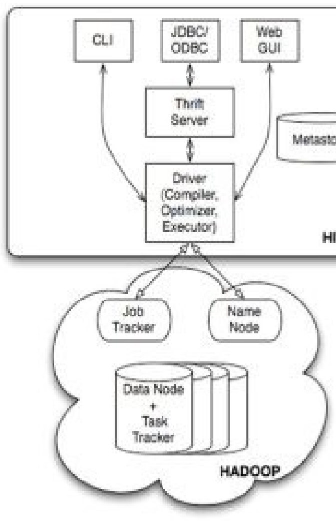
ASSIGNMENT 6.3

1) **Explain Hive Architecture in Brief.**

Hadoop Hive Architecture

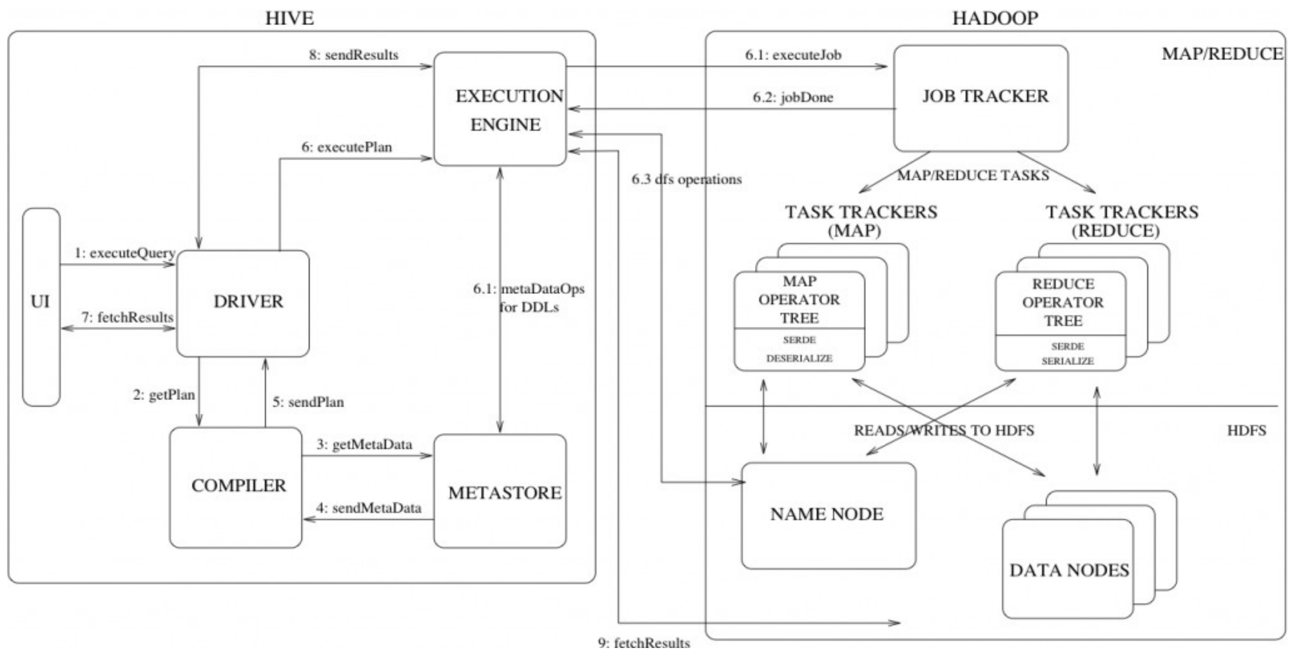
Hive is one of the most important component of Hadoop.



The above diagram shows the basic Hadoop Hive architecture. Primarily The diagram represents CLI (Command Line Interface),JDBC/ODBC and Web GUI (Web Graphical User Interface).This represents when user comes with CLI(Hive Terminal) it directly connected to Hive Drivers,When User comes with JDBC/ODBC(JDBC Program) at that time by using API(Thrift Server) it connected to Hive driver and when the user comes with Web GUI(Ambari server) it directly connected to Hive Driver.

The hive driver receives the tasks(Queries) from user and send to Hadoop architecture.The Hadoop architecture uses name node,data node,job tracker and task tracker for receiving and dividing the work what Hive sends to Hadoop (mapreduce). .

The below diagram represents clear internal Hadoop Hive Architecture



The above diagram shows how a typical query flows through the system

Step 1 :- The UI calls the execute interface to the Driver

Step 2 :- The Driver creates a session handle for the query and sends the query to the compiler to generate an execution plan

Step 3&4 :- The compiler needs the metadata so send a request for getMetaData and receives the sendMetaData request from MetaStore.

Step 5 :- This metadata is used to typecheck the expressions in the query tree as well as to prune partitions based on query predicates. The plan generated by the compiler is a DAG of stages with each stage being either a map/reduce job, a metadata operation or an operation on HDFS. For map/reduce stages, the plan contains map operator trees (operator trees that are executed on the mappers) and a reduce operator tree (for operations that need reducers).

Step 6 :- The execution engine submits these stages to appropriate components (steps 6, 6.1, 6.2 and 6.3). 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.Once the output generate it is written to a temporary HDFS file though the serializer. The temporary files are used to provide the to

subsequent map/reduce stages of the plan.For DML operations the final temporary file is moved to the table's location

Step 7&8&9 :- 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.

2) Explain Hive Components in Brief.

<u>User Interface</u>

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).

<u>Meta Store</u>

Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.

The Metastore provides two important but often overlooked features of a data warehouse: data abstraction and data discovery. Without the data abstractions provided in Hive, a user has to provide information about data formats, extractors and loaders along with the query. In Hive, this information is given during table creation and reused every time the table is referenced. This is very similar to the traditional warehousing systems. The second functionality, data discovery, enables users to discover and explore relevant and specific data in the warehouse. Other tools can be built using this metadata to expose and possibly enhance the information about the data and its availability. Hive accomplishes both of these

features by providing a metadata repository that is tightly integrated with the Hive query processing system so that data and metadata are in sync.

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.

HiveQL is an SQL-like query language for Hive. It mostly mimics SQL syntax for creation of tables, loading data into tables and querying the tables. HiveQL also allows users to embed their custom map-reduce scripts. These scripts can be written in any language using a simple row-based streaming interface – read rows from standard input and write out rows to standard output. This flexibility comes at a cost of a performance hit caused by converting rows from and to strings. However, we have seen that users do not mind this given that they can implement their scripts in the language of their choice. Another feature unique to HiveQL is multi-table insert. In this construct, users can perform multiple queries on the same input data using a single HiveQL query. Hive optimizes these queries to share the scan of the input data, thus increasing the throughput of these queries several orders of magnitude.

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.

Compiler

Parser – Transform a query string to a parse tree representation.

Semantic Analyser – Transform the parse tree to an internal query representation, which is still block based and not an operator tree. As part of this step, the column names are verified and expansions like * are

performed. Type-checking and any implicit type conversions are also performed at this stage. If the table under consideration is a partitioned table, which is the common scenario, all the expressions for that table are collected so that they can be later used to prune the partitions which are not needed. If the query has specified sampling, that is also collected to be used later on.

Logical Plan Generator – Convert the internal query representation to a logical plan, which consists of a tree of operators. Some of the operators are relational algebra operators like 'filter', 'join' etc. But some of the operators are Hive specific and are used later on to convert this plan into a series of map-reduce jobs. One such operator is a reduceSink operator which occurs at the map-reduce boundary. This step also includes the optimizer to transform the plan to improve performance – some of those transformations include: converting a series of joins into a single multi-way join, performing a map-side partial aggregation for a group-by, performing a group-by in 2 stages to avoid the scenario when a single reducer can become a bottleneck in presence of skewed data for the grouping key. Each operator comprises a descriptor which is a serializable object.

Query Plan Generator – Convert the logical plan to a series of map-reduce tasks. The operator tree is recursively traversed, to be broken up into a series of map-reduce serializable tasks which can be submitted later on to the map-reduce framework for the Hadoop distributed file system. The reduceSink operator is the map-reduce boundary, whose descriptor contains the reduction keys. The reduction keys in the reduceSink descriptor are used as the reduction keys in the map-reduce boundary. The plan consists of the required samples/partitions if the query specified so. The plan is serialized and written to a file.

HDFS or HBASE

Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.