for **storing and processing large volumes of data** in a **distributed computing environment**. It was developed by the **Apache Software Foundation (ASF)** and is widely used for **Big Data processing**. Hadoop enables businesses to handle vast amounts of structured, semi-structured, and unstructured data efficiently.

Key Features of Hadoop:

1. **Distributed Storage:** Data is stored across multiple nodes in a cluster, ensuring scalability and fault tolerance.
2. **Parallel Processing:** It processes data using parallel computing, improving efficiency and speed.
3. **Fault Tolerance:** If a node fails, Hadoop automatically recovers and continues processing using replicated data.
4. **Scalability:** New nodes can be added to the cluster without disrupting the system.
5. **Cost-Effective:** It runs on commodity hardware, reducing infrastructure costs compared to traditional databases.

Hadoop Ecosystem Components

Hadoop consists of **four core modules**, each responsible for specific functions:

1. Hadoop Distributed File System (HDFS)

- A highly **fault-tolerant distributed file system** that stores large datasets across multiple machines.
- **Data is split into blocks** (default size: 128MB or 256MB) and distributed across different nodes.
- Each block is **replicated (default: 3 copies)** for fault tolerance.
- Works on a **Master-Slave Architecture**, where:
    - **NameNode** (Master) – Manages metadata and file structure.
    - **DataNode** (Slave) – Stores the actual data blocks.

2. MapReduce

- A **programming model** for processing large datasets in parallel across multiple nodes.
- It works in two stages:
    1. **Map Phase** – Splits data into key-value pairs and processes them in parallel.
    2. **Reduce Phase** – Aggregates and processes the mapped data to produce the final result.

- Used for tasks like **log processing, text analysis, and ETL (Extract, Transform, Load) operations**.

3. Yet Another Resource Negotiator (YARN)

- **Manages and schedules computing resources** in a Hadoop cluster.
- Divides responsibilities into:
  - **ResourceManager** – Allocates resources and manages task execution.
  - **NodeManager** – Monitors resource usage on each node.
- Improves **scalability and efficiency** by optimizing resource allocation.

4. Hadoop Common

- Provides **libraries, utilities, and Java-based APIs** for interacting with Hadoop.
- Ensures smooth communication between all Hadoop components.

Hadoop Ecosystem and Tools

Hadoop integrates with several tools to extend its capabilities:

1. **Apache Hive** – SQL-like querying for structured data stored in HDFS.
2. **Apache Pig** – A high-level scripting language for data transformation.
3. **Apache HBase** – A NoSQL database that supports real-time read/write access.
4. **Apache Spark** – A fast, in-memory data processing engine that can replace MapReduce.
5. **Apache Flume & Apache Kafka** – Used for real-time data ingestion from multiple sources.
6. **Apache Sqoop** – Transfers data between Hadoop and relational databases (MySQL, PostgreSQL).

Advantages of Hadoop

1. **Handles Big Data** – Designed to process terabytes or petabytes of data efficiently.
2. **Highly Scalable** – Easily scales horizontally by adding more nodes.
3. **Low Cost** – Uses inexpensive commodity hardware.
4. **Fault Tolerance** – Replicates data across multiple nodes to prevent data loss.
5. **Supports Various Data Types** – Works with structured, semi-structured, and unstructured data.
6. **Open Source** – No licensing fees, making it accessible for businesses and researchers.

2. Difference Between Hadoop and Database

Hadoop and traditional databases serve different purposes. While **Hadoop** is designed for **Big Data processing and distributed storage**, a **database** (such as MySQL, PostgreSQL, or Oracle) is optimized for **structured data storage and retrieval** using SQL queries.

Feature	Hadoop	Database (RDBMS - Relational Database Management System)
Purpose	Stores and processes massive amounts of unstructured, semi-structured, and structured data.	Stores and manages structured data in tables with predefined schemas.
Data Type	Works with structured, semi-structured, and unstructured data.	Works primarily with structured data (rows and columns).
Data Processing	Batch processing (MapReduce, Spark) for large-scale data analytics.	Transactional processing (OLTP) for quick data retrieval and updates.
Schema	Schema-on-read (flexible, schema defined at query time).	Schema-on-write (fixed schema, defined before data insertion).
Storage System	Uses **HDFS (Hadoop Distributed File System)** to store large datasets across multiple nodes.	Stores data in **tables** on a single or clustered relational database system.
Scalability	**Highly scalable** – Can scale horizontally by adding more machines (nodes).	**Limited scalability** – Scaling requires adding more resources to a single server (vertical scaling).
Fault Tolerance	**High** – Data is replicated across nodes for fault tolerance.	**Low** – If the database server fails, data may be lost unless backed up.
Processing Speed	Slower for real-time queries but optimized for batch processing.	Faster for small-scale real-time transactions and queries.
Query Language	Uses **MapReduce, Spark, Hive (SQL-like)** for data processing.	Uses **SQL** (Structured Query Language) for querying and data manipulation.
Cost	**Low cost** – Open-source, runs on commodity hardware.	**High cost** – Requires expensive hardware and software licenses for large-scale data.

Use Cases	Big Data analytics, log processing, machine learning, data lakes.	Banking, e-commerce, customer relationship management (CRM), real-time transactions.

When to Use Hadoop vs. Database?

✔ **Use Hadoop when:**

- You need to process **huge volumes of data** (terabytes or petabytes).
- You are working with **semi-structured or unstructured data** (logs, social media, images, videos).
- You need a **cost-effective, distributed, and scalable solution**.
- Batch processing and analytics are more important than real-time transactions.

✔ **Use a Database when:**

- You need **real-time transactions** (e.g., banking, e-commerce, inventory management).
- You have **structured data** and need quick queries using **SQL**.
- Data integrity and ACID compliance (Atomicity, Consistency, Isolation, Durability) are critical.
- You are working with a **moderate amount of data** that fits in a single machine or a small cluster.

3. What is YARN?

YARN (**Yet Another Resource Negotiator**) is a **resource management and job scheduling framework** in the Hadoop ecosystem. It was introduced in **Hadoop 2.0** to **overcome the limitations of MapReduce v1** by **decoupling resource management from job execution**.

Purpose of YARN:

1. **Efficient Resource Management** – Allocates CPU, memory, and disk resources across different applications.
2. **Job Scheduling** – Manages and schedules multiple tasks running simultaneously in a Hadoop cluster.
3. **Supports Multiple Processing Frameworks** – Can run **MapReduce, Spark, Tez, and Flink** on the same Hadoop cluster.

Components of YARN

YARN has a **Master-Slave architecture**, consisting of the following main components:

1. ResourceManager (Master Node)

- The **central authority** that manages resources across the Hadoop cluster.
- Runs on a dedicated master node.
- **Components of ResourceManager:**
    - **Scheduler:** Allocates resources (CPU & memory) to applications based on predefined policies (FIFO, Fair Scheduling, etc.).
    - **Application Manager:** Manages application lifecycle (starting, monitoring, and restarting applications if they fail).

---

2. NodeManager (Slave Nodes)

- Runs on every **DataNode** in the cluster.
- Monitors **resource usage** (CPU, memory, disk) on that node.
- Launches and manages **containers** assigned by the ResourceManager.
- Sends **heartbeat signals** to the ResourceManager to report node health.

---

3. ApplicationMaster

- Manages the **execution of a single application** (MapReduce, Spark, or other workloads).
- Requests resources from the **ResourceManager** and monitors job progress.
- Each application running on YARN has its own **ApplicationMaster**.

---

4. Containers

- **Basic unit of processing in YARN** (like a virtual machine for running tasks).
- Allocated dynamically by the ResourceManager.
- Each container gets specific **CPU cores and memory** to execute tasks.
- Can run **multiple types of applications** (MapReduce, Spark, Tez, etc.).

YARN Workflow (How YARN Works)

1. **User submits an application** (e.g., MapReduce job, Spark job).
2. The **ResourceManager** assigns an **ApplicationMaster** for the job.
3. The **ApplicationMaster** requests resources from the **Scheduler** in ResourceManager.
4. **NodeManagers** launch **containers** to execute tasks.
5. **Containers process the data** in parallel and report back to the **ApplicationMaster**.
6. Once all tasks are complete, **resources are released**, and the job finishes.

4. What is a Hadoop Cluster?

A **Hadoop cluster** is a group of **interconnected computers (nodes)** that work together to **store and process large volumes of data** using the Hadoop framework. It follows a **distributed computing model**, where data is split across multiple nodes, and processing is done in parallel to ensure scalability, fault tolerance, and high performance.

Components of a Hadoop Cluster

A Hadoop cluster has a **Master-Slave architecture**, consisting of different types of nodes:

1. Master Nodes (Manages the Cluster)

Master nodes control and coordinate data storage and processing.

- **NameNode** (Manages Storage) – Manages the **HDFS metadata** (file locations, permissions) and directs **DataNodes** on where to store data.
- **ResourceManager** (Manages Processing) – Allocates resources (CPU, memory) and schedules tasks in **YARN**.
- **JobTracker** (Hadoop 1.x) / **ApplicationMaster** (Hadoop 2.x) – Oversees **job execution and progress tracking**.

2. Slave Nodes (Store and Process Data)

Slave nodes perform the actual data storage and processing.

- **DataNodes** (Storage) – Store the **actual data** in HDFS and replicate it for fault tolerance.
- **NodeManagers** (Processing) – Execute tasks by running **MapReduce, Spark, or other jobs** within YARN containers.

3. Client Node

- The user interacts with the Hadoop cluster via the **Client Node**.
- Used for **submitting jobs, retrieving results, and managing the cluster**.

Types of Hadoop Clusters

Hadoop clusters can be classified based on their deployment model:

1. Single-Node Cluster (Standalone Mode)

- All Hadoop components (**NameNode, DataNode, ResourceManager, NodeManager**) run on a **single machine**.
- Used for **testing and development**, not for production.

2. Multi-Node Cluster (Distributed Mode)

- Hadoop runs on **multiple machines**, with separate **master** and **slave nodes**.
- Used for **large-scale data processing** in production environments.

3. Cloud-Based Hadoop Cluster

- Hadoop is deployed on cloud platforms like **AWS (EMR), Azure HDInsight, or Google Cloud Dataproc**.
- Provides **scalability, flexibility, and managed services** without maintaining physical servers.

How a Hadoop Cluster Works (Workflow)

1. **Data is ingested** into the cluster and stored in **HDFS** across multiple **DataNodes**.
2. **Job submission:** A user submits a job using **MapReduce, Spark, Hive, or Pig**.
3. **Resource allocation: YARN ResourceManager** allocates resources and assigns tasks to nodes.
4. **Parallel Processing: NodeManagers** run tasks in **containers** to process data efficiently.
5. **Job Completion:** The final result is **aggregated and stored** in HDFS or an external system.