Unit - 4

16 Marks:

1. File-Based Data Structure

Definition

File-based data structures in Hadoop are designed to efficiently store, read, and write large datasets in a distributed environment. They provide optimized formats to handle key-value pairs, reduce storage overhead, and enhance performance during data processing tasks.

File-Based Data Structures

Hadoop primarily uses two key file formats: **Sequence File** and **Map File**, both designed for efficient data storage and retrieval in Hadoop Distributed File System (HDFS).

1. Sequence File

A Sequence File is a flat file that stores key-value pairs in binary format.

- **Structure**: Contains a header (metadata) and key-value records.
- **Use Case**: Commonly used for intermediate data storage in MapReduce.

2. Map File

A Map File is a sorted and indexed Sequence File.

- **Structure**: Contains key-value pairs with an index file for efficient lookups.
- **Use Case**: Suitable for scenarios requiring random access to data.

Read/Write Sequence File/Map File

- **Writing**: Use Hadoop's API to write data sequentially to these file formats.
- **Reading**: Sequence Files can be read sequentially, whereas Map Files allow random access through indexing.

Benefits of Sequence File/Map File

1. Efficient storage of large datasets.
2. Optimized for distributed processing.
3. Compression support reduces storage overhead.
4. Easy integration with Hadoop tools like Hive and Pig.

Disadvantages of Sequence File/Map File

1. Limited support for schema evolution.
2. Requires understanding of Hadoop APIs for access.
3. Map Files introduce additional overhead for maintaining indexes.

---

2. Integration of Hadoop

Unit - 4

Definition

Hadoop integration refers to combining Hadoop's distributed processing capabilities with other technologies to enhance data analysis, storage, and processing. Integration with tools like R expands Hadoop's utility for statistical analysis and machine learning.

R Integration with Hadoop

Integrating R, a statistical computing tool, with Hadoop allows data scientists to perform statistical modeling and machine learning on large datasets stored in Hadoop.

R-Hadoop Integration Methods

1. **Hadoop Streaming**:
    - Allows R scripts to act as mapper and reducer functions in a MapReduce job.
    - Uses standard input/output to communicate with Hadoop.
2. **RHIPE (R and Hadoop Integrated Programming Environment)**:
    - Provides an R interface for MapReduce using Hadoop.
    - Supports data manipulation and distributed computing in R.
3. **ORCH (Oracle R Connector for Hadoop)**:
    - A commercial tool for integrating R with Hadoop.
    - Provides an R interface for querying HDFS and Hive.

Diagram of R-Hadoop Integration

A typical integration involves:

1. Data storage in HDFS.
2. R scripts running on Hadoop nodes using streaming or connectors.
3. Results stored back in HDFS or processed locally in R.

---

3. HDFS Architecture

Definition

Hadoop Distributed File System (HDFS) is a fault-tolerant file storage system designed for storing and processing large datasets across clusters.

Types of HDFS Files

1. **Data Blocks**: Data is split into fixed-size blocks (default: 128MB).
2. **Metadata**: Contains information about file locations and blocks.

Read and Write Operation in HDFS

1. **Write Operation**:

- Data is split into blocks and distributed across DataNodes.
- Metadata is stored in the NameNode.

2. **Read Operation**:
   - The client fetches metadata from the NameNode and retrieves data blocks from DataNodes.

Design of HDFS

HDFS is designed to handle:

- Fault tolerance through replication.
- High throughput for large file processing.
- Compatibility with commodity hardware.

Distributed File System Processing

- Data is distributed across nodes to enable parallel processing.
- Replication ensures data reliability.

Hadoop Streaming and Pipes

- **Streaming**: Uses standard input/output for MapReduce.
- **Pipes**: Provides a C++ interface for MapReduce applications.

---

4. Hadoop I/O

Definition

Hadoop I/O refers to the mechanisms used to handle data input and output within Hadoop, ensuring efficient processing, storage, and retrieval of data.

Key Concepts

1. **Data Integrity**:
   - Ensures data consistency during read and write operations.
   - Uses checksums to verify data accuracy.
2. **Hadoop Local File System**:
   - A temporary storage layer used during intermediate processing.
   - Faster but not distributed like HDFS.
3. **Compression**:
   - Reduces storage requirements and speeds up data transfer.
   - Common formats: Gzip, Bzip2, Snappy.
4. **Serialization**:

- Converts data objects into byte streams for storage or transfer.
- Used by Hadoop for efficient data processing.

5. **Avro**:
    - A serialization framework supporting schema evolution.
    - Stores data along with its schema for compatibility.

---

5. Hadoop Streaming and Pipes

Definition

Hadoop Streaming and Pipes are tools that allow developers to write MapReduce jobs in languages other than Java.

Features of Hadoop Streaming

1. Executes MapReduce jobs using scripts or binaries.
2. Language-agnostic; works with Python, R, C++, etc.

Code Execution Process in Streaming

1. Input data is passed to the mapper script via standard input.
2. Mapper processes the data and outputs intermediate key-value pairs.
3. Reducer script processes the key-value pairs to produce the final output.

Hadoop Pipes

- Pipes is a C++ API for developing MapReduce applications.
- Provides high performance for compute-intensive tasks.

Execution of Streaming and Pipes

1. Streaming:
    - Write mapper and reducer scripts in your preferred language.
    - Submit the job using Hadoop's streaming utility.
2. Pipes:
    - Write C++ programs using the Hadoop Pipes API.
    - Compile and execute them on Hadoop clusters.