Page No.
Date

CCA2

Q1 Explain the YARN architecture in Hadoop. Discuss the roles of Resource Manager, Node Manager, Application Master & Containers. How does YARN manage rss accross multiple jobs?

→ YARN (Yet Another Resource Negotiator) is a resource management layer of Hadoop that allows for the management of resources in a Hadoop cluster.

Resource Manager: This is the master daemon responsible for managing the allocation of resources across all applications. It tracks the available resources & negotiates their allocation.

Node Manager: A per-node daemon responsible for monitoring resource usage (CPU, memory) of containers running on that node & reporting this information to the Resource Manager.

Application Master: Each application (MapReduce job) has its own instance of the Application Master that negotiates resources from the Resource Manager & works with the Node Manager to execute the tasks.

Containers: These are the units of processing that YARN allocates. A container encapsulates the resource requirements (memory, CPU) for a task.

Resource Management: Yarn efficiently manages resources by allowing multiple applications to run simultaneously

dynamically allocating resources to each application as needed based on its demands, optimizing cluster utilization.

Q2 Discuss how Apache Flume & Sqoop are used for data ingestion in the Hadoop ecosystem. Compare their use cases, advantages & challenges in transferring data into HDFS from various sources.

→ Apache Flume:-

- Use case: Primarily used for streaming log data into HDFS. Flume can collect data from various sources such as web servers & send it to HDFS in real time.
- Advantages: Highly scalable, fault-tolerant, & can handle large volumes of streaming data. It supports complex data flows.
- Challenges: Configuration can be complex & performance may degrade if not properly tuned for high-throughput scenarios.

Sqoop:-

- Use case: Used for batch data transfer between Hadoop & relational databases. Ideal for importing large datasets into HDFS from SQL databases.
- Advantages: Optimized for bulk transfer, supports parallel data transfer & can import & export data with ease.
- Challenges: Limited to structured data, handling complex data types may require additional transformation steps

Comparison: Flume is suited for continuous data ingestion (real-time) from various sources, while Sqoop is designed for transferring batch data from structured databases. Flume excels in handling unstructured data streams while Sqoop is better for structured data operation.

Q3 Explain how matrix multiplication can be implemented using the MapReduce framework. What are the key steps in dividing the task between the Mapper & Reducer phases for matrix multiplication?

→ Matrix multiplication using MapReduce involves dividing the multiplication task into smaller manageable sub-tasks processed in parallel.

Key steps:

Input Representation -
Each matrix (A & B) is represented as key-value pairs where keys represent the matrix indices & values represent the matrix values.

Mapper Phase :-
Mapper reads the input data for both matrices & emits intermediate key-value pairs based on the indices.
for Matrix A, emits pairs in the form of (row_index, (matrix A, column_index, value)).
for Matrix B, emits pairs as (column_index, (matrix_B, row_index, value)).
This step prepares the data for the Reducer by associating rows of Matrix A with corresponding

columns of Matrix B.

Shuffle & Sort:-
The MapReduce framework sorts the emitted key-value pairs, ensuring that all values corresponding to the same key (column index of B) are sent to the same Reducer.

Reducer Phase:-
Receives all pairs for a specific column index. It computes the dot product of the row from Matrix A & the column from Matrix B by iterating through the values:
for each (column index), calculate the sum of products of corresponding values to form a single entry in the resultant matrix.

Output:
The final output consists of key-value pairs representing the resultant matrix indices & their computed values. This approach efficiently utilizes the MapReduce framework to parallelize matrix multiplication, enabling the processing of large matrices that exceed memory limits of a single machine.