Unit – 3

16 Marks:

1. Hadoop MapReduce Framework

Definition

Hadoop MapReduce is a scalable, distributed computing framework designed for processing large datasets in parallel across a cluster. By dividing tasks into smaller sub-tasks, it enables efficient, fault-tolerant computation, abstracting the complexities of distributed programming.

MapReduce Overview

The MapReduce framework operates in two key phases:

1. **Map Phase**: Processes raw input data to generate intermediate key-value pairs.
2. **Reduce Phase**: Aggregates intermediate data to produce the final results.

This simplifies complex computations by abstracting the details of distributed processing.

Steps of Data Flow in MapReduce

1. **Input Splits**:
    - The dataset is divided into smaller chunks (splits) based on block size (128MB or 256MB).
    - Each split is processed independently.
2. **Map Phase**:
    - The Mapper function processes each split, transforming it into intermediate key-value pairs.
    - Example: In a word count program, the Mapper emits pairs like ("word", 1) for each occurrence of a word.
3. **Shuffle and Sort Phase**:
    - Groups intermediate key-value pairs by key.
    - Sorts the grouped data to ensure keys are processed sequentially in the Reduce phase.
4. **Reduce Phase**:
    - Processes grouped data to generate the final output.
    - Example: In word counting, the Reducer aggregates the counts for each word.
5. **Output Phase**:
    - Writes final results to HDFS in the specified output format.

Brief Working of Mapper

- The Mapper reads input splits line by line, emitting intermediate key-value pairs.

- Example: For server logs, a Mapper could output pairs like ("404", 1) for each occurrence of an error code.

Brief Working of Reducer

- The Reducer takes grouped intermediate pairs and processes them to produce the final output.
- Example: Summing all 1s for a key to compute the total count for that key.

---

2. Architecture of YARN

Definition

YARN (Yet Another Resource Negotiator) is Hadoop's resource management layer. It decouples resource management from job execution, allowing multiple data processing frameworks to share the same cluster resources efficiently.

Key Features

1. **Scalability**: Manages thousands of nodes and applications concurrently.
2. **Compatibility**: Supports various processing models like MapReduce and Spark.
3. **Dynamic Resource Allocation**: Improves cluster utilization by dynamically allocating resources.
4. **Multi-tenancy**: Enables multiple users and frameworks to share resources seamlessly.

Components of YARN Architecture

1. **Client**:
    - Submits jobs and monitors their progress.
2. **Resource Manager (RM)**:
    - Manages cluster resources and schedules jobs.
    - **Scheduler**: Allocates resources based on job requirements.
    - **Applications Manager**: Oversees the lifecycle of applications.
3. **Node Manager (NM)**:
    - Manages resources on individual nodes and executes tasks in containers.
4. **Application Master (AM)**:
    - Manages the lifecycle of a specific application.
    - Coordinates with the Resource Manager for resources and the Node Manager for task execution.
5. **Container**:

- A logical unit of computation, including CPU and memory resources, where tasks execute.

Advantages

- Supports diverse workloads.
- Enhances scalability and fault tolerance.
- Efficiently utilizes cluster resources.

Disadvantages

- Adds complexity to cluster management.
- Debugging can be challenging due to its distributed and decoupled architecture.

---

3. Failures in Classic MapReduce

Definition

Failures in Classic MapReduce occur when tasks or nodes fail during execution. The framework, which relied on a centralized JobTracker, was less fault-tolerant compared to YARN.

Types of Failures

1. **Task Failure**:
   - Occurs when a map or reduce task fails due to software bugs, corrupted data, or insufficient memory.
2. **TaskTracker Failure**:
   - Happens if a TaskTracker (responsible for executing tasks on worker nodes) crashes or loses network connectivity.
3. **JobTracker Failure**:
   - The JobTracker, which schedules and monitors tasks, is a single point of failure. Its failure halts the entire job.

Overcoming Task Failure

- The framework retries failed tasks on other nodes.
- Configuring the number of retries in the job ensures fault tolerance.

---

4. Job Scheduling in MapReduce

Definition

Job scheduling in MapReduce ensures efficient allocation of resources and prioritization of jobs in a multi-user environment.

Hadoop Schedulers

Unit – 3

1. **FIFO Scheduler**:
    - Processes jobs in the order of submission.
    - Simple but unsuitable for environments with multiple users.
2. **Capacity Scheduler**:
    - Divides cluster resources into queues with specific capacities.
    - Ensures priority for specific users or applications.
3. **Fair Scheduler**:
    - Allocates resources equally among all running jobs.
    - Prevents long-running jobs from monopolizing resources.

Advantages

- Optimizes cluster utilization.
- Supports priority-based execution.

Disadvantages

- Advanced schedulers like Capacity and Fair increase complexity.

---

5. MapReduce Types

Definition

MapReduce supports various input and output formats to handle diverse data structures and processing requirements.

Types in MapReduce

1. **Input Types**: Define how data is read. Examples include:
    - **TextInputFormat**: Reads data line by line.
    - **KeyValueInputFormat**: Reads key-value pairs.
    - **SequenceFileInputFormat**: Reads binary data from sequence files.
2. **Output Types**: Define how data is written. Examples include:
    - **TextOutputFormat**: Outputs plain text.
    - **SequenceFileOutputFormat**: Outputs binary key-value pairs.

Java API for MapReduce

The MapReduce Java API provides classes and interfaces to configure and execute jobs:

- **Mapper**: Defines the map function.
- **Reducer**: Defines the reduce function.

- **Job**: Configures and submits jobs.
- **Configuration**: Stores job-specific settings.

Advantages

- Flexible input/output formats enable diverse use cases.
- Sequence File formats improve efficiency for binary data processing.