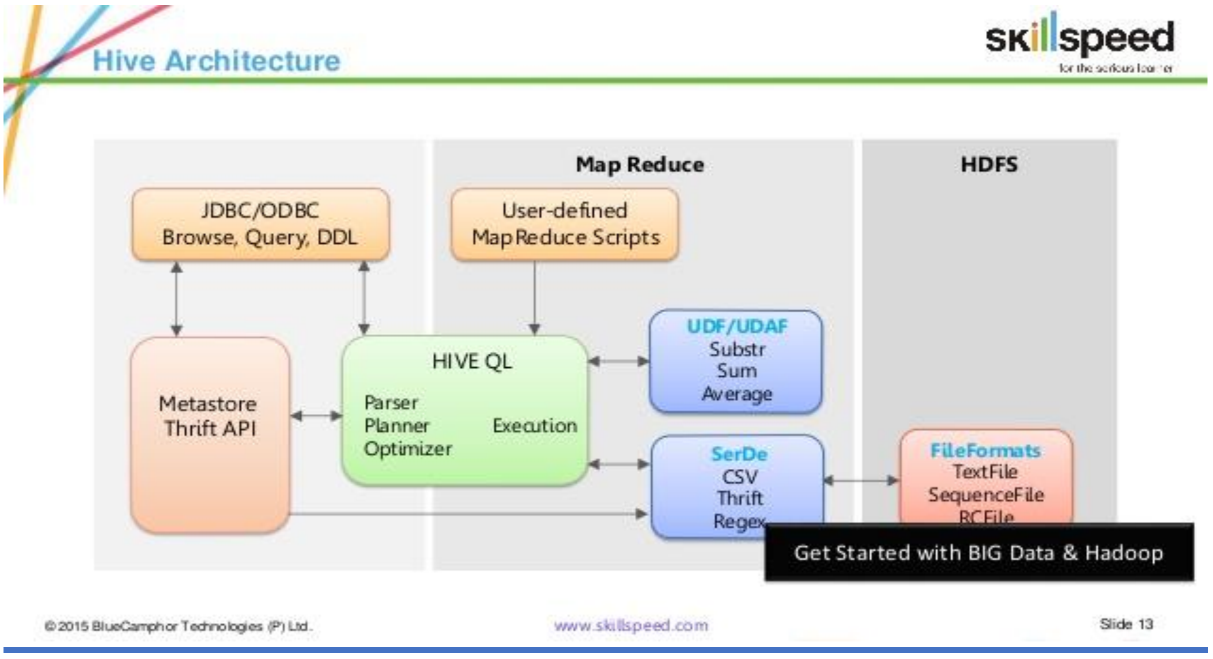
HIVE ARCHITECTURE



Hive Architecture
skillspeed
for the serious learner
Map Reduce
HDFS
JDBC/ODBC Browse, Query, DDL
User-defined MapReduce Scripts
UDF/UDAF Substr Sum Average
Metastore Thrift API
HIVE QL
Parser Planner Optimizer
Execution
SerDe CSV Thrift Regex
FileFormats TextFile SequenceFile RCFile
Get Started with BIG Data & Hadoop
©2015 BlueCamphor Technologies (P) Ltd.
www.skillspeed.com
Slide 13

Above diagram indicates the Hadoop Hive Architecture.

- When User comes from Command Line Interface (Hive Terminal), it directly gets connected to Hive Drivers.
- When User comes via JDBC/ODBC, it gets connected to Hive Drivers via API (Thrift Server).
- When User comes with Web GUI, it directly gets connected to Hive Drivers.
- Hive Driver receives tasks from User and transfer it to HADOOP architecture.
- HADOOP uses MAPREDUCE architecture for performing tasks or queries by user.

HIVE EXECUTION FLOW

1) UI calls the execute interface to the Hive Driver.

2) Driver creates a session handle for the query.

3) Driver sends the query to the compiler to generate an execution plan.

4) Since compiler needs metadata, a request is being sent to METASTORE via getMetaData method and receives the metadata via sendMetaData method.

5) This metadata is used to type check the expressions in the query tree as well as to prune partitions based on query predicates.

6) Compiler generates the execution plan which is a DAG (DIRECTED ACYCLIC GRAPH) of stages.

7) The DAG divides operators to MapReduce stages and tasks based on the input query and data.

8) Hive Execution Engine submits this DAG to appropriate components (Job tracker, Task tracker and Name Nodes).

9) In each task (mapper/reducer), the deserializer associated with the table or intermediate outputs, is used to read the rows from HDFS files and these are passed through the associated operator tree.

10) Once the output gets generated, it is written to a temporary HDFS file through the serializer.

11) The temporary files are used to provide the data to subsequent map/reduce stages of the plan.

12) For DML operations the final temporary file is moved to the table’s location.

13) For queries, the contents of the temporary file are read by the execution engine directly from HDFS as part of the fetch call from the Driver.

HIVE COMPONENTS

PFB the major components of Hive

- METASTORE
- HIVE DRIVER
- COMPILER
- OPTIMIZER
- EXECUTION ENGINE
- CLI, UI, API THRIFT SERVER
- SERDE

USER INTERFACE:

The UI is the User Interface. The User submits queries and requests to Hive from the User Interface.

COMMAND LINE INTERFACE:

A command-line interface (CLI) provides a user interface for an external user to interact with Hive by submitting queries, instructions and monitoring the process status.

API THRIFT SERVER:

Thrift server allows external clients to interact with Hive over a network, similar to the JDBC or ODBC protocols.

HIVE DRIVER:

UI, GUI and API send queries to the driver. The driver creates a session handle for the query and sends the query to the compiler to generate an execution plan.

COMPILER:

The compiler parses the query which is received from Driver. It performs semantic analysis on the different query blocks and query expressions. It generates an execution plan with the help of metadata received from METASTORE.

OPTIMIZER:

Optimizations on DAG, produced by COMPILER, are performed by OPTIMIZER. It facilitates better performance and scalability to DAG.

METASTORE:

Metastore stores all the structural information about the tables, partitions. DESCRIBE HQL command returns the METADATA OF the Table which is stored in METASTORE. It stores the SERDE information corresponding to each HIVE Table which is required to read and write data. It stores the information about the corresponding HDFS FILES where data is stored.

EXECUTION ENGINE:

The execution engine executes the execution plan created by COMPILER. It manages the dependencies between these different stages of the plan. It executes these stages on the appropriate system components.

SERDE:

SERDE interface is a combination of a serializer and deserializer, which allows you to instruct Hive for processing of records. SERIALIZER takes JAVA Object and turns into serialized objects that HIVE can write to HDFS. On the other hand, DESERIALIZER takes data from records from HDFS and converts it into JAVA Object.