HIVE ARCHITECTURE & COMPONENTS

BIG DATA HADOOP & SPARK TRAINING

ACADGILD

ASSIGNMENT 6.3

BY :-

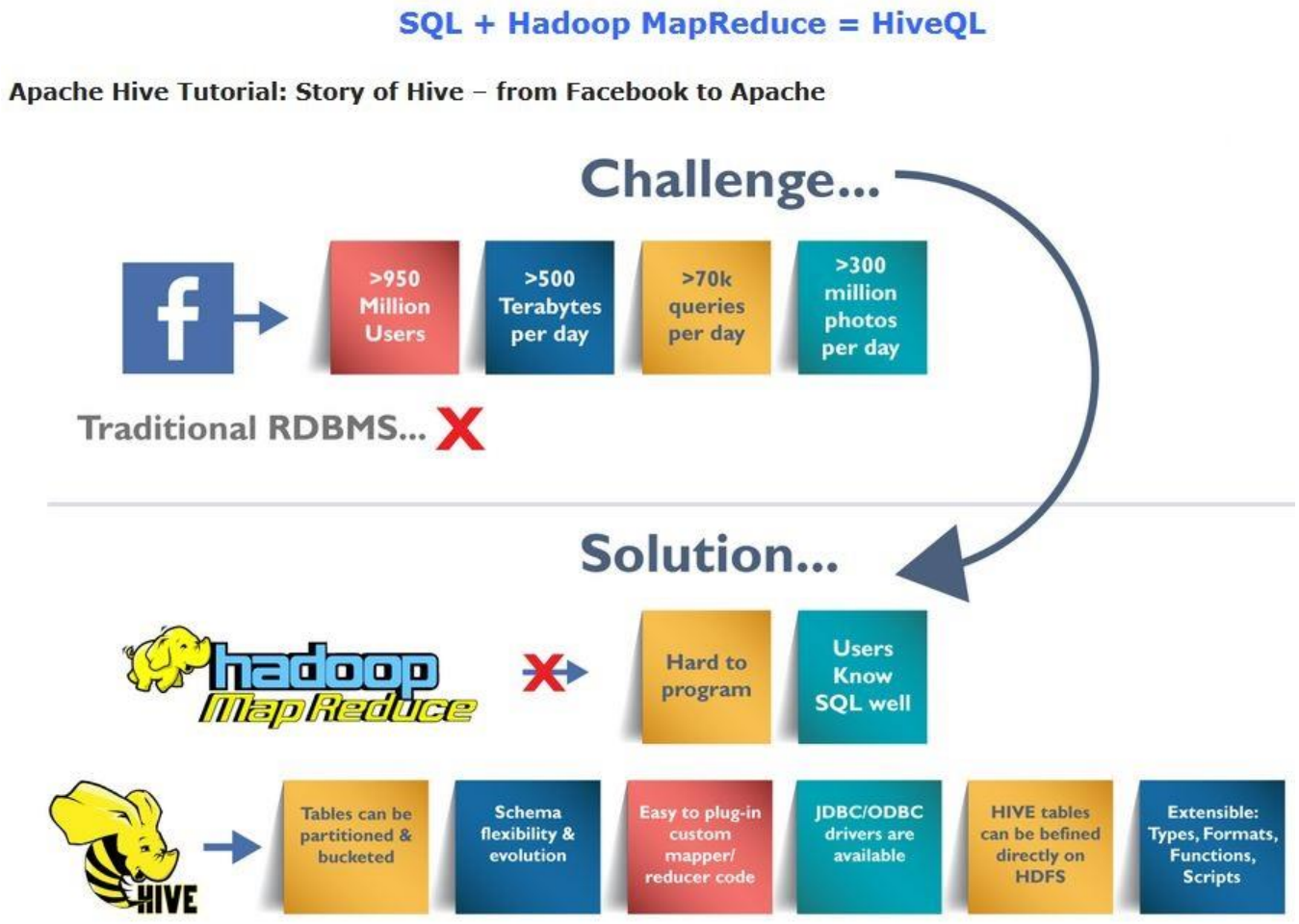
SAHIL KHURANA

What is HIVE?

Hive is an ETL and Data warehousing tool developed on top of Hadoop Distributed File System (HDFS). Hive makes job easy for performing operations like

- Data encapsulation
- Ad-hoc queries
- Analysis of huge datasets

Basically, it provides a mechanism to project structure onto the data and perform queries written in HQL (Hive Query Language) that are similar to SQL statements. Internally, these queries or HQL gets converted to map reduce jobs by the Hive compiler. Therefore, you don't need to worry about writing complex MapReduce programs to process your data using Hadoop. It is targeted towards users who are comfortable with SQL. Apache Hive supports Data Definition Language (DDL), Data Manipulation Language (DML) and User Defined Functions (UDF).



Challenges at Facebook: Exponential Growth of Data

Before 2008, all the data processing infrastructure in Facebook was built around a data warehouse based on commercial RDBMS. These infrastructures were capable enough to suffice the needs of Facebook at that time. But, as the data started growing very fast, it became a huge challenge to manage and process this huge dataset. According to a Facebook article, the data scaled from a 15 TB data set in 2007 to a 2 PB data in 2009. Also, many Facebook products involve analysis of the data like Audience Insights, Facebook Lexicon, Facebook Ads, etc. So, they needed a scalable and economical solution to cope up with this very problem and, therefore started using the Hadoop framework.

Important characteristics of Hive

1. In Hive, tables and databases are created first and then data is loaded into these tables.
2. Hive as data warehouse designed for managing and querying only structured data that is stored in tables.
3. While dealing with structured data, Map Reduce doesn't have optimization and usability features like UDFs but Hive framework does. Query optimization refers to an effective way of query execution in terms of performance.
4. Hive's SQL-inspired language separates the user from the complexity of Map Reduce programming. It reuses familiar concepts from the relational database world, such as tables, rows, columns and schema, etc. for ease of learning.
5. Hadoop's programming works on flat files. So, Hive can use directory structures to "partition" data to improve performance on certain queries.
6. A new and important component of Hive i.e. Metastore used for storing schema information. This Metastore typically resides in a relational database. We can interact with Hive using methods like
    - Web GUI
    - Java Database Connectivity (JDBC) interface
7. Most interactions tend to take place over a command line interface (CLI). Hive provides a CLI to write Hive queries using Hive Query Language(HQL)
8. Generally, HQL syntax is similar to the SQL syntax that most data analysts are familiar with. The Sample query below display all the records present in mentioned table name.
    - Sample query : Select * from <TableName>
9. Hive supports four file formats those are TEXTFILE, SEQUENCEFILE, ORC and RCFILE (Record Columnar File).
10. For single user metadata storage, Hive uses derby database and for multiple user Metadata or shared Metadata case Hive uses MYSQL.

Some of the key points about Hive:

- The major difference between HQL and SQL is that Hive query executes on Hadoop's infrastructure rather than the traditional database.
- The Hive query execution is going to be like series of automatically generated map reduce Jobs.
- Hive supports partition and buckets concepts for easy retrieval of data when the client executes the query.
- Hive supports custom specific UDF (User Defined Functions) for data cleansing, filtering, etc. According to the requirements of the programmers one can define Hive UDFs.

Hive Vs Relational Databases:-

By using Hive, we can perform some peculiar functionality that is not achieved in Relational Databases. For a huge amount of data that is in peta-bytes, querying it and getting results in seconds is important. And Hive does this quite efficiently, it processes the queries fast and produce results in second's time.

Let see now what makes Hive so fast.

Some key differences between Hive and relational databases are the following;

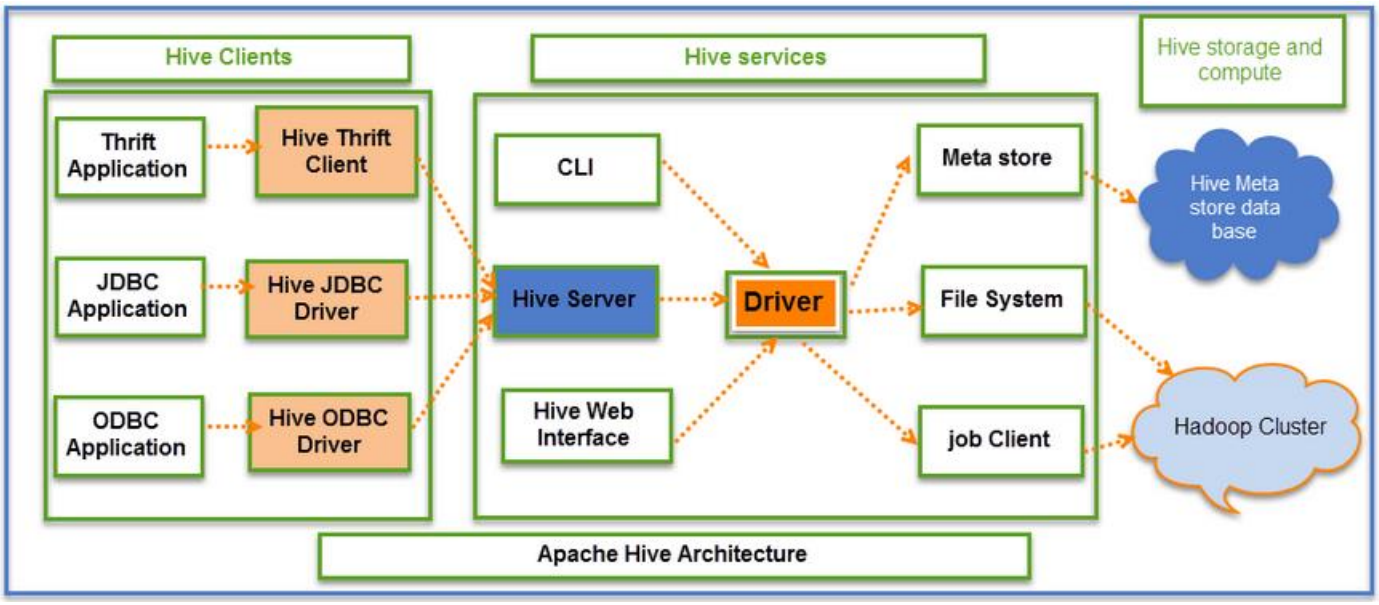
Relational databases are of "Schema on READ and Schema on Write". First creating a table then inserting data into the particular table. On relational database tables, functions like Insertions, Updates, and Modifications can be performed.

Hive is "Schema on READ only". So, functions like the update, modifications, etc. don't work with this. Because the Hive query in a typical cluster runs on multiple Data Nodes. So it is not possible to update and modify data across multiple nodes.(Hive versions below 0.13)

Also, Hive supports "READ Many WRITE Once" pattern. Which means that after inserting table we can update the table in the latest Hive versions.

NOTE: However the new version of Hive comes with updated features. Hive versions (Hive 0.14) comes up with Update and Delete options as new features.

HIVE ARCHITECTURE



As shown in the above image, the Hive Architecture can be categorized into the following components:

- Hive Clients: Hive supports application written in many languages like Java, C++, Python etc. using JDBC, Thrift and ODBC drivers. Hence one can always write hive client application written in a language of their choice.
- Hive Services: Apache Hive provides various services like CLI, Web Interface etc. to perform queries. We will explore each one of them shortly in this Hive tutorial blog.
- Processing framework and Resource Management: Internally, Hive uses Hadoop MapReduce framework as de facto engine to execute the queries. Hadoop MapReduce framework is a separate topic in itself and therefore, is not discussed here.
- Distributed Storage: As Hive is installed on top of Hadoop, it uses the underlying HDFS for the distributed storage. You can refer to the HDFS blog to learn more about it.

Now, let us explore the first two major components in the Hive Architecture:

1. Hive Clients:

Apache Hive supports different types of client applications for performing queries on the Hive. These clients can be categorized into three types:

- Thrift Clients: As Hive server is based on Apache Thrift, it can serve the request from all those programming language that supports Thrift.

- JDBC Clients: Hive allows Java applications to connect to it using the JDBC driver which is defined in the class org.apache.hadoop.hive.jdbc.HiveDriver.
- ODBC Clients: The Hive ODBC Driver allows applications that support the ODBC protocol to connect to Hive. (Like the JDBC driver, the ODBC driver uses Thrift to communicate with the Hive server.)

2. Hive Services:

Hive provides many services as shown in the image above. Let us have a look at each of them:

- Hive CLI (Command Line Interface): This is the default shell provided by the Hive where you can execute your Hive queries and commands directly.
- Apache Hive Web Interfaces: Apart from the command line interface, Hive also provides a web based GUI for executing Hive queries and commands.
- Hive Server: Hive server is built on Apache Thrift and therefore, is also referred as Thrift Server that allows different clients to submit requests to Hive and retrieve the final result.
- Apache Hive Driver: It is responsible for receiving the queries submitted through the CLI, the web UI, Thrift, ODBC or JDBC interfaces by a client. Then, the driver passes the query to the compiler where parsing, type checking and semantic analysis takes place with the help of schema present in the metastore. In the next step, an optimized logical plan is generated in the form of a DAG (Directed Acyclic Graph) of map-reduce tasks and HDFS tasks. Finally, the execution engine executes these tasks in the order of their dependencies, using Hadoop.
- Metastore: You can think metastore as a central repository for storing all the Hive metadata information. Hive metadata includes various types of information like structure of tables and the partitions along with the column, column type, serializer and deserializer which is required for Read/Write operation on the data present in HDFS. The metastore comprises of two fundamental units:
    - ✓ A service that provides metastore access to other Hive services.
    - ✓ Disk storage for the metadata which is separate from HDFS storage.

Now, let us understand the different ways of implementing Hive metastore in the next section of this Hive Tutorial.

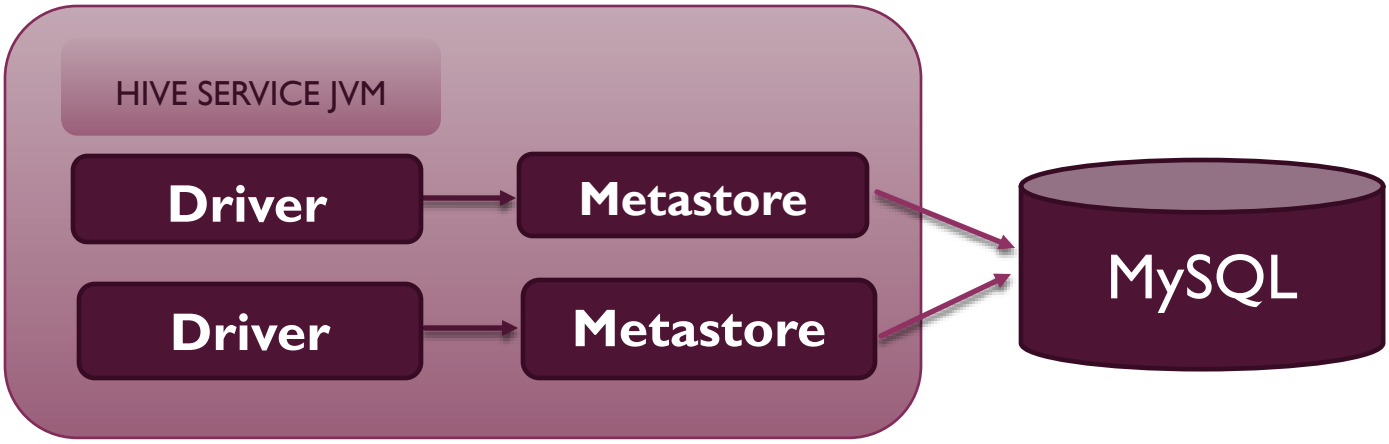
Metastore Configuration

Metastore stores the meta data information using RDBMS and an open source ORM (Object Relational Model) layer called Data Nucleus which converts the object representation into relational schema and vice versa. The reason for choosing RDBMS instead of HDFS is to achieve low latency. We can implement metastore in following three configurations:

1. Embedded Metastore:

metastore database at a time. If you start a second instance of Hive driver, you will get an error. This is good for unit testing, but not for the practical solutions.

2. Local Metastore:

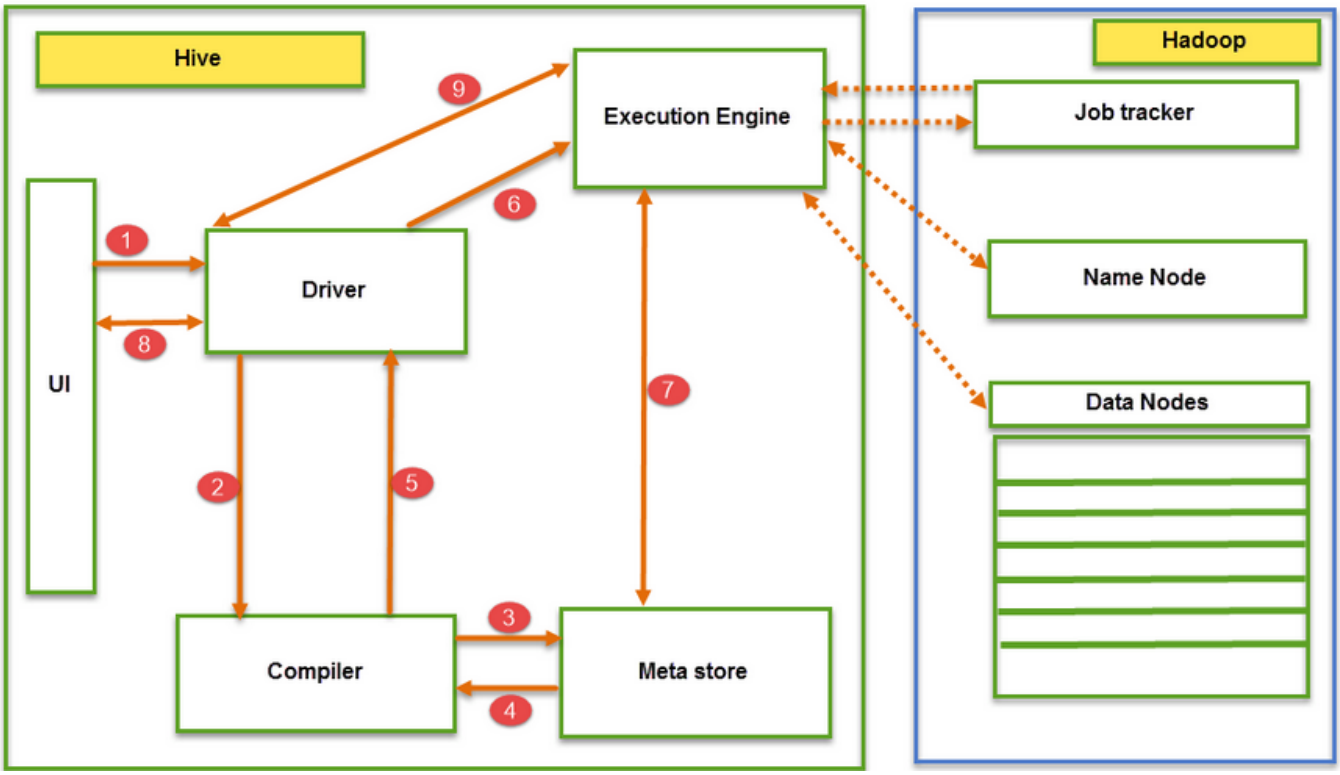


This configuration allows us to have multiple Hive sessions i.e. Multiple users can use the metastore database at the same time. This is achieved by using any JDBC compliant database like MySQL which runs in a separate JVM or a different machine than that of the Hive service and metastore service which are running in the same JVM as shown above. In general, the most popular choice is to implement a MySQL server as the metastore database.

3. Remote Metastore:

In the remote metastore configuration, the metastore service runs on its own separate JVM and not inthe Hive service JVM. Other processes communicate with the metastore server using Thrift Network APIs. You can have one or more metastore servers in this case to provide more availability. The main advantage of using remote metastore is you do not need to share JDBC login credential with each Hive user to access the metastore database.

Job Execution inside Hive



From the above screenshot we can understand the Job execution flow in Hive with Hadoop

The data flow in Hive behaves in the following pattern:-

1. Executing Query from the UI (User Interface)
2. The driver is interacting with Compiler for getting the plan. (Here plan refers to query execution) process and its related metadata information gathering
3. The compiler creates the plan for a job to be executed. Compiler communicating with Meta store for getting metadata request
4. Meta store sends metadata information back to compiler

5. Compiler communicating with Driver with the proposed plan to execute the query
6. Driver sending execution plans to Execution engine
7. Execution Engine (EE) acts as a bridge between Hive and Hadoop to process the query. For DFS operations.
    - EE should first contacts Name Node and then to Data nodes to get the values stored in tables.
    - EE is going to fetch desired records from Data Nodes. The actual data of tables resides in data node only. While from Name Node it only fetches the metadata information for the query.
    - It collects actual data from data nodes related to mentioned query
    - Execution Engine (EE) communicates bi-directionally with Meta store present in Hive to perform DDL (Data Definition Language) operations. Here DDL operations like CREATE, DROP and ALTERING tables and databases are done. Meta store will store information about database name, table names and column names only. It will fetch data related to query mentioned.
    - Execution Engine (EE) in turn communicates with Hadoop daemons such as Name node, Data nodes, and job tracker to execute the query on top of Hadoop file system

8. Fetching results from driver
9. Sending results to Execution engine. Once the results fetched from data nodes to the EE, it will send results back to driver and to UI (front end)

Hive Continuously in contact with Hadoop file system and its daemons via Execution engine. The dotted arrow in the Job flow diagram shows the Execution engine communication with Hadoop daemons.

Different modes of Hive

Hive can operate in two modes depending on the size of data nodes in Hadoop.

These modes are,

1. Local mode
2. Map reduce mode

When to use Local mode:

- If the Hadoop installed under pseudo mode with having one data node we use Hive in this mode
- If the data size is smaller in term of limited to single local machine, we can use this mode

- Processing will be very fast on smaller data sets present in the local machine

When to use Map reduce mode:

- If Hadoop is having multiple data nodes and data is distributed across different node we use Hive in this mode
- It will perform on large amount of data sets and query going to execute in parallel way
- Processing of large data sets with better performance can be achieved through this mode

In Hive, we can set this property to mention which mode Hive can work? By default, it works on Map Reduce mode and for local mode you can have the following setting.

Hive to work in local mode set

SET mapred.job.tracker=local;

From the Hive version 0.7 it supports a mode to run map reduce jobs in local mode automatically.

Hive Data Modeling

In Hive data modeling - Tables, Partitions and Buckets come in to picture.

Coming to Tables, it's just like the way that we create a table in Traditional relational databases. The functionalities such as filtering, joins can be performed on the tables. Hive deals with two types of table structures - Internal and External, depends on the design of schema and how the data is getting loaded in to Hive.

Internal Table is tightly coupled with nature. At first, we have to create tables and load the data. We can call this one as data on the schema. By dropping this table, both data and schema will be removed. The stored location of this table will be at "/user/hive/warehouse".

External Table is loosely coupled with nature. Data will be available in HDFS; the table is going to get created with HDFS data. We can say that it's creating schema of data. At the time of dropping the table, it dropped only schema, data will be available in HDFS as before. External tables provide an option to create multiple schemas for the data stored in HDFS instead of deleting the data every time whenever schema updates.

Partitions

Partitions come into place, when table is having one or more Partition keys which is the basis for determining how the data is stored. For Example: - "Client has Some E–commerce data which belong to India operations in which each state (29 states) operations mentioned in as a whole. If we take the state as partition key and perform partitions on that India data as a whole, we will be able to get a Number of partitions (29 partitions) which is equal to the number of states (29) present in India. Each state data can be viewed separately in the partition tables."

Buckets

Buckets are used for efficient querying. The data, i.e. present in that partition can be divided further into buckets. The division is performed based on hash of a particular column that we had selected in the table.

Summary:

Hive is an ETL and data warehouse tool on top of Hadoop ecosystem and used for processing structured and semi structured data.

- Hive is a database present in Hadoop ecosystem performs DDL and DML operations, and it provides flexible query language such as HQL for better querying and processing of data.
- It provides so many features compared to RDMS which has certain limitations.

For user specific logic to meet client requirements.

- It provides option of writing and deploying custom defined scripts and User defined functions.
- In addition, it provides partitions and buckets for storage specific logics.