|  |
| --- |
| **Assignment 23.3** |
|  |

|  |
| --- |
|  |
| **HIVE ARCHITECTURE** |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| * The UI calls the execute interface to the Driver |
|  |

|  |
| --- |
| * The Driver creates a session handle for the query and sends the query to the compiler to generate an execution plan |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| * The compiler needs the metadata so send a request for get Meta Data and receives the send Meta Data request from Meta Store. |
|  |

|  |
| --- |
| * This metadata is used to type check 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). |
|  |

|  |
| --- |
| * 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 |
|  |

|  |
| --- |
| * 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** |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **UI** :- UI means User Interface, The user interface for users to submit queries and other operations to the system. |
|  |

|  |
| --- |
| **Driver** :- The Driver is used for receives the quires from UI .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. |
|  |