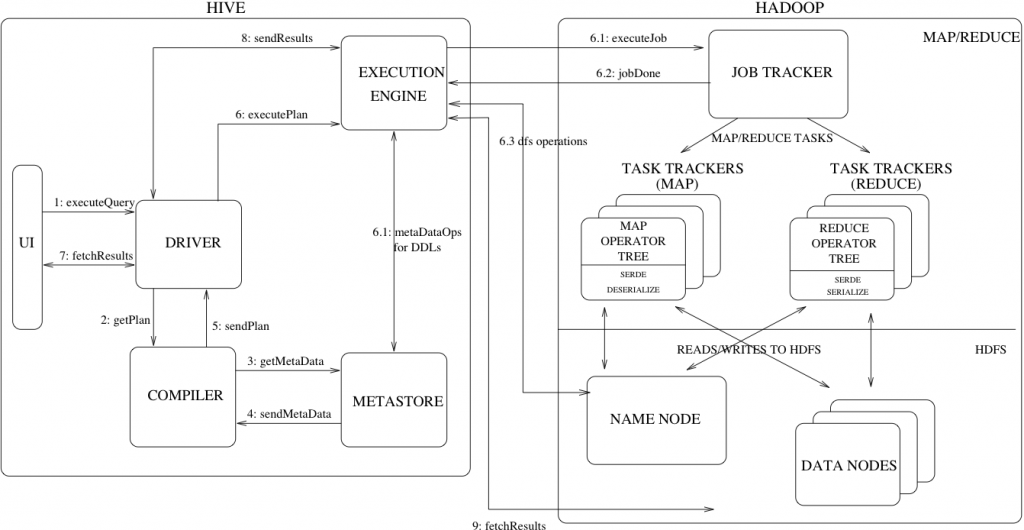


The above diagram shows the basic **Hadoop Hive architecture**. Primarily The diagram represents CLI (Command Line Interface),JDBC/ODBC and Web GUI (Web Graphical User Interface ).This represents when user comes with CLI(Hive Terminal) it directly connected to Hive Drivers, When User comes with JDBC/ODBC(JDBC 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 Architecture**](http://www.hadooptpoint.com/hadoop-mapreduce/)) .

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