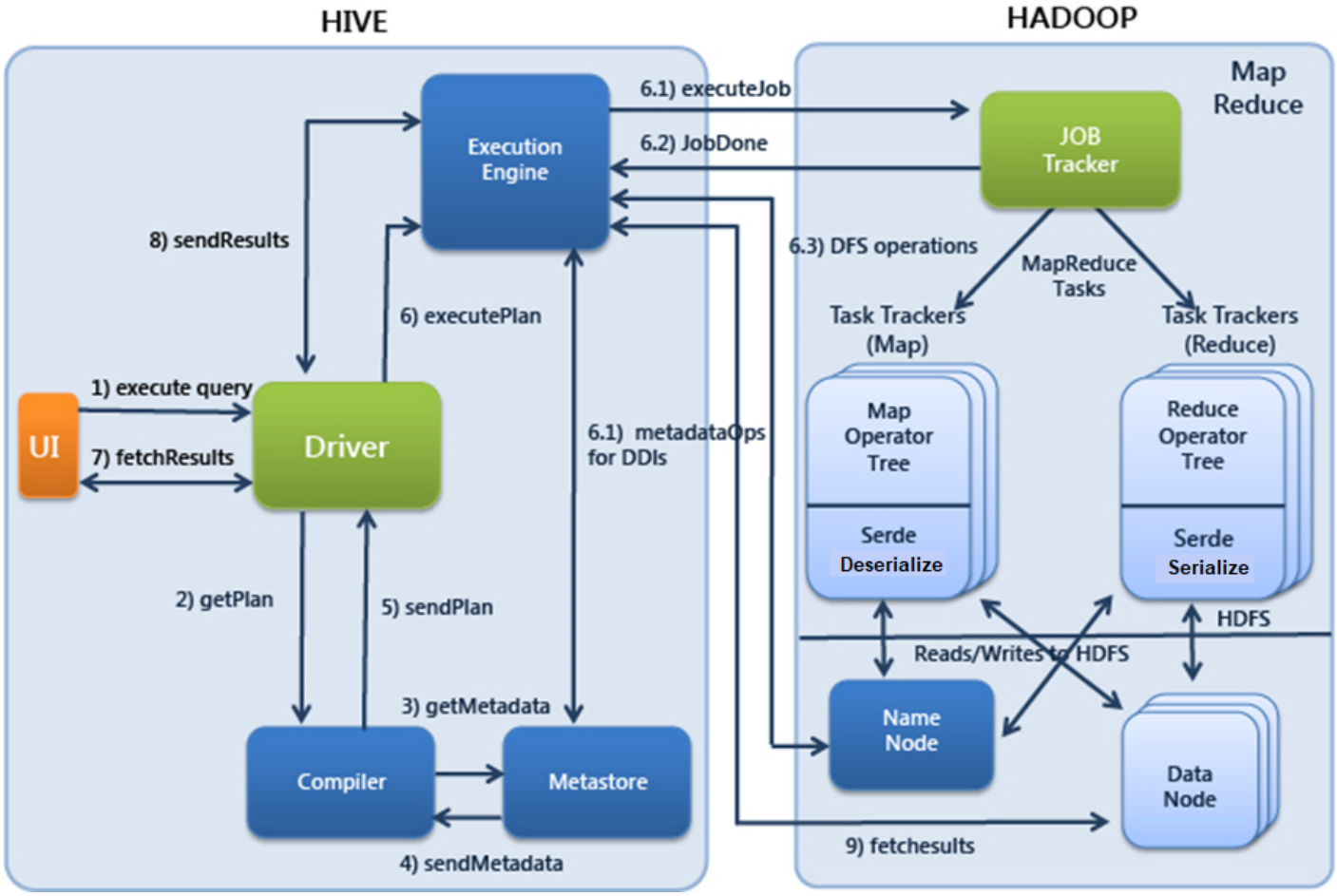
Initially there was only **Hive** - Just a client side application with an interactive shell. So it was not possible to use it from your regular SQL clients.

That's why later came **HiveServer 1** - it allows JDBC/ODBC connections from your regular SQL clinets like Tableue. But the problem here was lack of concurrency.

So, **HiveServer 2 + Beeline** came - it allows JDBC/ODBC and also provides concurrency. And it has a new client side command line tool called Beeline.

HiveServer 2 could be installed on the Hadoop Cluster or outside of the cluster on a client machine.

Hive In Progress



- **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 and so it sends 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 for intermediate outputs is used to read the rows from HDFS files and these are passed through the associated operator tree.
    - Once the output gets generated it is written to a temporary HDFS file though the serializer. The temporary files are used to provide the 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.

Example of Dynamic Partitions

```
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode=nonstrict;

CREATE TABLE employees (name STRING, salary FLOAT, dept STRING, state STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

LOAD DATA INPATH '/user/cloudera/input/employeeWithStates.txt' OVERWRITE INTO TABLE employees;

CREATE TABLE employees_partitioned (name STRING, salary FLOAT, dept STRING) PARTITIONED BY (state STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

INSERT INTO employees_partitioned partition (state) select * from employees;
```

MapReduce, Pig, Hive

- — MapReduce is a compiled language whereas Pig is a scripting language and Hive is a SQL like query language.
- — Pig and Hive provide higher level of abstraction whereas MapReduce provides low level of abstraction.
- — MapReduce requires more lines of code when compared to Pig and Hive. Hive requires very few lines of code when compared to Pig and MapReduce because of its SQL like resemblance.
- — Whenever a job requires implementing a custom partitioner, hadoop developers can choose MapReduce/Pig over Hive.
- — MapReduce requires more development effort than Pig and Hive.
- — Pig and Hive coding approaches are slower than a fully tuned Hadoop MapReduce program.
- — There is very limited possibility for the developer to write java level bugs when coding in Pig or Hive.

Pig Vs. Hive

1. Hive is used mainly by data analysts whereas Pig is generally used by Researchers and Programmers.
2. Hive is used for completely structured Data whereas Pig is used for semi structured data.
3. Hive has a declarative SQLish language (HiveQL) whereas Pig has a procedural data flow language (Pig Latin)
4. Hive is mainly used for creating reports whereas Pig is mainly used for programming.
5. You can join, order and sort data dynamically in an aggregated manner with Hive and Pig however Pig also provides you an additional COGROUP feature for performing outer joins.
6. Hive can start an optional thrift based server that can send queries from any nook and corner directly to the Hive server which will execute them whereas this feature is not available with Pig.
7. Hive directly leverages SQL expertise and thus can be learnt easily whereas Pig is also SQL-like but varies to a great extent and thus it will take some time and efforts to master Pig.
8. Hive makes use of exact variation of the SQL DLL language by defining the tables beforehand and storing the schema details in a local DB whereas in case of Pig, there is no dedicated metadata DB and the schemas or data types will be defined in the script itself.
9. Hive has a provision for partitions so that you can process the subset of data by date or in an alphabetical order whereas Pig does not have any notion for partitions though you can achieve this through filters.
10. Apache Pig is the most concise and compact language compared to Hive.