(Big) Data Engineering In Depth

Moustafa Mahmoud
Data Solution Architect

Outline

Chapter: Hive

Chapter Objectives

- Query data lakes/HDFS using Hive.

Chapter Objectives

- Query data lakes/HDFS using Hive.
- Introduction to Hive

Chapter Objectives

- Query data lakes/HDFS using Hive.
- Introduction to Hive
- Comparing Hive to Traditional databases.

Chapter Objectives

- Query data lakes/HDFS using Hive.
- Introduction to Hive
- Comparing Hive to Traditional databases.
- Hive Components and Architecture.

Chapter Objectives

- Query data lakes/HDFS using Hive.
- Introduction to Hive
- Comparing Hive to Traditional databases.
- Hive Components and Architecture.
- Relational Data Analysis with Hive.

Chapter Objectives

- Query data lakes/HDFS using Hive.
- Introduction to Hive
- Comparing Hive to Traditional databases.
- Hive Components and Architecture.
- Relational Data Analysis with Hive.
- Hive Data Management.

Chapter Objectives

- Query data lakes/HDFS using Hive.
- Introduction to Hive
- Comparing Hive to Traditional databases.
- Hive Components and Architecture.
- Relational Data Analysis with Hive.
- Hive Data Management.
- Hive Optimization.

Section: Query data lakes/HDFS using Hive

Query data lakes/HDFS using Hive

- Data lakes and HDFS store vast amounts of petabyte-scale data.

Query data lakes/HDFS using Hive

- Data lakes and HDFS store vast amounts of petabyte-scale data.
- Data analysts require efficient ways to query this data without delving into intricate Map-Reduce programming.

Query data lakes/HDFS using Hive

- Data lakes and HDFS store vast amounts of petabyte-scale data.
- Data analysts require efficient ways to query this data without delving into intricate Map-Reduce programming.
- SQL is a commonly used language for data manipulation, embraced by data analysts, developers, and business users worldwide.

Query data lakes/HDFS using Hive

- Data lakes and HDFS store vast amounts of petabyte-scale data.
- Data analysts require efficient ways to query this data without delving into intricate Map-Reduce programming.
- SQL is a commonly used language for data manipulation, embraced by data analysts, developers, and business users worldwide.
- Hive, one of the early tools in this field, simplifies data analysis on big data by offering SQL-like querying capabilities for data lakes and HDFS.

Query data lakes/HDFS using Hive

- Data lakes and HDFS store vast amounts of petabyte-scale data.
- Data analysts require efficient ways to query this data without delving into intricate Map-Reduce programming.
- SQL is a commonly used language for data manipulation, embraced by data analysts, developers, and business users worldwide.
- Hive, one of the early tools in this field, simplifies data analysis on big data by offering SQL-like querying capabilities for data lakes and HDFS.
- There are other tools like Trino/Presto, Snowflake, and Databricks that can be faster for data lake queries, which we will discuss later.

Query data lakes/HDFS using Hive

- Data lakes and HDFS store vast amounts of petabyte-scale data.
- Data analysts require efficient ways to query this data without delving into intricate Map-Reduce programming.
- SQL is a commonly used language for data manipulation, embraced by data analysts, developers, and business users worldwide.
- Hive, one of the early tools in this field, simplifies data analysis on big data by offering SQL-like querying capabilities for data lakes and HDFS.
- There are other tools like Trino/Presto, Snowflake, and Databricks that can be faster for data lake queries, which we will discuss later.

Section: Introduction to hive

Introduction to hive

- Apache Hive is a powerful data warehousing and SQL-like query language tool within the Hadoop ecosystem.

Introduction to hive

- Apache Hive is a powerful data warehousing and SQL-like query language tool within the Hadoop ecosystem.
- It was developed by Facebook and later contributed to the Apache Software Foundation, making it an open-source project.

Introduction to hive

- Apache Hive is a powerful data warehousing and SQL-like query language tool within the Hadoop ecosystem.

- It was developed by Facebook and later contributed to the Apache Software Foundation, making it an open-source project.

- Apache Hive had its initial release on October 1, 2010.

Introduction to hive

- Apache Hive is a powerful data warehousing and SQL-like query language tool within the Hadoop ecosystem.

- It was developed by Facebook and later contributed to the Apache Software Foundation, making it an open-source project.

- Apache Hive had its initial release on October 1, 2010.

Introduction to hive

- Hive provides a user-friendly interface to work with and analyze large datasets stored in Hadoop Distributed File System (HDFS) or other compatible storage systems.

Introduction to hive

- Hive provides a user-friendly interface to work with and analyze large datasets stored in Hadoop Distributed File System (HDFS) or other compatible storage systems.

- With its SQL-like language called HiveQL, users can write queries to extract, transform, and analyze data, making it accessible to a wide range of data professionals

Introduction to hive

- Hive provides a user-friendly interface to work with and analyze large datasets stored in Hadoop Distributed File System (HDFS) or other compatible storage systems.

- With its SQL-like language called HiveQL, users can write queries to extract, transform, and analyze data, making it accessible to a wide range of data professionals

- Apache Hive bridges the gap between the world of big data and traditional relational databases, making it a valuable tool for data engineers, analysts, and data scientists.

Introduction to hive

- Hive provides a user-friendly interface to work with and analyze large datasets stored in Hadoop Distributed File System (HDFS) or other compatible storage systems.

- With its SQL-like language called HiveQL, users can write queries to extract, transform, and analyze data, making it accessible to a wide range of data professionals

- Apache Hive bridges the gap between the world of big data and traditional relational databases, making it a valuable tool for data engineers, analysts, and data scientists.

Sub-Section: Overview of Apache Hive

Overview of Apache Hive

- Apache Hive is a high-level data warehousing and SQL-like query language tool built on top of MapReduce.

Overview of Apache Hive

- Apache Hive is a high-level data warehousing and SQL-like query language tool built on top of MapReduce.
- It acts as an abstraction layer, allowing users to work with large datasets stored in Hadoop Distributed File System (HDFS) without needing to write complex MapReduce jobs themselves.

Overview of Apache Hive

- Apache Hive is a high-level data warehousing and SQL-like query language tool built on top of MapReduce.
- It acts as an abstraction layer, allowing users to work with large datasets stored in Hadoop Distributed File System (HDFS) without needing to write complex MapReduce jobs themselves.
- Under the hood, Hive generates MapReduce jobs that are executed on the Hadoop cluster. This means it leverages the distributed processing capabilities of Hadoop for scalability and parallel processing.

Overview of Apache Hive

- Apache Hive is a high-level data warehousing and SQL-like query language tool built on top of MapReduce.

- It acts as an abstraction layer, allowing users to work with large datasets stored in Hadoop Distributed File System (HDFS) without needing to write complex MapReduce jobs themselves.

- Under the hood, Hive generates MapReduce jobs that are executed on the Hadoop cluster. This means it leverages the distributed processing capabilities of Hadoop for scalability and parallel processing.

- Hive has evolved beyond MapReduce to support other execution engines, such as Tez and Spark, which can enhance performance and usability. This will be discussed later.

Overview of Apache Hive

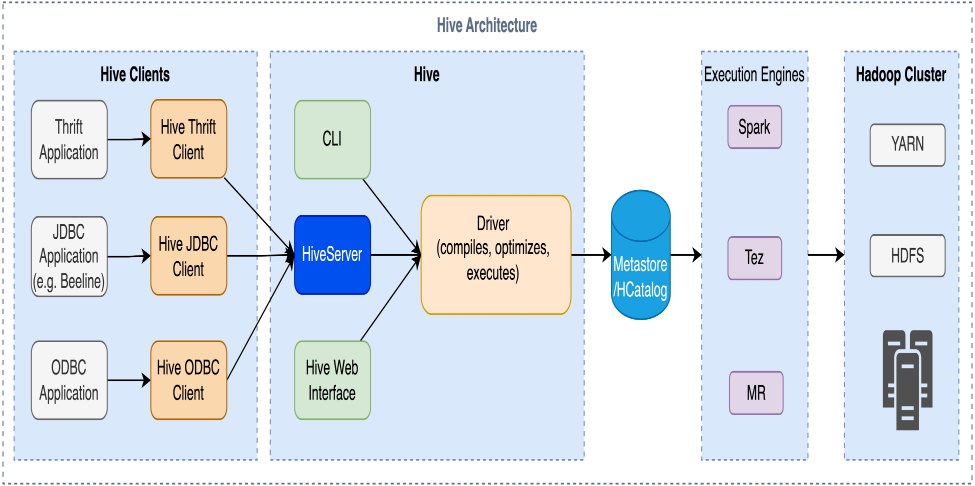
- Apache Hive is a high-level data warehousing and SQL-like query language tool built on top of MapReduce.
- It acts as an abstraction layer, allowing users to work with large datasets stored in Hadoop Distributed File System (HDFS) without needing to write complex MapReduce jobs themselves.
- Under the hood, Hive generates MapReduce jobs that are executed on the Hadoop cluster. This means it leverages the distributed processing capabilities of Hadoop for scalability and parallel processing.
- Hive has evolved beyond MapReduce to support other execution engines, such as Tez and Spark, which can enhance performance and usability. This will be discussed later.

Section: Hive Architecture

Abstract Components of Apache Hive

- Hive Clients.
- Hive Services.
- Hive Storage and Computing.

Abstract Components of Apache Hive



Hive Clients

- Hive provides various drivers for seamless communication with different types of applications.
- For Thrift-based applications, Hive offers the Thrift client for effective communication.
- If you are working with Java-related applications, Hive provides JDBC drivers for smooth integration.
- Additionally, for other types of applications, Hive offers ODBC drivers, ensuring versatility.
- These clients and drivers serve as intermediaries, connecting your applications with Hive Server in the Hive services.

Hive Services

- Hive Services act as the intermediaries for client interactions with Hive.
- When clients need to perform query-related operations in Hive, they communicate through Hive Services.
- All drivers, including JDBC, ODBC, and other client-specific applications, communicate with Hive Server and the primary driver within Hive Services.
- The main driver in Hive Services processes requests from various applications, directing them to the metastore and data systems for further processing.

Hive Services | CLI (Continued)

- Hive Command Line Interface (CLI)
  - The CLI is the most common way to access Hive.
  - Its design can make it challenging to use programmatically.
  - It is a fat client, requiring a local copy of all Hive components and configurations.
  - It needs a copy of a Hadoop client and its configuration.
  - The CLI functions as an HDFS client, a MapReduce client, and a JDBC client (for accessing the metastore).
  - Even with the correct client installation, ensuring all necessary network access can be complex, especially across subnets or datacenters.

Hive Services | HiveServer2 (Continued)

- **HiveServer2**: HiveServer2 is a service that allows clients to submit HiveQL queries programmatically. It provides a remote interface for running Hive queries and managing sessions.
- **JDBC and ODBC**: Hive supports JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity) protocols, enabling users to connect to Hive using popular programming languages and tools.
- **Thrift Service**: Hive uses the Apache Thrift framework to provide a cross-language service interface. This enables communication between clients and the Hive server.
- **Sessions**: When clients connect to HiveServer2, sessions are established to manage their interactions. Sessions help keep track of query state and context.

Hive Services | Hive Driver

- The Hive Driver is a critical component responsible for query execution.
- It consists of several key components:
    - Query Compiler.
    - Optimizer.
    - Execution
- Together, these components ensure efficient and effective query processing in Hive.

Hive Services | Hive Driver (Continued)

- Query Compiler
  - The Query Compiler takes HiveQL queries and translates them into executable jobs.
  - It's responsible for the logical and physical query planning, ensuring that the queries are optimized for efficient execution.

Hive Services | Hive Driver (Continued)

- Query Optimizer
    - It applies optimization techniques, including predicate pushdown and join optimization, to enhance query performance.
    - This ensures that queries are executed as efficiently as possible.

Hive Services | Hive Driver (Continued)

- The Execution Engine is responsible for the actual execution of queries.
- It encompasses several key tasks, including:
    - **Plan Execution**: It executes the query plan generated by the Query Compiler.
    - **Job(s) Generation**: Depending on the chosen execution engine (e.g., MapReduce), it generates the necessary jobs to process data in parallel.
    - **Submission to Hadoop**: It submits these jobs to the Hadoop cluster or other compatible compute environments for execution.
    - **Progress Monitoring**: It continuously monitors the progress of the query execution, providing insights into job completion and overall performance.

Metadata Store (e.g., MySQL)

- The Metadata Store is a relational database, such as MySQL, that stores critical information about tables, columns, partitions, and their relationships.
- This database acts as a catalog, enabling Hive to understand the data’s structure and schema.

Data Storage

- Hive operates on data stored in HDFS or compatible storage systems.
- Instead of transforming the data, it interprets it using a schema on read approach.
- This allows users to work with data without the need for extensive data preparation.

Sub-Section: Job Execution Flow in Hive

Job Execution Flow in Hive

- Receive SQL Query.
    - Parse HiveQL.
    - Make optimization.
    - Plan execution.
    - Submit job(s) to the cluster.
    - Monitor the progress.
    - Process the data in MapReduce or Spark.
    - Store the data in HDFS.

Sub-Section: Hive Schema and Data Storage

Hive Schema and Data Storage

- Hive queries operate on tables, similar to RDBMS.
    - A table corresponds to a directory in storage (HDFS, S3, GCS, or Azure).
    - Each table comprises one or more files.
    - Every table is associated with a specific file format.
    - Hive stores table structure and location in the metadata store (RDBMS).
    - Hive supports various file formats, such as Parquet, ORC, and Text.

Hive Schema and Data Storage (Continued)

- Hive queries reference the metastore to access table location and structure.
- While queries interact with the file system, metadata is stored in the RDBMS.

Section: Performance Tuning

Sub-Section: Query Execution Plan

Query Execution Plan

- The Hive driver is responsible for translating SQL statements into an execution plan for the target execution engine.

- The process involves several key steps:
  1. The parser parses the SQL statement and generates an Abstract Syntax Tree (AST) representing logical operations like SELECTs, JOINs, UNIONs, groupings, and more.

  2. The planner retrieves table metadata from the Hive Metastore, including HDFS file locations, storage formats, row counts, etc.

  3. The query optimizer utilizes the AST and table metadata to produce a physical operation tree known as the execution plan, defining the physical operations needed to retrieve data, such as nested loop joins, sort-merge joins, hash joins, index joins, and more.

Query Execution Plan

- The execution plan determines the tasks executed on the Hadoop cluster and significantly impacts performance in data analytics systems like Hive.
- The execution plan generated by the query optimizer has a substantial impact on performance.
- Differences in the execution plan can result in significant variations in execution time, ranging from seconds to hours.
- An optimal execution plan is crucial for efficient query processing in Hive.

Query Execution Plan

- The Cost-Based Optimization (CBO) plays a pivotal role in enhancing the execution plan.
- CBO leverages table statistics to make informed decisions regarding the performance costs associated with each potential execution plan.
- This intelligent optimization ensures that the Hive driver produces an optimal execution plan, improving query performance.

Sub-Section: Cost-Based Optimization

Section: Further Readings and Assignment

Thank you for watching!

See you in the next video ☺