This assignment has two questions – one on architecture and other the components. I have combined the two answers below since one cannot be explained without the other and if both explained separately a lot of content is repeated.

Hive

* A data warehouse solution built on top of Hadoop developed by Facebook.
* provides an SQL (Structured Query Language) dialect (called as Hive Query Language)
* queries the data underlying in HDFS
* uses HDFS for storage but stores all metadata information about database an table locally to Hive
* most suited for data warehouse applications, where relatively static data is analyzed, fast response times are not required, and when the data is not changing rapidly.
* It is an abstraction over Map-Reduce.
* The operations/functions are internally translated to MapReduce
* It reduces development time and effort

Before getting into the architecture, just a look at the main components

**UI**

* The user interface for users to submit queries and other operations to the system.
* Eg: command line interface and a web based GUI.

**Driver**

* The component which receives the queries.
* This component implements the notion of session handles and provides execute and fetch APIs modeled on JDBC/ODBC interfaces.

**Compiler**

* The component that parses the query, does semantic analysis on the different query blocks and query expressions and eventually generates an execution plan with the help of the table and partition metadata looked up from the metastore.

**Metastore**

* The component that stores all the structure information of the various tables and partitions in the warehouse including column and column type information, the serializers and deserializers necessary to read and write data and the corresponding HDFS files where the data is stored.

**Execution Engine**

* The component which executes the execution plan created by the compiler. The plan is a DAG of stages. The execution engine manages the dependencies between these different stages of the plan and executes these stages on the appropriate system components.

**Hive server**

* Provides a thrift interface and a JDBC/ODBC server and a way of integrating Hive with other applications.

**Extensibility Interfaces**

* Is like the Command Line Interface (CLI), the web UI and JDBC/ODBC driver. It includes SerDe as well as the UDF(User Defined Function) and UDAF(User Defined Aggregate Function) interfaces that enable users to define their own custom functions.

**The Hive internal architecture**



How it works

**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.

**Step 6**

The execution engine submits these stages to appropriate components.

Once the output generate it is written to a temporary HDFS file though the serializer.

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

Execution engines of Hive

* **MapReduce** is the default execution engine for Hive because Hive was written originally to use MapReduce.
* The execution engine is controlled by the hive.execution.engine property, which defaults to mr (for MapReduce).
* Hive can also be run on other execution engines like Tez, spark.