



# Hadoop and MapReduce Programming

PRITHVIN K C - (22Z249)  
SANDEEP K - (22Z257)  
VIGNESHWARAN P - (22Z271)  
KRISHANU DEY - (22Z277)  
MUKESH E - ( 22Z278)



# Introduction To Hadoop

Vigneshwaran P - 22z271

# What is Hadoop?



**Hadoop** is an **open-source** framework designed for **storing and processing large datasets** in a **distributed computing environment**.

Developed by **Apache Software Foundation**.

Works on clusters of **commodity hardware** (low-cost, off-the-shelf computers).

## Key Characteristics of Hadoop:

- **Distributed Storage**: Splits data into smaller chunks and stores them across multiple nodes.
- **Parallel Processing**: Processes data in parallel using MapReduce for fast computation.
- **Fault Tolerance**: Automatically replicates data across nodes to prevent data loss.
- **Scalability**: Can easily scale from a few machines to thousands.
- **Cost-Effective**: Uses inexpensive hardware to store and process vast amounts of data.

# Hadoop Architecture



Hadoop follows a **Master-Slave Architecture**, where a **master node** manages the system, and multiple **slave nodes** store and process data.

It has **three core layers**:

**HDFS (Hadoop Distributed File System) – Storage Layer**

**YARN (Yet Another Resource Negotiator) – Resource Management**

**MapReduce – Processing Layer**

# Master-Slave Architecture



Hadoop's architecture consists of two main types of nodes:

- ◆ **Master Node (Manages & Coordinates)**

- Controls data storage and processing.
- Includes **NameNode (HDFS)**, **ResourceManager (YARN)**, and **JobTracker (MapReduce)**.

- ◆ **Slave Nodes (Store & Process Data)**

- Store actual data and perform computations.
- Includes **DataNodes (HDFS)**, **NodeManagers (YARN)**, and **TaskTrackers (MapReduce)**.

# Master Node



The Master Node is responsible for **managing the system, coordinating tasks, and ensuring fault tolerance**. It includes:

- ◆ **NameNode (HDFS - Storage Master)**
  - Stores **metadata** (information about file locations, block distribution).
  - Does **not store actual data**, only the directory structure.
  - Sends **instructions to DataNodes** for read/write operations.
- ◆ **ResourceManager (YARN - Resource Management Master)**
  - Allocates cluster resources and schedules tasks.
  - Monitors NodeManagers and assigns computational jobs.
- ◆ **JobTracker (MapReduce - Processing Master)**
  - Assigns MapReduce tasks to TaskTrackers.

# Slave Nodes (Storage & Processing Layer)



The Slave Nodes handle **data storage and computation**. Each slave node includes:

- ◆ **DataNode (HDFS - Storage Worker)**

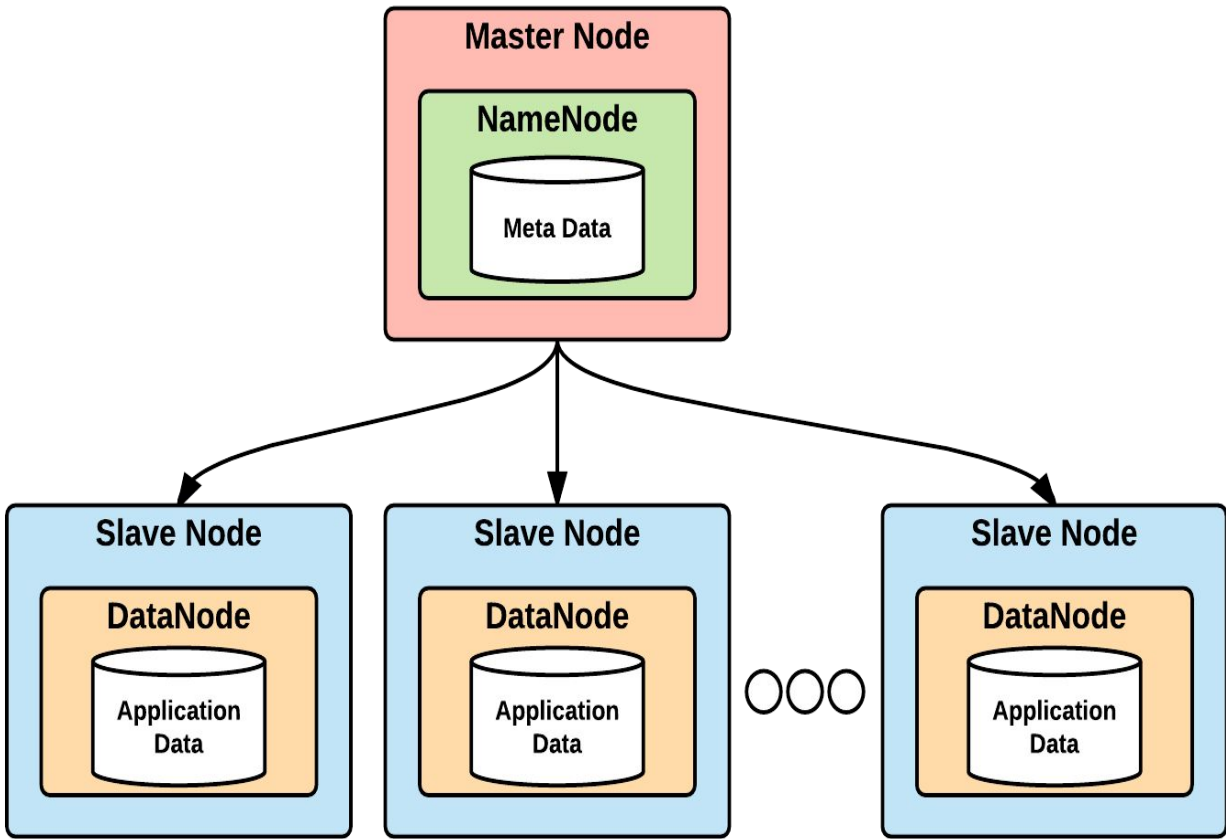
- Stores actual **blocks of data**.
- Responds to read/write requests from the NameNode.
- Sends periodic **heartbeat signals** to the NameNode.

- ◆ **NodeManager (YARN - Resource Management Worker)**

- Executes tasks assigned by the **ResourceManager**.
- Monitors resource usage and reports status.

- ◆ **TaskTracker (MapReduce - Processing Worker)**

- Executes MapReduce tasks assigned by the JobTracker.
- Reports task progress and completion.







# How Hadoop Works (Processing Workflow)

The **workflow** involves three main stages:

**Data Ingestion (Storing in HDFS)**

**Data Processing (Using MapReduce)**

**Result Storage (Final Output in HDFS or External Systems)**



## Step 1: Data Ingestion into HDFS

- ◆ Large files (e.g., logs, CSV, JSON, images) are **split into blocks** (default: **128MB or 256MB**).
- ◆ Blocks are **distributed** and stored across **multiple DataNodes** in the Hadoop cluster.
- ◆ Each block is **replicated (default: 3 copies)** to ensure **fault tolerance**.



## Step 2: Job Execution using MapReduce

- ◆ **User submits a job** to process data stored in HDFS.
- ◆ **YARN allocates resources** and assigns tasks across the cluster.
- ◆ **MapReduce processes the data in two main phases:**
  - **Map Phase:** Data is read, split, and processed in parallel.
  - **Shuffle & Sort:** Intermediate results are grouped together.
  - **Reduce Phase:** Aggregates and finalizes the output.



## Step 3: Storing the Processed Output

- ◆ The processed data is **stored back into HDFS** for further analysis.
- ◆ It can also be exported to **external databases** (SQL, NoSQL) or **visualization tools**.



# **Hadoop Distributed File System**

**Krishanu Dey-22z277**



## Data Storage Mechanism

- Files are divided into blocks.
- Blocks are replicated across DataNodes.
- Replication ensures fault tolerance and availability.



## Read and Write Operations

### Write Process:

- Client splits data into blocks.
- Blocks are replicated and distributed across different DataNodes.
- NameNode manages the metadata.

### Read Process:

- Client requests metadata from NameNode.
- Retrieves data directly from multiple DataNodes.



## Key Features of HDFS

**High Fault Tolerance:** Automatically recovers from **DataNode failures**.

**Scalability:** Handles **petabytes of data** across **thousands of nodes**.

**High Throughput:** Optimized for **batch processing** and large-scale data handling.

**Cost-Effective:** Runs on **commodity hardware**, reducing infrastructure costs.





## Limitations of HDFS

**Not suitable for real-time applications** (high latency for small data requests).

**Inefficient for small files** (optimized for large block-based storage).

**High metadata storage overhead** (NameNode keeps track of all file locations).



## HDFS vs. Traditional File Systems

HDFS	Traditional File Systems
Data Distributed Across Multiple Nodes	Data Stored on a single system
It is Scalable	It is limited hardware
Replication ensures reliability	Data loss possible if storage fails
Parallel data access speeds up processing	Slower due to single storage unit



# MapReduce Programming Model

Mukesh E - 22z278

# What is MapReduce?



**MapReduce** is a parallel, distributed programming model in the Hadoop framework that can be used to access the extensive data stored in the Hadoop Distributed File System (HDFS). The Hadoop is capable of running the MapReduce program written in various languages such as Java, Ruby, and Python.

One of the beneficial factors that MapReduce aids is that MapReduce programs are inherently parallel, making the very large scale easier for data analysis. When the MapReduce programs run in parallel, it speeds up the process.

# Process Of MapReduce Programs



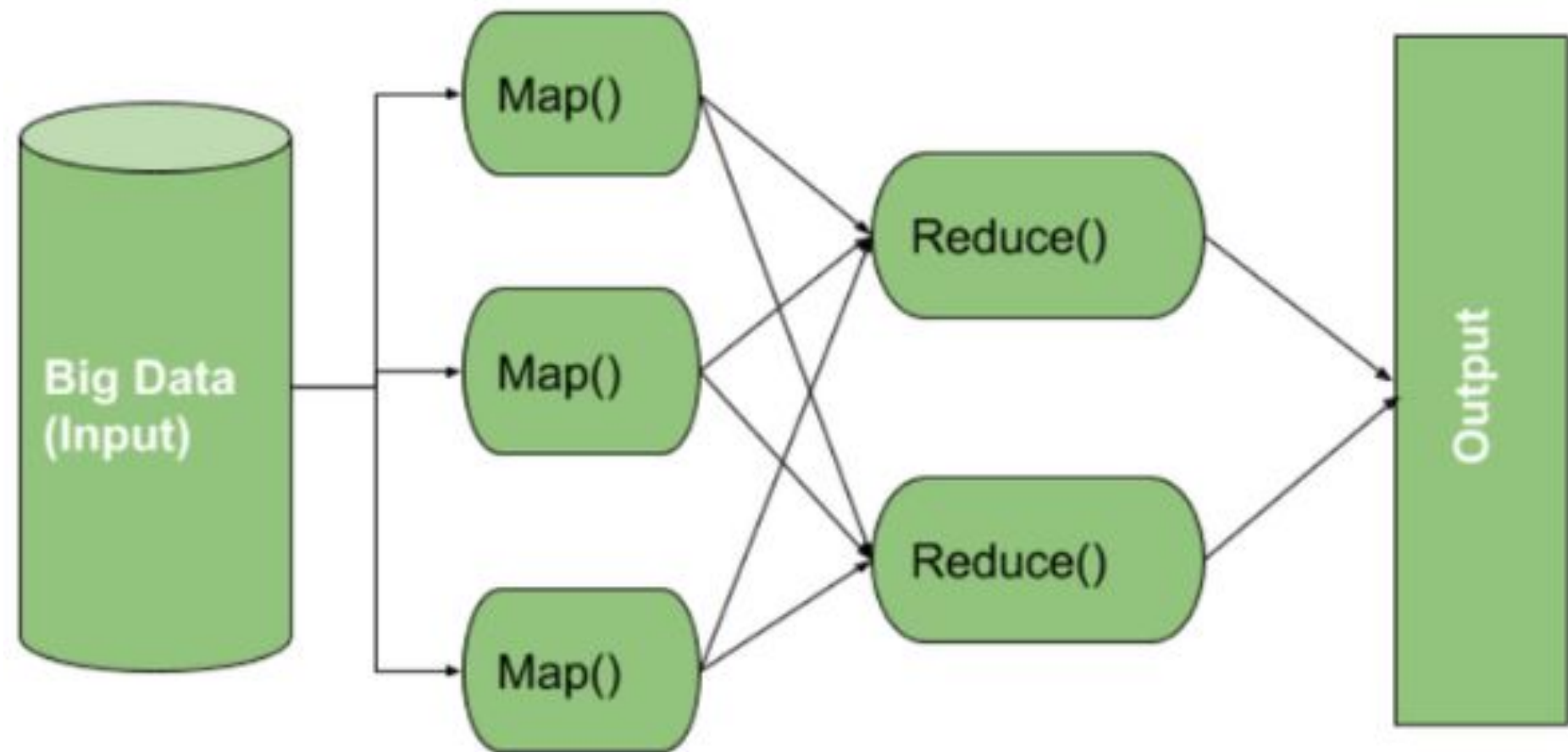
- **Dividing the input into fixed-size chunks:** Initially, it divides the work into equal-sized pieces. When the file size varies, dividing the work into equal-sized pieces isn't the straightforward method to follow, because some processes will finish much earlier than others while some may take a very long run to complete their work.
- **Combining the results:** Combining results from independent processes is a crucial task in MapReduce programming because it may often need additional processing such as aggregating and finalizing the results.

# Key components of MapReduce

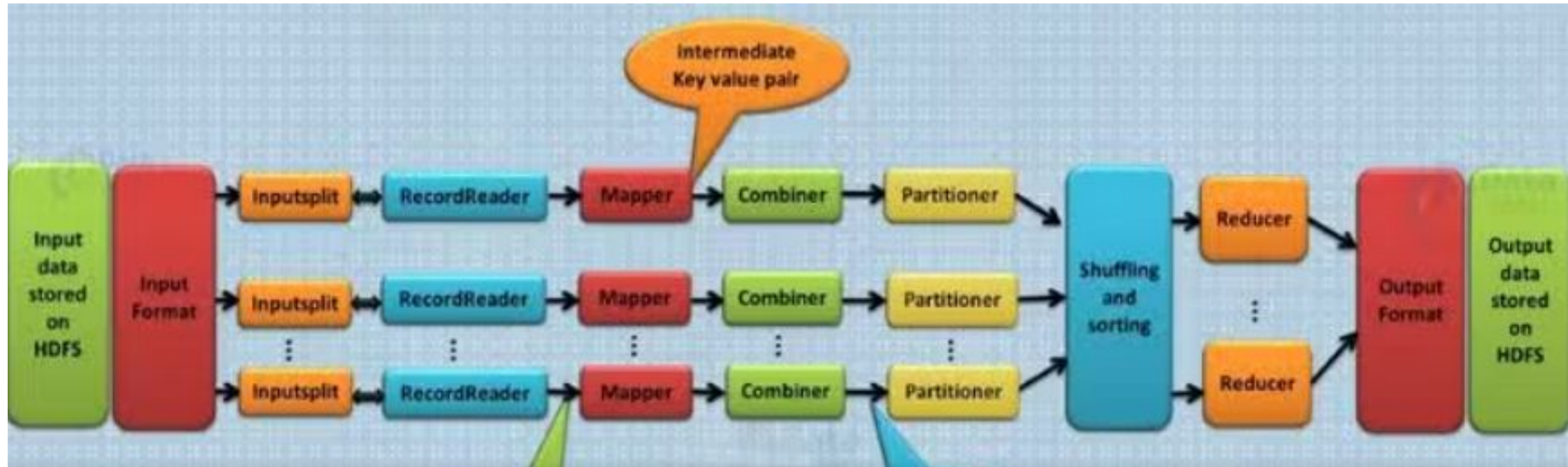


**Mapper:** Mapper is the first phase of the MapReduce. The Mapper is responsible for processing each input record and the key-value pairs are generated by the InputSplit and RecordReader. Where these key-value pairs can be completely different from the input pair. The MapReduce output holds the collection of all these key-value pairs.

**Reducer:** The reducer phase is the second phase of the MapReduce. It is responsible for processing the output of the mapper. Once it completes processing the output of the mapper, the reducer now generates a new set of output that can be stored in HDFS as the final output data.



# Execution workflow of MapReduce



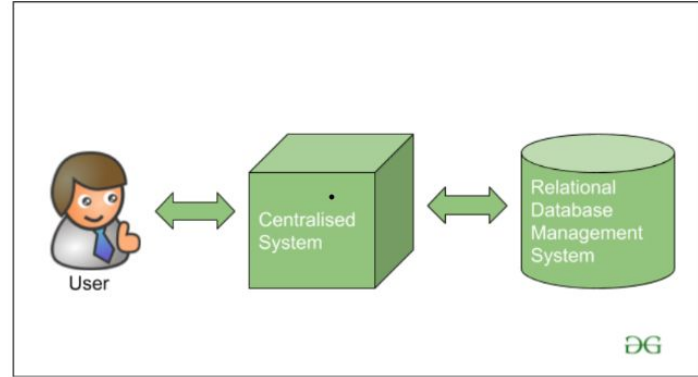




# Hadoop Ecosystem

Prithvin K C-22Z249

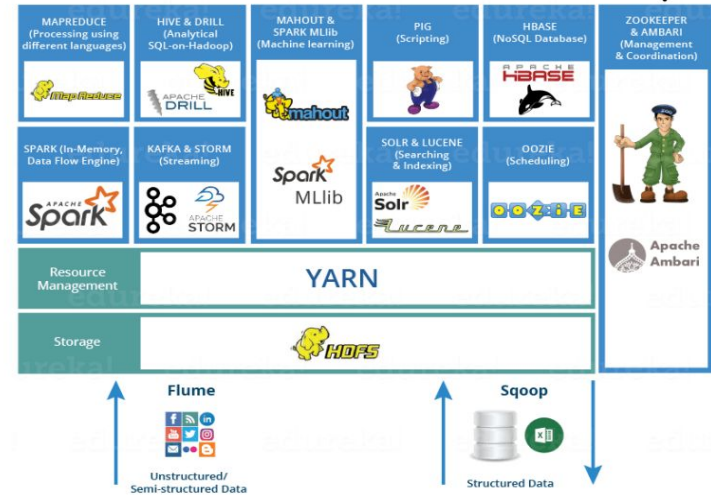
# Hadoop Ecosystem



*Hadoop Ecosystem* is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions. There are *four major elements of Hadoop* i.e. **HDFS, MapReduce, YARN, and Hadoop Common Utilities**. Most of the tools or solutions are used to supplement or support these major elements. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.

Following are the components that collectively form a Hadoop ecosystem:

- **HDFS:** Hadoop Distributed File System
- **YARN:** Yet Another Resource Negotiator
- **MapReduce:** Programming based Data Processing
- **Spark:** In-Memory data processing
- **PIG, HIVE:** Query based processing of data services
- **HBase:** NoSQL Database
- **Mahout, Spark MLlib:** [Machine Learning](#) algorithm libraries
- **Solar, Lucene:** Searching and Indexing
- **Zookeeper:** Managing cluster
- **Oozie:** Job Scheduling



HADOOP ECOSYSTEM

## HDFS:



- HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of two core components i.e.
  1. Name node
  2. Data Node
- Name Node is the prime node which contains metadata (data about data) requiring comparatively fewer resources than the data nodes that stores the actual data. These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective.
- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.

## YARN:



- Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- Consists of three major components i.e.
  1. Resource Manager
  2. Nodes Manager
  3. Application Manager
- Resource manager has the privilege of allocating resources for the applications in a system whereas Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager. Application manager works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.


## MapReduce:

- By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.
- MapReduce makes the use of two functions i.e. Map() and Reduce() whose task is:
  1. **Map()** performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.
  2. **Reduce()**, as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.



**Pig:** Developed by Yahoo, Pig uses **Pig Latin**, an SQL-like scripting language, to process and analyze large datasets. It simplifies programming by handling **MapReduce** execution in the background. Pig structures data flow, automates processing, and stores results in **HDFS**, making it an essential part of Hadoop.

**Hive:** A **data warehouse system** that enables SQL-like querying through **HQL (Hive Query Language)**. It supports both **batch and real-time processing**, making data retrieval more efficient. Hive integrates with **JDBC and ODBC drivers** for database connectivity and provides an easy-to-use interface for handling large datasets.



**Mahout:** A **machine learning framework** that provides pre-built algorithms for **clustering, classification, and collaborative filtering**. It helps applications learn from data patterns and enables scalable machine learning within Hadoop. Mahout simplifies ML model execution by leveraging Hadoop's distributed computing power.

**Apache Spark:** A **fast, in-memory computing framework** designed for real-time data analytics and large-scale batch processing. It efficiently handles **interactive, iterative, and graph-based computations**, offering better performance than Hadoop's MapReduce. Spark is widely used for streaming data and advanced analytics.

**Apache HBase:** A **NoSQL database** designed for quick read and write operations on massive datasets. Inspired by **Google's BigTable**, HBase is optimized for real-time data access and retrieval. It provides a scalable and fault-tolerant way to store and process structured and unstructured data within the Hadoop ecosystem.





- **Solr & Lucene:**
  - **Lucene:** A Java-based **text search library** offering indexing and spell-checking capabilities.
  - **Solr:** A **search platform** built on Lucene that enhances indexing, searching, and data retrieval efficiency for large datasets.
- **Zookeeper:** A **coordination and synchronization service** for managing distributed applications in Hadoop. It ensures **consistent configuration, communication, and resource management** across different components, preventing conflicts and inconsistencies.
- **Oozie:** A **workflow scheduler** that automates Hadoop job execution by managing dependencies and sequences.
  - **Workflow:** Executes jobs in a defined order.
  - **Coordinator:** Triggers jobs dynamically based on data arrival or external events.

Based On	Hadoop	MapReduce
<b>Definition</b>	Hadoop is an open-source software framework that is used for storing and processing large amounts of data in a distributed computing environment.	MapReduce is a programming model which is implemented for processing and generation Big Data sets with distributed algorithm on a cluster
<b>Invention</b>	Hadoop was created by Doug Cutting and Mike Cafarella	MapReduce was created by Google
<b>Framework</b>	Hadoop has a storage framework which stores data and creates name nodes and data nodes. It also has frameworks that includes MapReduce itself.	MapReduce is a programming framework that has a key and value mappings to process the data.
<b>Features</b>	Hadoop is an open source. The Hadoop cluster is highly scalable.	MapReduce provides a fault tolerance. It also provides with high availability.
<b>Language</b>	Hadoop is a multitude of modules and so may include other programming languages too.	MapReduce is fundamentally written in the java programming language.
<b>Pre-Requisites</b>	Hadoop works on the Hadoop Distributed File System (HDFS)	MapReduce can run on HDFS, GFS, NDFS or any other Distributed System.



# Challenges and Real World applications of Hadoop and MapReduce

Sandeep K-22Z257



## Challenges Related to Hadoop and MapReduce

1. **High Latency** - Inefficient for real-time analytics due to batch processing.
2. **Complex Programming** - Writing MapReduce jobs requires advanced coding skills.
3. **Resource-Intensive** - Demands substantial memory and CPU power.
4. **Data Shuffling Overhead** - Large data movement between phases impacts performance.
5. **Limited Support for Small Files** - Handling many small files reduces efficiency.



## Key Features of MapReduce

- **Scalability:** Efficiently processes terabytes or petabytes of data.
- **Fault Tolerance:** Automatic recovery ensures continued processing even if some nodes fail.
- **Parallel Processing:** Tasks run simultaneously across multiple nodes for faster data processing.
- **Flexibility:** MapReduce can process both structured and unstructured data, making it versatile for various use cases.
- **Cost-Effectiveness:** By commodity hardware, MapReduce provides a cost-efficient solution for big data processing.



# Real-World Applications of MapReduce

## 1. Retail Sales Analysis & Social Media Analytics

- ◆ **Retail Sales Analysis** Tracks trends, predicts demand, and optimizes inventory.

**Example:** Walmart improves stock planning with MapReduce.

- ◆ **Social Media Analytics** Processes user data for trend insights.

**Example:** Facebook studies engagement patterns with MapReduce.



## Real-World Applications of MapReduce

### 2. Healthcare Data Analysis & Financial Fraud Detection

- ◆ **Healthcare Data Analysis** Analyzes medical data to improve diagnostics.

**Example:** Hospitals enhance imaging analysis with MapReduce.

- ◆ **Financial Fraud Detection** Detects suspicious transactions in financial data.

**Example:** PayPal boosts fraud detection using MapReduce.



# Future of Hadoop with MapReduce

- 1. Integration with Cloud Services:** MapReduce adapts for scalability in AWS, Azure, and Google Cloud.
- 2. AI & Machine Learning Integration:** MapReduce aids large-scale data preparation for AI models.
- 3. Real-Time Analytics:** Emerging tools like Apache Spark enhance MapReduce for faster insights.
- 4. Improved Security & Governance:** Future developments focus on stronger encryption and access controls.



The background of the image is a dense, repeating pattern of dark blue, almost black, leaves. The leaves are elongated and pointed, with visible veins. They are arranged in a way that creates a textured, organic feel. A semi-transparent dark blue rectangular box is centered over the image, serving as a backdrop for the text.

*Thank you*