Introduction to Hive

A.Baskar

- The Hadoop ecosystem contains different sub-projects (tools) such as Sqoop, Pig, and Hive that are used to help Hadoop modules.
- Sqoop: It is used to import and export data to and from between HDFS and RDBMS.
- Pig: It is a procedural language platform used to develop a script for MapReduce operations.
- Hive: It is a platform used to develop SQL type scripts to do MapReduce operations.
- Note: There are various ways to execute MapReduce operations:
- The traditional approach using Java MapReduce program for structured, semi-structured, and unstructured data.
- The scripting approach for MapReduce to process structured and semi structured data using Pig.
- The Hive Query Language (HiveQL or HQL) for MapReduce to process structured data using Hive.

What is Hive?

- Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.
- Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.
- Apache Hive is an **open-source data warehousing tool** for performing distributed processing and data analysis.

Hive is not

- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

Features of Hive

- It stores schema in a database and processed data into HDFS.
- It is designed for **OnLine Analytical Processing (**OLAP).
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Hive suitable for

1.Hive suitable for DWH applications.
2. processes batch jobs on huge data that is immutable.
3. EG: Web logs, Application logs.

Hive history

2007 : Born at FACEBOOK to analyse incoming data

2008 : Hive became Apache Hadoop sub-project.

Hive versions:

Hive 0.10

1. Batch
2. Read only data
3. HiveQL
4. MR

Hive 0.13

1. Interactive
2. Read-only data
3. Substantial SQL
4. MR

Hive 0.14

1. Transactions with ACID semantics
2. Cost based optimizer
3. SQL temporary tables.

Features of Hive

1.It is similar to SQL.

2.HQL is easy to code.

3.Hive supports rich data types such as structs, lists, and maps.

4.Hive supports SQL filters, group-by and order-by clauses.

5.Custom Types, Custom functions can be defined.

Hive Data Units

Hive Data Units

- **Databases**: The namespace for tables

- **Tables** : Set of records that have similar schema

- **Partitions**: Logical separation of data based on classification of given information as per specific attributes. Once hive has partitioned the data based on specified key, it starts to assemble the records into specific folders as when the records are inserted.

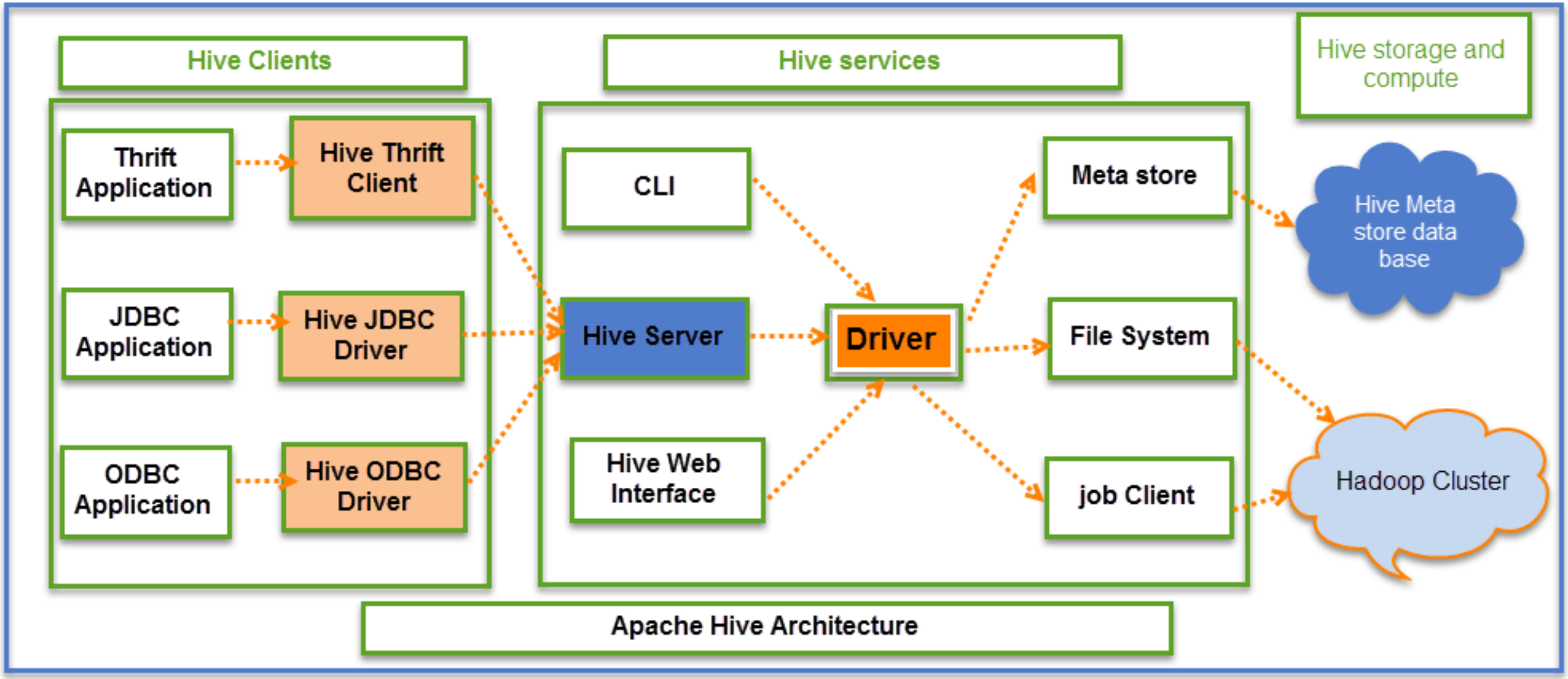
- **Buckets (Clusters**) : Similar to partition but uses hash function to segregate the data and determines the cluster or bucket into which the record should be placed.

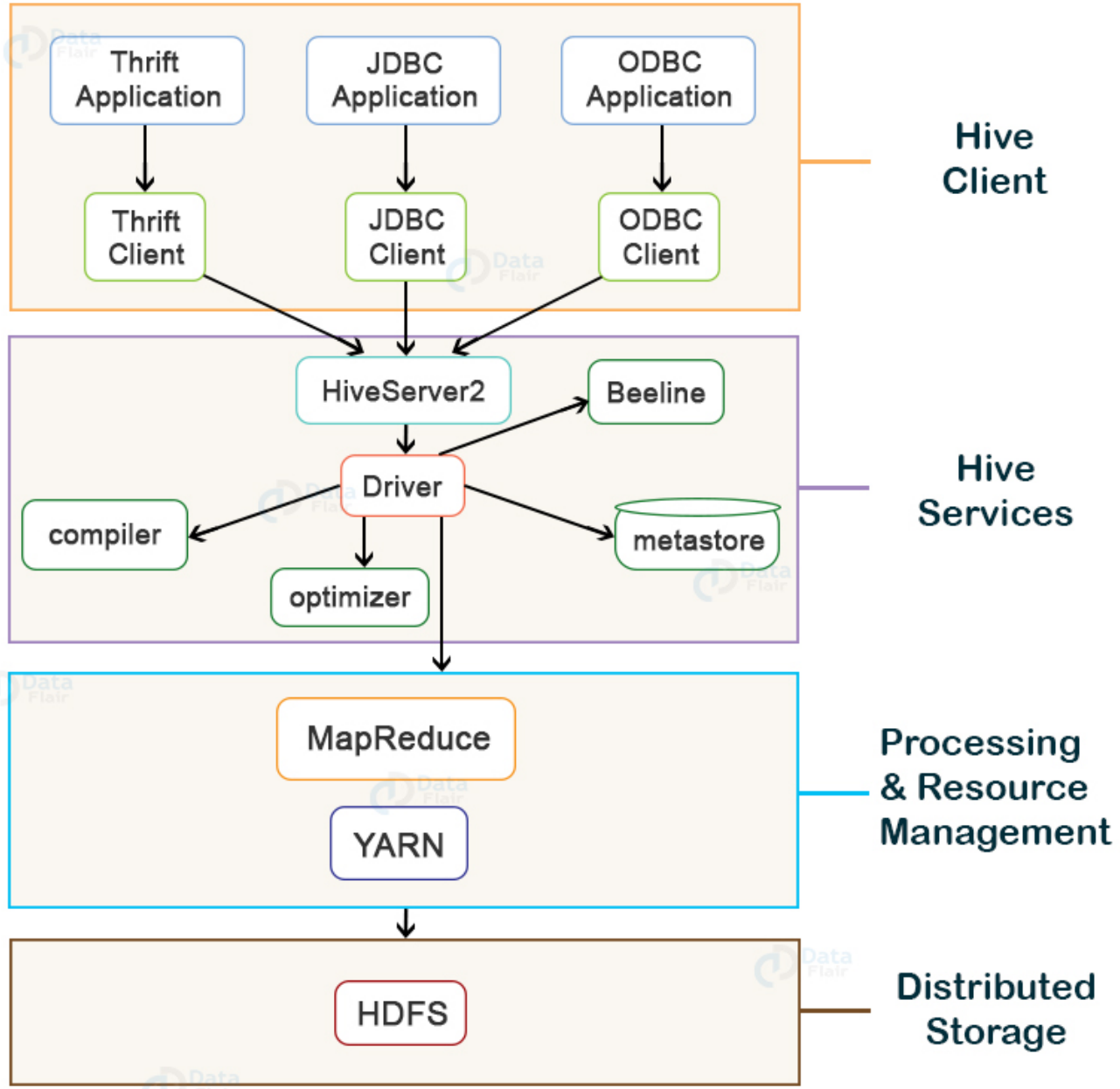
Semblance of hive structure with database

Database: Several tables, table constitute with rows and columns.

Hive:

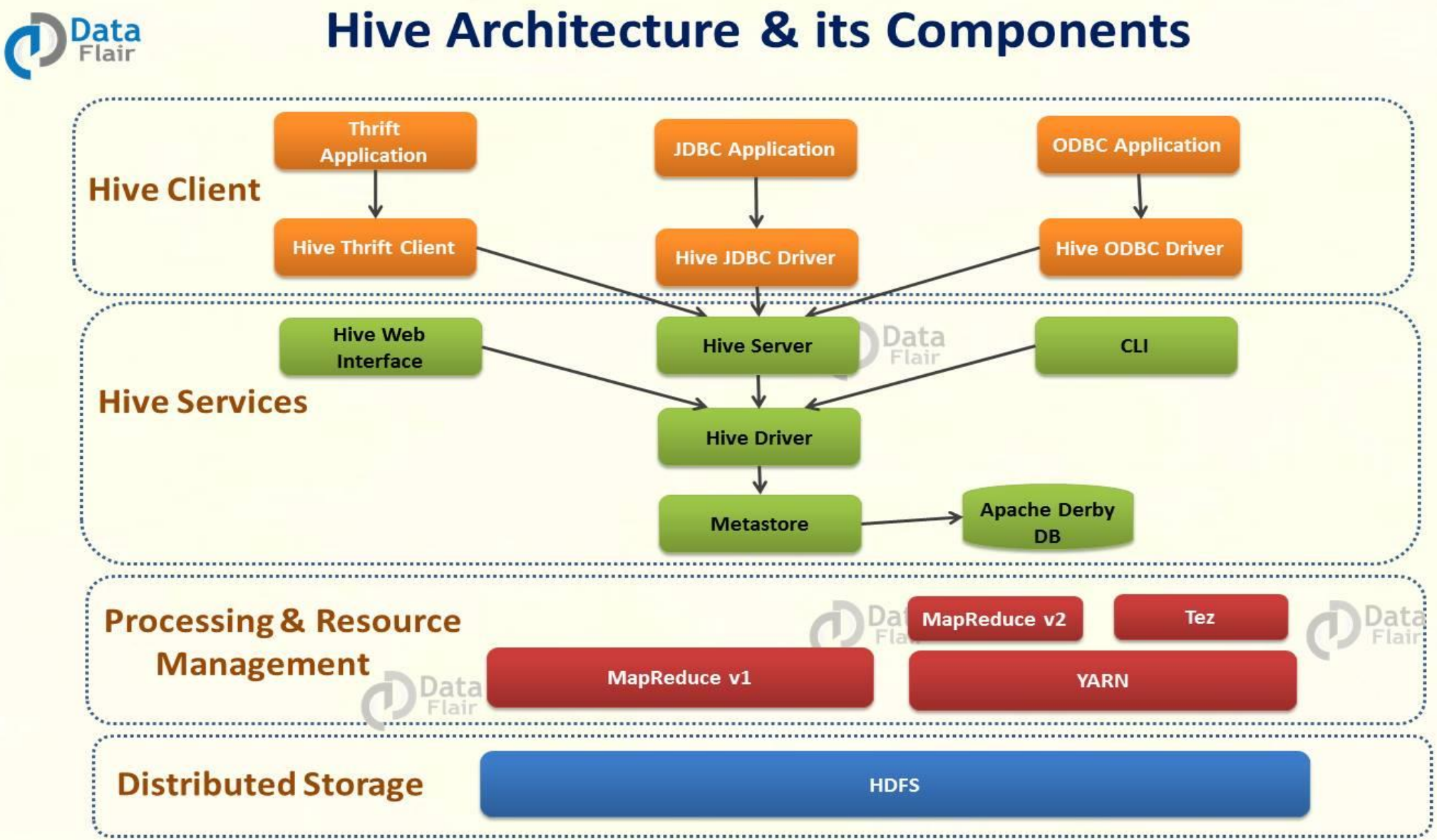
Tables :	Folder
Partition tables:	sub directory
Bucketed tables:	stored as files.


Hive Clients
Hive services
Hive storage and compute
Thrift Application
Hive Thrift Client
JDBC Application
Hive JDBC Driver
ODBC Application
Hive ODBC Driver
CLI
Hive Server
Hive Web Interface
Driver
Meta store
File System
job Client
Hive Meta store data base
Hadoop Cluster
Apache Hive Architecture



Hive Architecture & Its Components

Hive Architecture



- **Hive Client**
- Hive supports applications written in any language like Python, Java, C++, Ruby, etc. using JDBC, ODBC, and Thrift drivers, for performing queries on the Hive. Hence, one can easily write a hive client application in any language of its own choice.
- Hive provides different drivers for communication with a different type of applications. For Thrift based applications, it will provide Thrift client for communication.
- For Java related applications, it provides JDBC Drivers. Other than any type of applications provided ODBC drivers. These Clients and drivers in turn again communicate with Hive server in the Hive services.

- Hive clients are categorized into three types:

1. Thrift Clients

- The Hive server is based on Apache Thrift so that it can serve the request from a thrift client.

2. JDBC client

- Hive allows for the Java applications to connect to it using the JDBC driver. JDBC driver uses Thrift to communicate with the Hive Server.

3. ODBC client

- Hive ODBC driver allows applications based on the ODBC protocol to connect to Hive. Similar to the JDBC driver, the ODBC driver uses Thrift to communicate with the Hive Server.

- **2. Hive Server 2**
- HiveServer2 is the successor of HiveServer1. HiveServer2 enables clients to execute queries against the Hive. It allows multiple clients to submit requests to Hive and retrieve the final results. It is basically designed to provide the best support for open API clients like JDBC and ODBC.

Note: Hive server1, also called a Thrift server, is built on Apache Thrift protocol to handle the cross-platform communication with Hive. It allows different client applications to submit requests to Hive and retrieve the final results. It does not handle concurrent requests from more than one client due to which it was replaced by HiveServer2.

3. Hive Driver

- The Hive driver receives the **HiveQL** statements submitted by the user through the command shell. It creates the session handles for the query and sends the query to the compiler.

- **Hive Compiler**
- Hive compiler parses the query. It performs semantic analysis and type-checking on the different query blocks and query expressions by using the metadata stored in metastore and generates an execution plan.
- The execution plan created by the compiler is the **DAG(Directed Acyclic Graph)**, where each stage is a map/reduce job, operation on HDFS, a metadata operation.

Optimizer

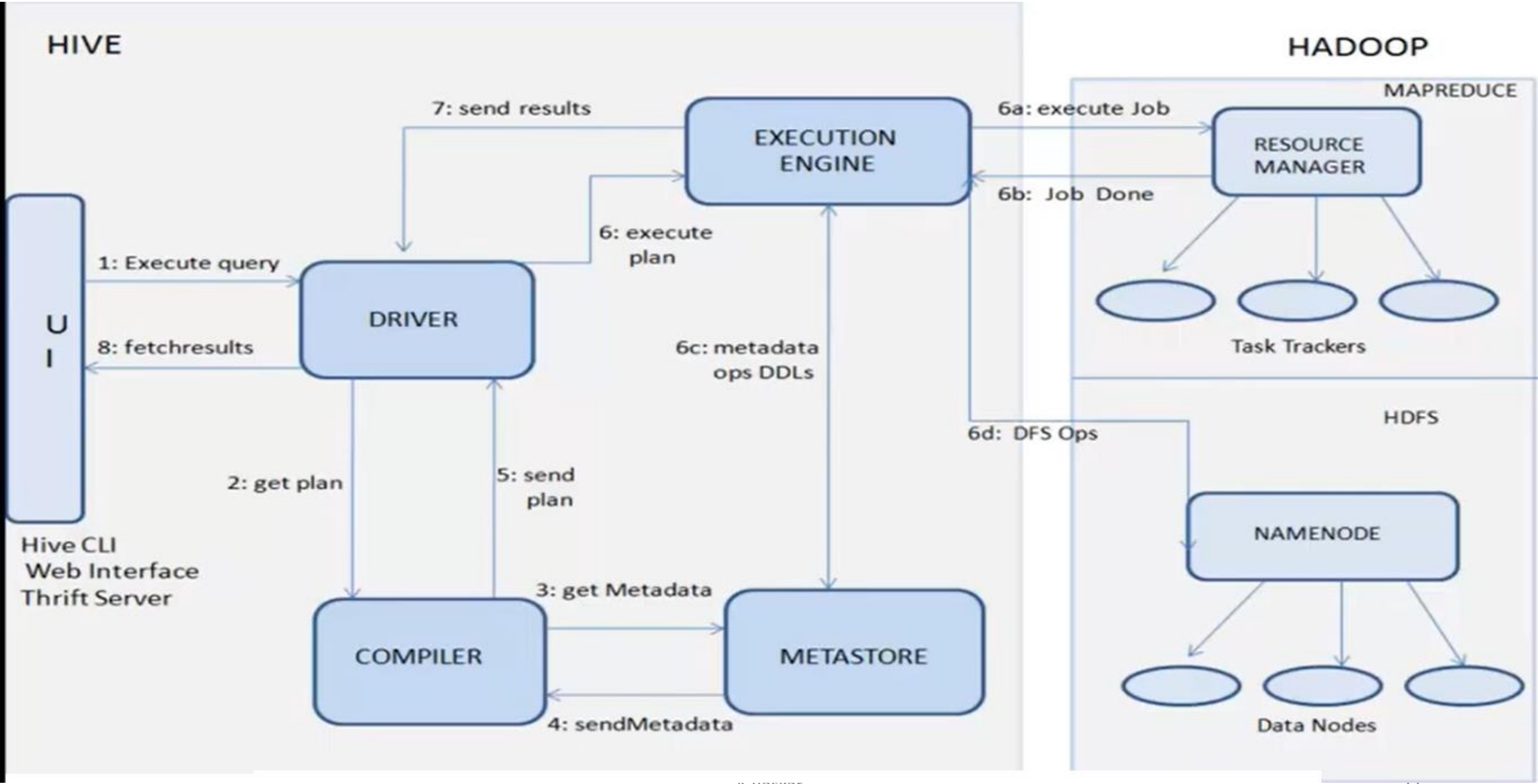
- Optimizer performs the transformation operations on the execution plan and splits the task to improve efficiency and scalability.

Execution Engine

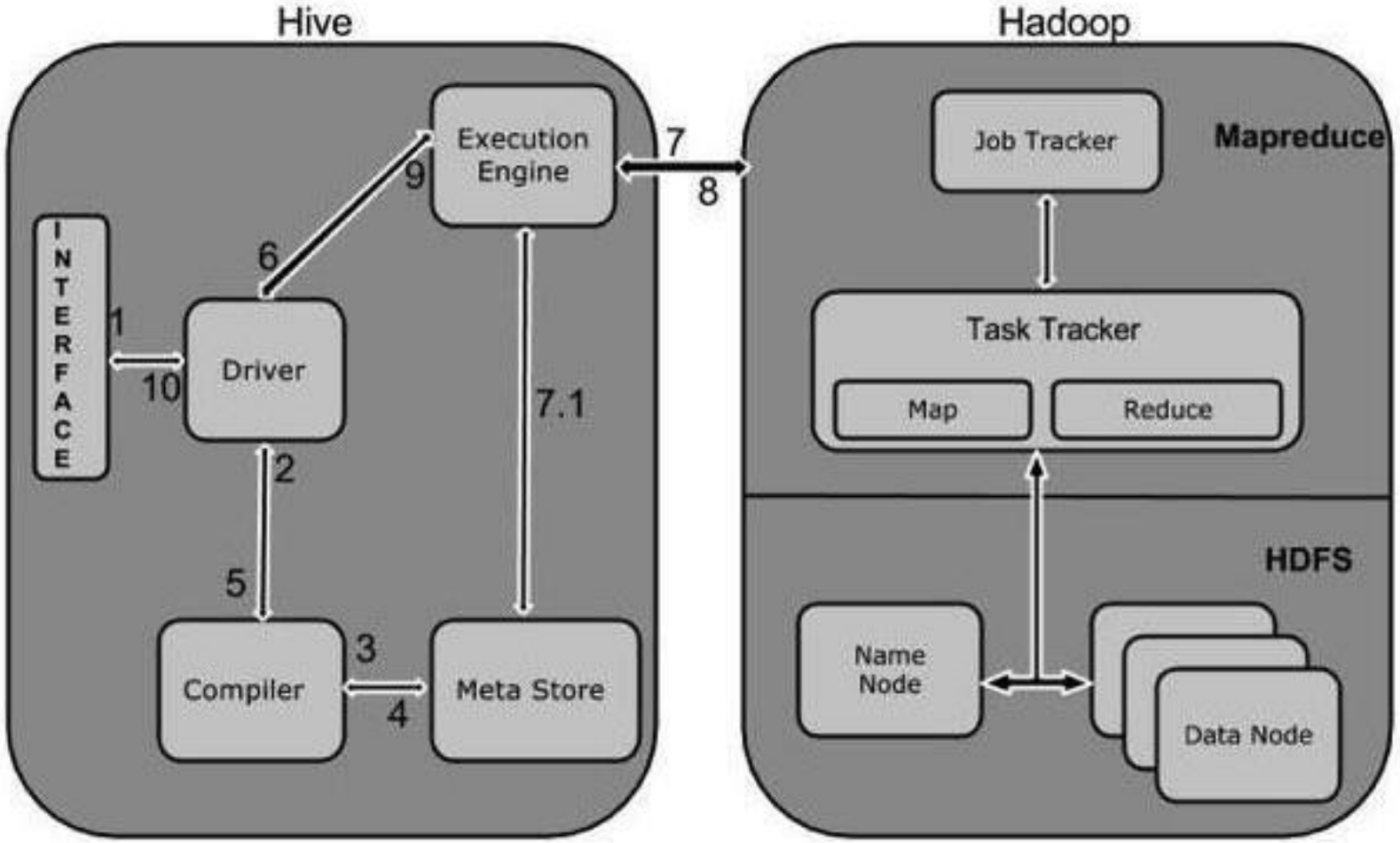
- Execution engine, after the compilation and optimization steps, executes the execution plan created by the compiler in order of their dependencies using Hadoop.

Metastore

- Metastore is a central repository that stores the metadata information about the structure of tables and partitions, including column and column type information.
- It also stores information of serializer and deserializer, required for the read/write operation, and HDFS files where data is stored. This metastore is generally a relational database.


HIVE
HADOOP
MAPREDUCE
7: send results
EXECUTION ENGINE
6a: execute Job
RESOURCE MANAGER
6b: Job Done
6: execute plan
1: Execute query
DRIVER
U I
8: fetchresults
6c: metadata ops DDLs
Task Trackers
HDFS
6d: DFS Ops
2: get plan
5: send plan
NAMENODE
Hive CLI
Web Interface
Thrift Server
3: get Metadata
COMPILER
METASTORE
4: sendMetadata
Data Nodes

Working of Hive



Ref: https://www.tutorialspoint.com/hive/hive_introduction.htm

Working of Hive

- **Step 1: executeQuery:** The user interface calls the execute interface to the driver.
- **Step 2: getPlan:** The driver accepts the query, creates a session handle for the query, and passes the query to the compiler for generating the execution plan.
- **Step 3: getMetaData:** The compiler sends the metadata request to the metastore.
- **Step 4: sendMetaData:** The metastore sends the metadata to the compiler.
- The compiler uses this metadata for performing type-checking and semantic analysis on the expressions in the query tree. The compiler then generates the execution plan (**Directed acyclic Graph**). For Map Reduce jobs, the plan contains **map operator trees** (operator trees which are executed on mapper) and **reduce operator tree** (operator trees which are executed on reducer).

- **Step 5: sendPlan:** The compiler then sends the generated execution plan to the driver.
- **Step 6: executePlan:** After receiving the execution plan from compiler, driver sends the execution plan to the execution engine for executing the plan.
- **Step 7: submit job to MapReduce:** The execution engine then sends these stages of DAG to appropriate components. For each task, either mapper or reducer, the deserializer associated with a table or intermediate output is used in order to read the rows from HDFS files. These are then passed through the associated operator tree.
- Once the output gets generated, it is then written to the HDFS temporary file through the serializer. These temporary HDFS files are then used to provide data to the subsequent map/reduce stages of the plan.
- For DML operations, the final temporary file is then moved to the table's location.

- **Step 8,9,10: sendResult:** Now for queries, the execution engine reads the contents of the temporary files directly from HDFS as part of a fetch call from the driver. The driver then sends results to the Hive interface.

Hive Meta store

Meta store : Hive table definitions and mappings to the data are stored in Meta store.

Meta store consists of the following:

- Metastore services: Offers interface to the hive
- Databases: Stores data definitions, mappings to the data and others.

Meta data includes ,

- ID s of DBS
- IDs of Tables
- IDs of Indexes
- The time of creation of tables
- Input format
- Output format

Types of Meta Sore

Embedded Meta store:

Used for unit tests
Default meta store
only one process is allowed to connect at a time
Here both DB and Meta store services embedded with main Hive server process.

Local meta store:

Meta data can be stored in any RDBMS components like MySQL.
Allows multiple connections at a time
Here Meta store service runs in Hive Server but DB runs in separate process.

Remote Meta store:

Hive driver and Meta store interface runs on different JVM.
This way the DB can be firewalled from the hive users and also DB credentials are completely isolated from hive users.