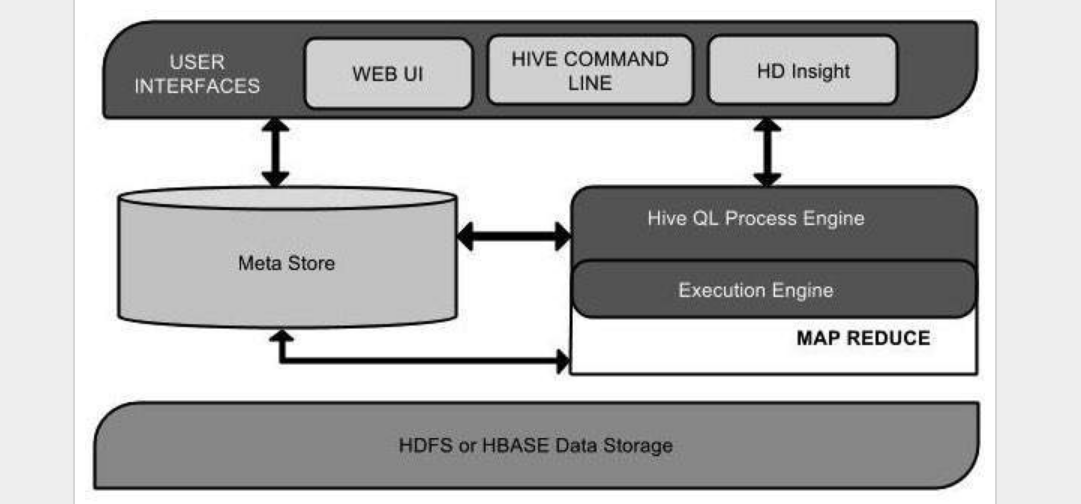
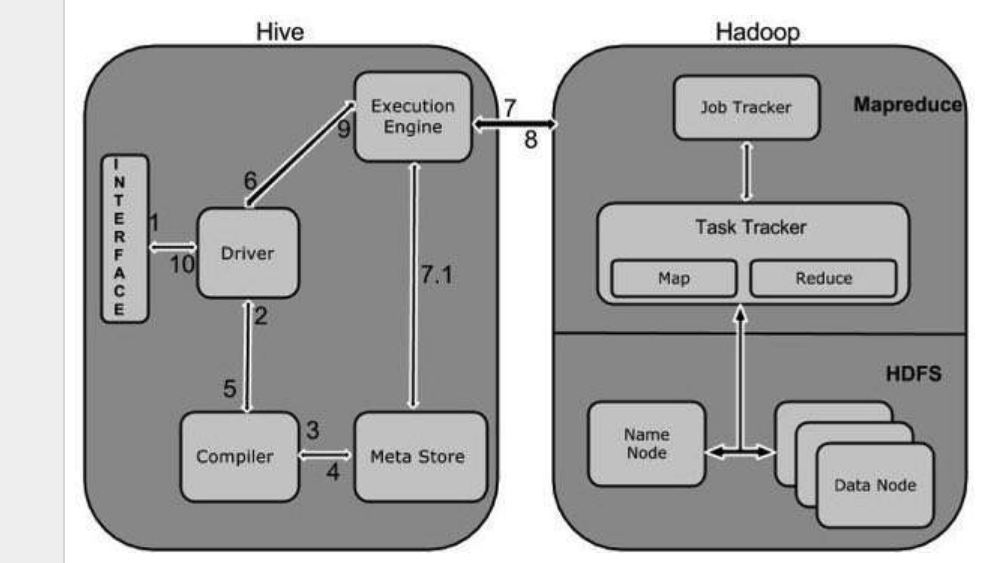
**Hive Architecture in brief:**

The following component diagram depicts the architecture of Hive:



|  |  |
| --- | --- |
| **Unit Name** | **Operation** |
| 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 Insight (In Windows server). |
| 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 Metastore. 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 file system |

The following diagram depicts the workflow between Hive and Hadoop.



|  |  |
| --- | --- |
| 1 | **Execute Query**  The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute. |
| 2 | **Get Plan**  The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query. |
| 3 | **Get Metadata**  The compiler sends metadata request to Metastore (any database). |
| 4 | **Send Metadata**  Metastore sends metadata as a response to the compiler. |
| 5 | **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. |
| 6 | **Execute Plan**  The driver sends the execute plan to the execution engine. |
| 7 | **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. |
| 7.1 | **Metadata Ops**  Meanwhile in execution, the execution engine can execute metadata operations with Metastore. |
| 8 | **Fetch Result**  The execution engine receives the results from Data nodes. |
| 9 | **Send Results**  The execution engine sends those resultant values to the driver. |
| 10 | **Send Results**  The driver sends the results to Hive Interfaces. |

**Hive Components in brief:**

**Hive Clients** – Apache Hive supports all application written in languages like C++, Java, Python etc. using JDBC, Thrift and ODBC drivers. Thus, one can easily write Hive client application written in a language of their choice.

The Hive supports different types of client applications for performing queries. These clients are categorized into 3 types:

**Thrift Clients** – As Apache Hive server is based on Thrift, so it can serve the request from all those languages that support Thrift.

**JDBC Clients** – Apache Hive allows Java applications to connect to it using JDBC driver. It is defined in the class apache.hadoop.hive.jdbc.HiveDriver.

**ODBC Clients** – ODBC Driver allows applications that support ODBC protocol to connect to Hive. For example JDBC driver, ODBC uses Thrift to communicate with the Hive server.

**Hive Services** – Hive provides various services like web Interface, CLI etc. to perform queries.

a) CLI(Command Line Interface) – This is the default shell that Hive provides, in which you can execute your Hive queries and command directly.

b) Web Interface – Hive also provides web based GUI for executing Hive queries and commands.

c) Hive Server – It is built on Apache Thrift and thus is also called as Thrift server. It allows different clients to submit requests to Hive and retrieve the final result.

d) Hive Driver – Driver is responsible for receiving the queries submitted Thrift, JDBC, ODBC, CLI, Web UL interface by a Hive client.

Complier –After that hive 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.

Optimizer – It generates the optimized logical plan in the form of a DAG (Directed Acyclic Graph) of MapReduce and HDFS tasks.

Executor – Once compilation and optimization complete, execution engine executes these tasks in the order of their dependencies using Hadoop.

e) Metastore – [Metastore](http://data-flair.training/blogs/apache-hive-metastore/) is the central repository of Apache Hive metadata in the Hive Architecture. It stores metadata for Hive tables (like their schema and location) and partitions in a relational database. It provides client access to this information by using metastore service API. Hive metastore consists of two fundamental units:

A service that provides metastore access to other Apache Hive services.

Disk storage for the Hive metadata which is separate from [HDFS](http://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/) storage.

Processing framework and Resource Management – Hive internally uses Hadoop MapReduce framework to execute the queries.

Distributed Storage – Hive is built on the top of Hadoop, so it uses the underlying HDFS for the distributed storage.