HIVE

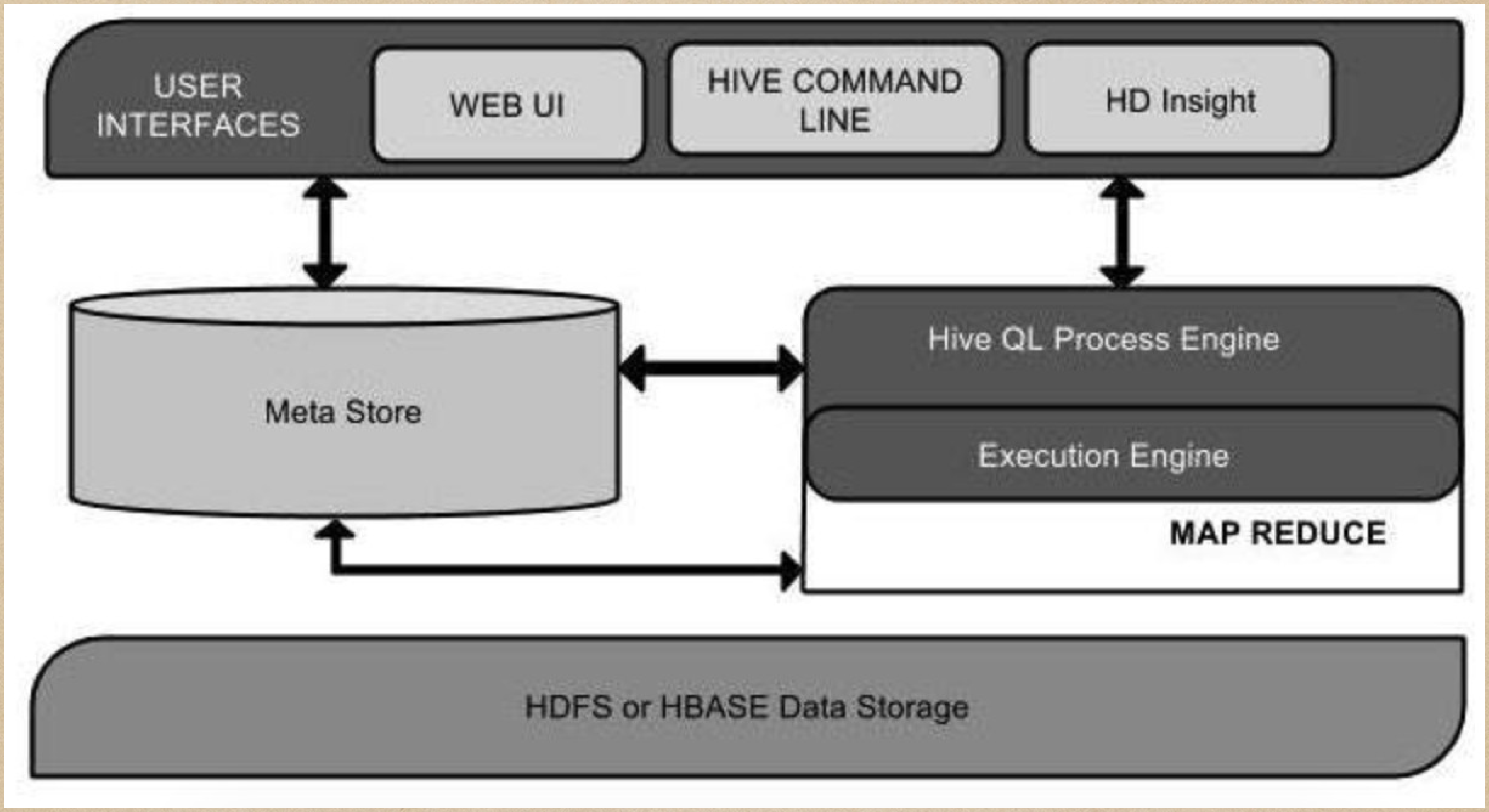
Abhinav Tyagi

What is Hive?

- Hive is a data warehouse infrastructure tool to process structure data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.
- Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.

Features of Hive

- It stores Schema in a database and processed data into HDFS(Hadoop Distributed File System).
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

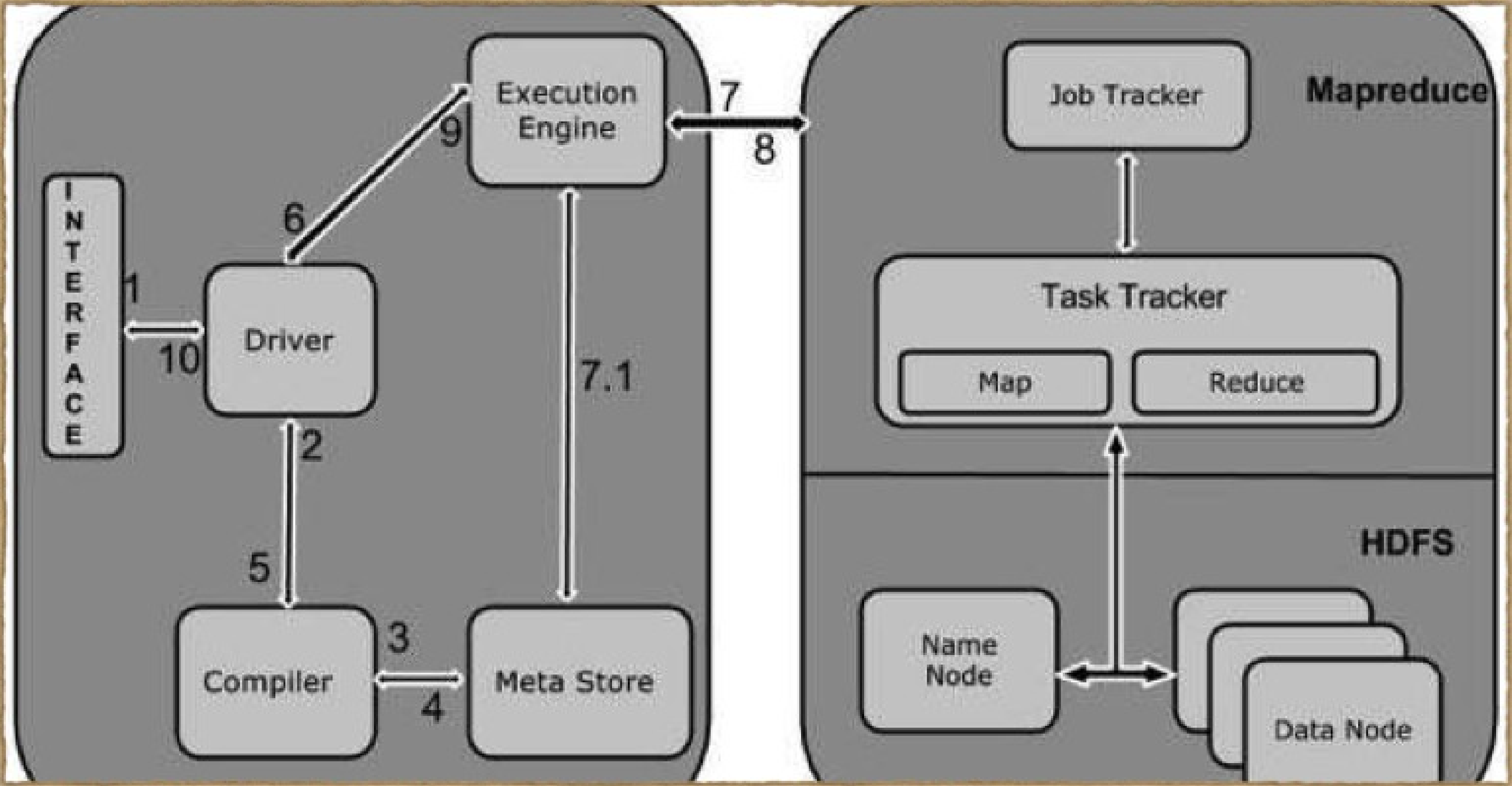


Architecture of Hive

Architecture of Hive

- User Interface - Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD.
- Meta Store -Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types and HDFS mapping.
- HiveQL Process Engine- HiveQL is similar to SQL for querying on schema info on the Megastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.

- Execution Engine - The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
- HDFS or HBASE - Hadoop distributed file system or HBASE are the data storage techniques to store data into the file system.



Working of Hive

Working of Hive

- Execute Query- The Hive interface such as Command Line or Web UI sends query Driver to execute.
- Get Plan- The driver takes the help of query complier that parses the query to check the syntax and query plan or the requirement of query.
- Get Metadata- The compiler sends metadata request to Megastore
- Send Metadata- Metastore sends metadata as a response to the compiler.

- Send Plan- The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
- Execute Plan- the driver sends the execute plan to the execution engine.
- Execute Job- Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.

- Metadata Ops- Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
- Fetch Result- The execution engine receives the results from Data nodes.
- Send Results- The execution engine sends those resultant values to the driver.
- Send Results- The driver sends the results to Hive Interfaces.

Hive- Data Types

All the data types in hive are classified into four types

- Column Types
- Literals
- Null Values
- Complex Types

Column Types

- Integral Types - Integer type data can be specified using integral data types, INT . When the data range exceeds the range of INT , you need to use BIGINT and if the data range is smaller than the INT , you use SMALLINT . TINYINT is smaller than SMALLINT .
- String Types - String type data types can be specified using single quotes (' ') or double quotes (" "). It contains two data types: VARCHAR and CHAR. Hive follows C-types escape characters.

- Timestamp - It supports traditional UNIX timestamp with optional nanosecond precision. It supports java.sql.Timestamp format "YYYY-MM-DD HH:MM:SS.fffffffff" and format "yyyy-mm-dd hh:mm:ss.ffffffffff".
- Dates - DATE values are described in year/month/day format in the form {{YYYY-MM-DD}}.
- Decimals -The DECIMAL type in Hive is as same as Big Decimal format of Java. It is used for representing immutable arbitrary precision.
- Union Types - Union is a collection of heterogeneous data types. You can create an instance using create union.

Literals

- Floating Point Types - Floating point types are nothing but numbers with decimal points. Generally, this type of data is composed of DOUBLE data type.
- Decimal Type - Decimal type data is nothing but floating point value with higher range than DOUBLE data type. The range of decimal type is approximately -10^{-308} to 10^{308}.

Complex Types

- Arrays - Arrays in Hive are used the same way they are used in Java.

Syntax: ARRAY<data_type>

- Maps - Maps in Hive are similar to Java Maps.

Syntax: MAP<primitive_type, data_type>

- Structs - Structs in Hive is similar to using complex data with comment.

Syntax: STRUCT<col_name : data_type [COMMENT col_comment, ...]>

Create Database

- hive> CREATE DATABASE [IF NOT EXISTS] userdb;
- hive> CREATE SCHEMA userdb;
- hive> SHOW DATABASES;

Drop Database

- hive>DROP DATABASE [IF EXISTS] userdb;
- hive> DROP DATABASE [IF EXISTS] userdb CASCADE;
- hive> DROP SCHEMA userdb;

Create Table

```
•hive> CREATE TABLE IF NOT EXISTS employee(eid int, name String, salary String, destination String)
>COMMENT 'Employee details'
>ROW FORMAT DELIMITED
>FIELDS TERMINATED BY '\t'
>LINES TERMINATED BY '\n'
>STORED AS TEXTFILE;
```

Partition

- Hive organizes tables into partitions. It is a way of dividing a table into related parts based on the values of partitioned columns such as date, city, and department. Using partition, it is easy to query a portion of the data.
- Adding partition- Syntax - hive> ALTER TABLE employee ADD PARTITION(year ='2013') location '/2012/part2012';
- Dropping partition - Syntax - hive>ALTER TABLE employee DROP [IF EXISTS] PARTITION (year='2013');

```
id, name, dept, year
1, Mark, TP, 2012
2, Bob, HR, 2012
3, Sam,SC, 2013
4, Adam, SC, 2013
```

HiveQL - Select Where

- The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore.

✓

```
hive> SELECT * FROM employee
WHERE salary>30000;
```

HiveQL - Select Order By

- The ORDER BY clause is used to retrieve the details based on one column and sort the result set by ascending or descending order.
- hive> SELECT Id, Name, Dept FROM employee ORDER BY DEPT;

HiveQL - Select-Group By

- The GROUP BY clause is used to group all the records in a result set using a particular collection column. It is used to query a group of records.
- hive> SELECT Dept,count(*) FROM employee GROUP BY DEPT;

HiveQL - Select-Joins

- JOIN is a clause that is used for combining specific fields from two tables by using values common to each one. It is used to combine records from two or more tables in the database. It is more or less similar to SQL JOIN.
- There are different types of joins given as follows:
- JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

JOIN

- JOIN clause is used to combine and retrieve the records from multiple tables. JOIN is same as OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign keys of the tables.

- hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT FROM CUSTOMERS c JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);

Table

Left Outer Join

- The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.
- hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c LEFT OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);

Right Outer Join

- The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.

- hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c RIGHT OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);

Full Outer Join

- The HiveQL FULL OUTER JOIN combines the records of both the left and the right outer tables that fulfill the JOIN condition. The joined table contains either all the records from both the tables, or fills in NULL values for missing matches on either side.
- hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c FULL OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);

Thank You