**Kalinga University**

**Faculty of CS & IT**

**Course- BCAAIML Sem-IV  
Subject- Big Data and IoT Security**

**Subject Code: BCAAIML405**

**UNIT-III**

### ****First Steps with the Hadoop Ecosystem****

The Hadoop ecosystem is a collection of open-source tools designed to handle big data storage, processing, and analysis. It operates primarily on distributed systems, offering scalability and fault tolerance.

* **What is the Hadoop Ecosystem?**
  + A framework designed to store and process large datasets across distributed systems.
  + Core components: HDFS (storage) and MapReduce (processing).
  + Supportive tools like Hive, Pig, and HBase enable easier data handling.
* **Advantages of Hadoop Ecosystem:**
  + Scalability: Can scale from a single server to thousands of machines.
  + Cost-effective: Built on commodity hardware.
  + Flexibility: Handles structured, semi-structured, and unstructured data.
* **Core Principles:**
  + Data is stored in a distributed fashion.
  + Processing occurs where the data resides (data locality).

### ****Introduction to Hadoop****

Hadoop is an open-source framework developed by Apache that facilitates the processing of large datasets in a distributed computing environment.

* **Core Components:**
  + **HDFS (Hadoop Distributed File System):**
    - Provides distributed storage.
    - Breaks data into blocks and stores them across multiple nodes.
    - Ensures fault tolerance through data replication.
  + **MapReduce:**
    - A programming model for processing large datasets.
    - Divides tasks into "Map" (data processing) and "Reduce" (aggregating results).
  + **YARN (Yet Another Resource Negotiator):**
    - Manages cluster resources and schedules tasks.
* **Key Features:**
  + Resilience to hardware failure.
  + High throughput for large data sets.
  + Suitable for batch processing.

### ****Exercises****

1. **Setup and Exploration:**
   * Install Hadoop on a single-node cluster.
   * Explore HDFS commands like hdfs dfs -put, hdfs dfs -ls, and hdfs dfs -cat.
2. **Simple MapReduce Program:**
   * Write a Word Count program using MapReduce.
   * Test it on sample text data.
3. **Data Analysis with Hive:**
   * Load a CSV file into Hive.
   * Perform basic SQL operations like SELECT, GROUP BY, and COUNT.
4. **HBase Hands-On:**
   * Create a table in HBase.
   * Insert data and perform CRUD operations.

### ****Hadoop Components****

The Hadoop ecosystem includes several tools and technologies that complement its core functionality:

1. **HDFS (Storage Layer):**
   * Distributed and fault-tolerant file system.
2. **MapReduce (Processing Layer):**
   * Framework for parallel data processing.
3. **YARN:**
   * Resource management and job scheduling.
4. **Additional Components:**
   * **Pig:** For scripting and data transformation.
   * **Hive:** For querying data using SQL-like syntax.
   * **HBase:** For NoSQL storage and real-time access.
   * **Sqoop:** For importing/exporting data between Hadoop and relational databases.
   * **Flume:** For ingesting log and event data into Hadoop.

### ****MapReduce****

MapReduce is a programming model used for distributed data processing. It splits the processing into two main phases:

1. **Map Phase:**
   * Processes input data and generates intermediate key-value pairs.
   * Example: Counting words in a text file (key: word, value: count).
2. **Reduce Phase:**
   * Aggregates the intermediate key-value pairs to produce the final output.
   * Example: Summing the counts for each unique word.

* **How It Works:**
  + Input data is divided into splits and processed by mappers.
  + Reducers consolidate the mapper outputs.
* **Key Features:**
  + Fault-tolerant due to data replication.
  + Scalable to handle large datasets.

### ****Pig****

Pig is a high-level platform for creating MapReduce programs, primarily used for data manipulation and transformation.

* **Pig Latin:**
  + A scripting language used to write Pig programs.
  + Easier than writing Java-based MapReduce code.
* **Execution Modes:**
  + Local Mode: Runs on a single machine for smaller data sets.
  + MapReduce Mode: Processes data in a Hadoop cluster.
* **Common Use Cases:**
  + ETL (Extract, Transform, Load) processes.
  + Data cleaning and preparation.

### ****Hive****

Hive is a data warehousing tool built on top of Hadoop. It allows querying and managing large datasets using an SQL-like language called HiveQL.

* **Key Features:**
  + Familiar SQL-like interface.
  + Schema-based storage (tables, rows, and columns).
  + Supports functions and joins.
* **Components:**
  + Metastore: Stores metadata about tables and schemas.
  + Driver: Compiles and executes HiveQL queries.
* **Use Cases:**
  + Data summarization.
  + Ad-hoc querying.
  + Analysis of large datasets.

### ****HBase****

HBase is a NoSQL database that provides real-time read/write access to large datasets stored in HDFS.

* **Features:**
  + Distributed and scalable.
  + Column-oriented storage.
  + Supports random access and updates.
* **Components:**
  + HMaster: Manages the cluster and region servers.
  + Region Servers: Store and manage data.
  + Zookeeper: Coordinates distributed systems.
* **Common Use Cases:**
  + Real-time analytics.
  + Storing sparse datasets.
  + Applications needing high throughput and low latency.

### ****Working with Hadoop****

Hadoop is an open-source framework designed to store and process large datasets in a distributed computing environment. It enables the efficient processing of vast amounts of data across a cluster of computers. The Hadoop ecosystem is built around the Hadoop Distributed File System (HDFS) and MapReduce programming model. Here’s how you can work with Hadoop:

* **Hadoop Distributed File System (HDFS):**  
  HDFS is the primary storage layer in Hadoop. It is designed to store large files by splitting them into blocks and distributing these blocks across multiple machines in a cluster. HDFS ensures fault tolerance and high availability by replicating data blocks. A typical Hadoop cluster consists of:
  + **NameNode:** The master server that manages the file system’s namespace.
  + **DataNodes:** Worker nodes that store the actual data blocks.
* **MapReduce Framework:**  
  MapReduce is the core processing model in Hadoop, designed to process large data sets in parallel across a distributed environment. It breaks the process into two steps:
  + **Map Phase:** The input data is divided into chunks, and a map function is applied to each chunk to process it.
  + **Reduce Phase:** After the map phase, the output is combined and processed by the reduce function to provide the final output.
* **Ecosystem Components:**  
  Hadoop has several ecosystem components that make it more powerful and flexible:
  + **Hive:** A data warehouse infrastructure for querying and managing large datasets. It allows users to write SQL-like queries to access data stored in HDFS.
  + **Pig:** A platform for analyzing large datasets using a language called Pig Latin, which is a higher-level abstraction of MapReduce.
  + **HBase:** A NoSQL database built on top of HDFS, providing real-time random read/write access to large datasets.
  + **YARN:** Yet Another Resource Negotiator, responsible for resource management in Hadoop clusters.

### 2. ****Loading Data into Hadoop****

Loading data into Hadoop is an essential step in utilizing its distributed computing capabilities. Hadoop can handle a variety of data formats, such as structured, semi-structured, and unstructured data. Here are some common methods for loading data into Hadoop:

* **Using the Hadoop Command Line Interface (CLI):**  
  The hadoop fs command provides an interface for interacting with HDFS. The -put and -copyFromLocal commands are often used to load data from a local file system into HDFS.

Example:

bash

Copy code

hadoop fs -put /local/data/file.txt /user/hadoop/input/

* **Using Apache Sqoop:**  
  Apache Sqoop is a tool designed for efficiently transferring bulk data between Hadoop and relational databases. It allows users to import data from databases like MySQL, PostgreSQL, and Oracle into HDFS and export data back into these databases.

Example:

bash

Copy code

sqoop import --connect jdbc:mysql://localhost/db --table users --target-dir /user/hadoop/input

* **Using Apache Flume:**  
  Apache Flume is a distributed service for collecting, aggregating, and moving large amounts of log data into HDFS. It is often used to stream data from web servers, application logs, or social media feeds.
* **Using Kafka (for Real-Time Data):**  
  Apache Kafka can be used for real-time data streaming into Hadoop. Kafka captures streaming data and pushes it into HDFS or HBase for real-time analytics.

### 3. ****Handling Files in Hadoop****

In Hadoop, managing files within HDFS involves a range of tasks such as uploading, listing, copying, and deleting files. The hadoop fs command allows users to interact with the file system, manage files, and ensure data is stored and retrieved efficiently.

* **Listing Files:**  
  To list files in a directory within HDFS, use the following command:

bash

Copy code

hadoop fs -ls /user/hadoop/

* **Copying Files:**  
  You can copy files between the local file system and HDFS using the -copyToLocal or -copyFromLocal commands. For example:

bash

Copy code

hadoop fs -copyFromLocal /local/path/to/file /user/hadoop/hdfs/

* **Renaming Files:**  
  Hadoop provides the ability to rename files within HDFS. This can be done using the -mv command:

bash

Copy code

hadoop fs -mv /user/hadoop/oldfile.txt /user/hadoop/newfile.txt

* **Deleting Files:**  
  To delete files from HDFS, the -rm command is used:

bash

Copy code

hadoop fs -rm /user/hadoop/file.txt

* **Getting Data from HDFS to Local File System:**  
  You can retrieve files from HDFS to the local file system using the -get command:

bash

Copy code

hadoop fs -get /user/hadoop/input/file.txt /local/path/to/destination/

* **Checking Disk Space:**  
  To check the disk space usage of HDFS, you can use the -du command:

bash

Copy code

hadoop fs -du -h /user/hadoop/

### 4. ****Getting Data from Hadoop****

Once the data is loaded into HDFS, you may need to retrieve or query it for analysis, reporting, or further processing. There are several tools and methods for accessing data in Hadoop:

* **Using Hadoop CLI (hadoop fs):**  
  The Hadoop CLI is a basic way to access files stored in HDFS. Commands like -cat, -get, and -tail can be used to view or retrieve data directly from HDFS.
  + **View File Contents:**

bash

Copy code

hadoop fs -cat /user/hadoop/input/file.txt

* **Using Hive for Querying Data:**  
  Hive provides an SQL-like interface to query large datasets in Hadoop. You can query the data stored in HDFS using Hive’s Query Language (HQL).

Example:

sql

Copy code

SELECT \* FROM users WHERE age > 30;

Hive allows you to perform operations like filtering, aggregation, and joining datasets, all while abstracting away the complexities of MapReduce.

* **Using Pig for Data Transformation:**  
  Pig allows users to perform data transformations using a high-level scripting language called Pig Latin. Data can be loaded from HDFS, transformed, and then stored back in HDFS or exported to another system.

Example of loading data into Pig:

pig

Copy code

users = LOAD '/user/hadoop/input/users.csv' USING PigStorage(',') AS (name:chararray, age:int);

* **Using HBase for Real-Time Access:**  
  If you need random read/write access to large datasets, HBase (a NoSQL database built on top of HDFS) is ideal. You can use the HBase shell or APIs to retrieve and manipulate data in real-time.

Example using HBase shell:

bash

Copy code

scan 'users'

* **Using Apache Spark for Advanced Analytics:**  
  Apache Spark can be used to process large datasets in real-time from Hadoop’s HDFS. Spark provides in-memory processing and supports data analytics, machine learning, and graph processing.

Example using Spark to load and process data from HDFS:

python

Copy code

from pyspark import SparkContext

sc = SparkContext("local", "Hadoop Example")

data = sc.textFile("hdfs://namenode:9000/user/hadoop/input/data.txt")

### SQL and Querying Big Data

**1. Introduction to the SQL Language**

SQL (Structured Query Language) is the standard programming language for managing and manipulating relational databases. It provides a set of commands for performing tasks such as querying data, updating records, inserting new rows, and deleting data. SQL is crucial in the world of big data, particularly for querying structured data stored in relational databases. However, when dealing with big data systems like Hadoop, traditional SQL can encounter limitations in performance and scalability. To address this, adaptations of SQL, such as Apache Hive, have been created to handle massive volumes of data more effectively.

**Key Concepts in SQL:**

* **SELECT**: Retrieves data from one or more tables.
* **INSERT**: Adds new records to a table.
* **UPDATE**: Modifies existing records.
* **DELETE**: Removes records from a table.
* **WHERE**: Specifies the conditions for filtering records.
* **JOIN**: Combines data from two or more tables based on related columns.
* **GROUP BY**: Groups rows that have the same values into summary rows, often used with aggregate functions like COUNT, SUM, AVG, etc.
* **ORDER BY**: Sorts the result set in ascending or descending order.

**SQL in Big Data:**

* In the context of big data, SQL-like queries can be used to analyze large datasets that are stored in distributed file systems like HDFS (Hadoop Distributed File System).
* For example, SQL queries can be used to perform complex data transformations and aggregations, allowing analysts to extract insights from petabytes of data.
* However, due to the sheer size and distributed nature of big data, traditional SQL is not suitable for large-scale processing. This is where SQL engines designed for big data, like Hive, come into play.

**2. Querying Big Data with Hive**

Apache Hive is a data warehouse infrastructure built on top of Hadoop. It provides a high-level query language, which is a SQL-like language called HiveQL, allowing users to query and analyze large datasets stored in Hadoop's HDFS. Hive abstracts the complexity of MapReduce and provides a more user-friendly interface to interact with Hadoop using SQL-like syntax.

**Hive Architecture:**

* **Metastore**: Hive stores metadata in a centralized repository known as the Metastore. The Metastore contains information about the structure of the data (schemas, tables, partitions) and allows Hive to know how to map the queries to the actual data stored in HDFS.
* **HiveQL**: The query language used in Hive is similar to SQL but tailored for the Hadoop ecosystem. It allows querying, inserting, updating, and deleting data in Hadoop.
* **Execution Engine**: Hive translates the SQL-like queries into MapReduce jobs, which are then executed across the Hadoop cluster. As of newer versions of Hive, other execution engines like Apache Tez and Apache Spark can be used for faster query execution.

**Key Features of Hive for Querying Big Data:**

* **SQL-Like Queries**: HiveQL supports most of the typical SQL operations, including SELECT, JOIN, GROUP BY, ORDER BY, and aggregations.
* **Partitioning**: Hive allows partitioning of large datasets, which divides data into smaller, manageable parts based on column values (e.g., by date or region). This helps optimize query performance by reducing the amount of data read during query execution.
* **Bucketing**: Bucketing is another technique in Hive that divides data into more manageable parts (buckets) based on a hash function. This further optimizes query performance when dealing with large datasets.
* **Data Types**: Hive supports a wide range of data types such as INT, STRING, DOUBLE, and more complex types like MAP, ARRAY, and STRUCT for working with semi-structured or complex data.
* **External Tables**: Hive allows creating external tables that map to data stored outside of Hive’s default storage location, such as HDFS or other distributed file systems. This makes it easier to integrate data stored in various systems.

**Sample Query in Hive:**

sql

Copy code

SELECT department, AVG(salary)

FROM employees

WHERE join\_date > '2020-01-01'

GROUP BY department;

* This query calculates the average salary for each department in the employees table, where employees joined after January 1, 2020. Hive translates this SQL query into a series of MapReduce jobs to run on Hadoop.

**Performance Optimizations in Hive:**

* **Tez Execution Engine**: In newer versions, the Tez execution engine is used for faster execution of HiveQL queries as it avoids the overhead of MapReduce jobs.
* **Apache Spark**: Spark can also be used as the execution engine for Hive, providing even faster query performance due to its in-memory processing capabilities.
* **Indexes**: Hive supports indexing on tables to improve query performance by allowing faster retrieval of rows.
* **Query Caching**: Hive has the capability to cache query results, enabling faster subsequent queries on the same data.

**When to Use Hive for Big Data:**

* Hive is particularly useful when working with structured data, where SQL-like querying is required but the data scale exceeds what can be handled by traditional databases.
* It’s ideal for ETL (Extract, Transform, Load) tasks, data analytics, and batch processing on massive datasets stored in Hadoop.

**Limitations of Hive:**

* **Latency**: Hive is optimized for batch processing, which means it may not be suitable for real-time querying.
* **Not Suitable for Transactional Data**: Hive is not a transactional system and lacks some of the advanced features provided by traditional relational databases, such as ACID (Atomicity, Consistency, Isolation, Durability) properties.
* **Performance**: While Hive can handle large datasets, its performance may lag behind systems like Apache HBase or Apache Spark, especially in low-latency or real-time use cases.