BIG DATA

HIVE

HIVE : INTRODUCTION

Hive is a data warehousing infrastructure based on Apache Hadoop. Hadoop provides massive scale out and fault tolerance capabilities for data storage and processing on commodity hardware.

Hive is designed to enable easy data summarization, ad-hoc querying and analysis of large volumes of data. It provides SQL like which enables users to do ad-hoc querying, summarization and data analysis easily.

HIVE : INTRODUCTION

Apache Hive is an open-source data warehousing tool for performing distributed processing and data analysis. Hive was developed by **Facebook.**

Apache Hive features to provide Hive Query language(HQL), which is declarative language similar to SQL for Data Analysis. Hive translates the hive queries into MapReduce programs.

HIVE : CHARACTERISTICS

It supports developers to perform processing and analyses on structured and semi-structured data by replacing complex java MapReduce programs with hive queries.

It uses the concepts :

- It uses HDFS for storage and retrieval of data.
- The scripts of Hive uses MapReduce for execution

Principles of Hive

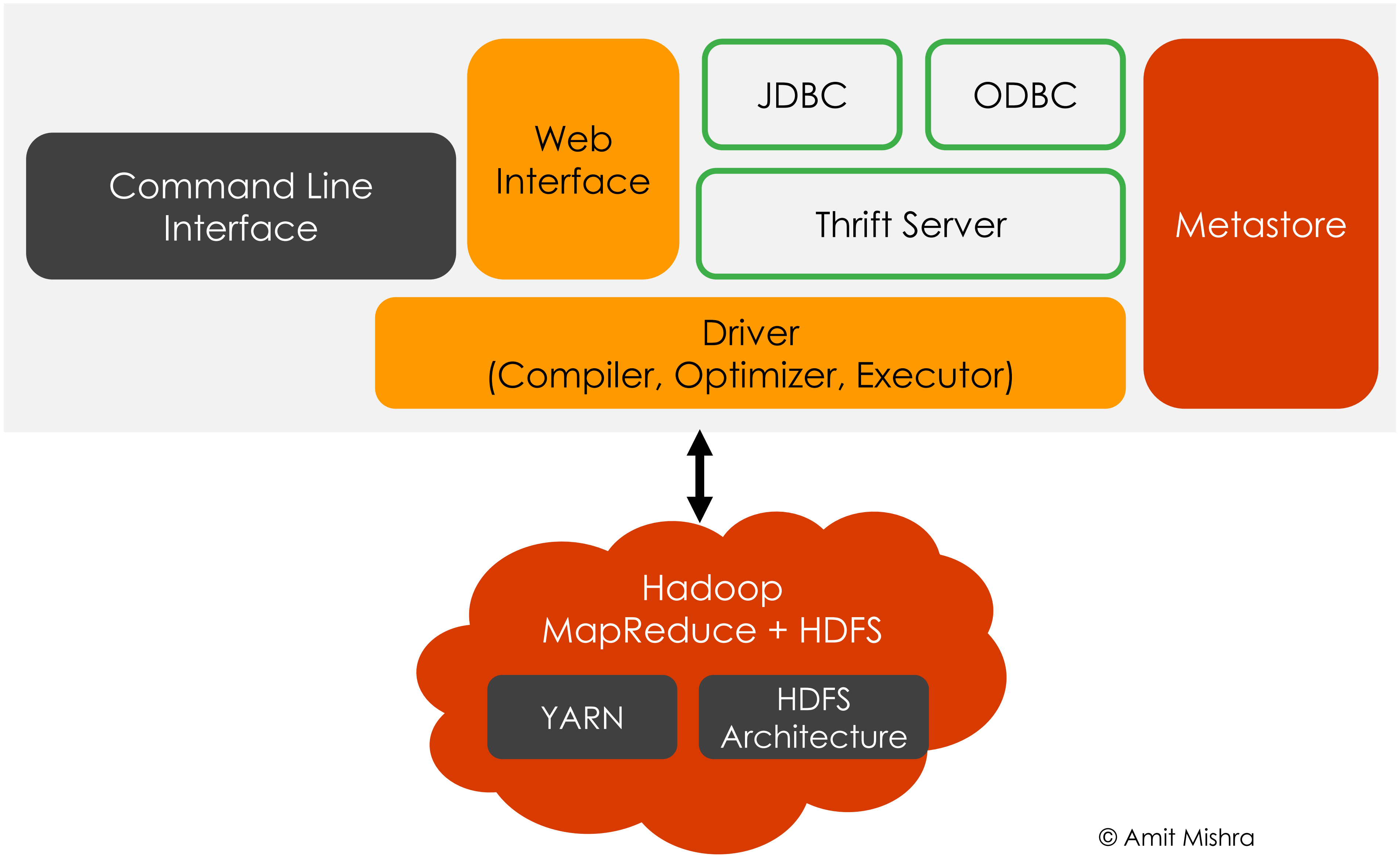
Hive commands are similar to SQL which is data warehousing tool.

Interoperability (extensible framework to support different files and data formats).

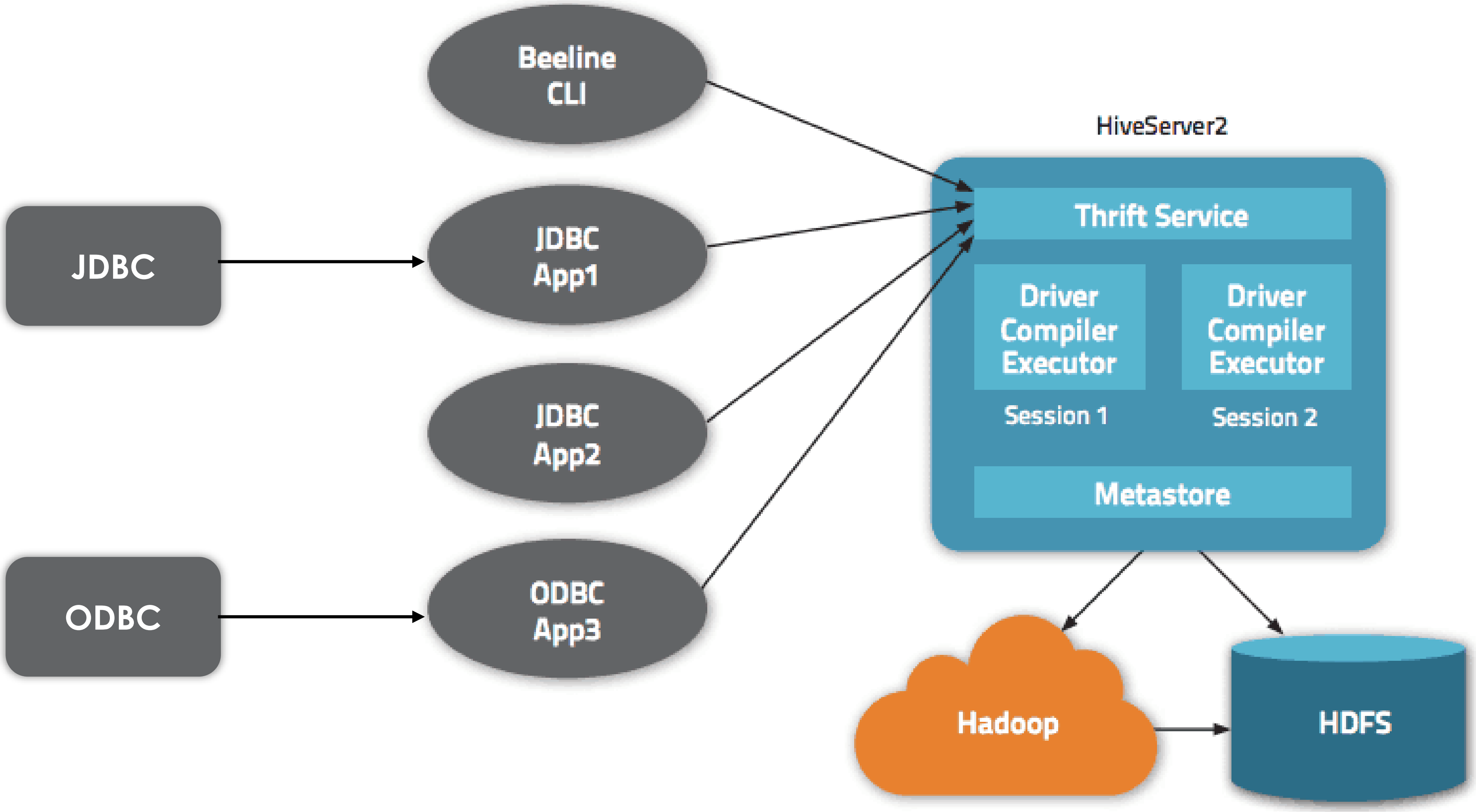
Extensibility (pluggable MapReduce scripts in the language of your choice – rich, user def data types & user def functions.

Performance is better in Hive since Hive engine uses the best in-built script to reduce the execution time while enabling high output.

HIVE : System Architecture and Components of Hive



HIVE : HIVESERVER2 ARCHITECTURE



HIVE : HIVESERVER2 ARCHITECTURE

HiveServer2 enables clients to execute queries against the Hive. It allows multiple clients to submit requests to Hive and retrieve the final results.

It implements a new Thrift-based RPC interface that can handle concurrent clients. The current release supports Kerberos, LDAP, and custom pluggable authentication. The new RPC interface also has better options for JDBC and ODBC clients, especially for metadata access.

Like, the older HiveServer1, HiveServer2 is a container for the Hive execution engine. For each client connection, it creates a new execution context that serves Hive SQL requests from the client. The new RPC interface enables the server to associate this Hive execution context with the thread serving the client's request.

HIVE : HIVE CLIENTS

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.

JDBC: JDBC driver that works with HiveServer2, enabling users to write JDBC applications against Hive. The application needs to use the JDBC driver class and specify the network address and port in the connection URL in order to connect to Hive.

ODBC: 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.

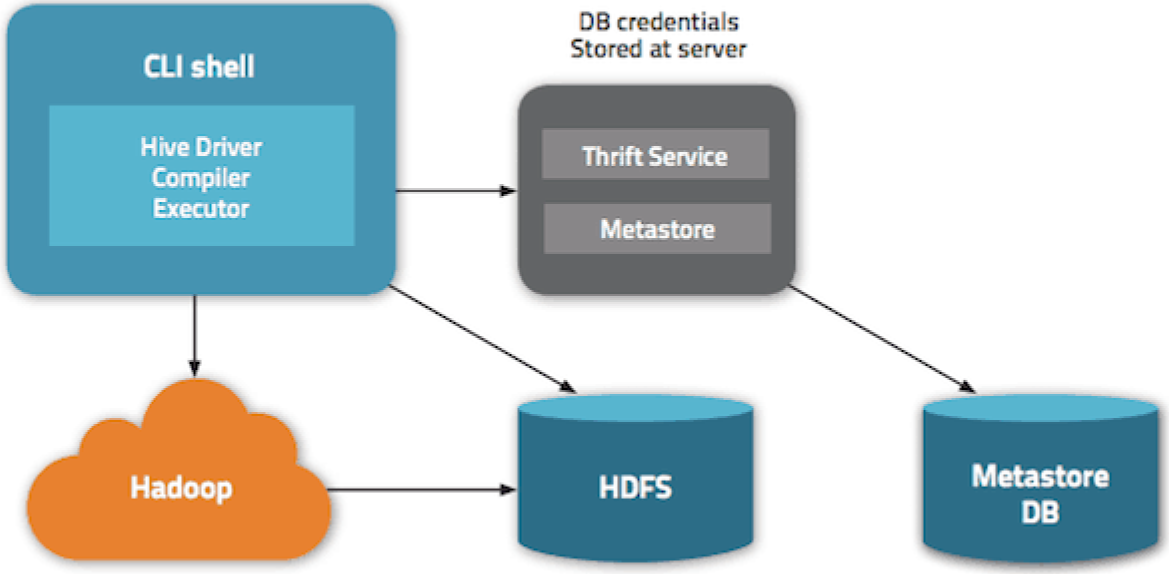
Beeline CLI: Hive also includes a new command-line interface (CLI) called Beeline that works with HiveServer2. Beeline is a JDBC application based on the SQLLine CLI (pure Java-console based utility for connecting with relational database and executing SQL queries), that supports embedded and remote-client modes. The embedded mode is where the Hive runtime is part of the client process itself, there’s no server involved.

HIVE : METASTORE

Metastore

- Metastore is the component that stores the system catalog and metadata about tables, columns, partitions, & so on . .
- Metadata is stored in traditional RDMS format, Apache Hive uses by default DERBY database. But its not compulsory its complimentary. If you wish you can add any JDBC database like MySQL.
- Metadata client : metastore_db

HIVE : METASTORE



The Hive metastore service runs in its own JVM process. Clients other than Hive, like Apache Pig, connect to this service via HCatalog for metadata access.

HiveServer2 supports local as well as remote metastore modes – which is useful when you have more than one service (Pig, Cloudera Impala, and so on) that needs access to metadata.

HIVE : DRIVER

Driver

(Compiler, Optimizer, Executor)

Hive driver is the component that:

- Manages the lifecycle of a HIVE Query Language(HQL) statement as it moves through Hive
- It maintains a session handle and any session statistics.
- It includes three basic components:
  - Compiler
  - Optimizer
  - Executer

HIVE : COMPILER

Compiler

Query Compiler is the driver components of Hive and checks for error, if no error encountered it converts HiveQL to Directed Acyclic Graph(DAG) of MapReduce tasks.

HIVE : OPTIMIZER

OPTIMIZER

- Query optimizer optimizes the HiveQL scripts for faster execution.
- It consists of a chain of transformations, so that the operator DAG resulting from on transformations is passed as an input to the next transformations.

HIVE : EXECUTER

EXECUTOR

- Executes the tasks produced by the compiler in proper dependency order.
- Interacts with the underlying Hadoop Interface to ensure perfect synchronization with Hadoop services.

BIG DATA

HIVE
(DATA MODEL & HIVE QUERY LANGUAGE)

HIVE : DATA TYPES

HIVE have three different Data types that are involved in Table Creation.

PREMITIVE TYPES

COMPLEX TYPES

USER-DEFINED TYPES

Integers: TINYINT, SMALLINT, INT and BIGINT
Boolean: BOOLEAN
Floating Types: FLOAT , DOUBLE
String: STRING (VARCHAR, CHAR)

HIVE : DATA TYPES

HIVE have three different Data types that are involved in Table Creation.

PREMITIVE TYPES

COMPLEX TYPES

USER-DEFINED TYPES

Structs: {a INT; b:INT}
Maps: M['group']
Arrays: ['a', 'b', 'c'], A[1] returns 'b'

HIVE : DATA TYPES

HIVE have three different Data types that are involved in Table Creation.

PREMITIVE TYPES

COMPLEX TYPES

USER-DEFINED TYPES

Structures with Attributes
Attributes can be of Any Type

HIVE : DATA MODELS

Tables in HIVE are analogous to Tables in Relational Databases. Tables can be filtered, projected, joined and unjoined. Additionally all the data of a table is stored in a directory in HDFS. Hive also supports the notion of external tables wherein a table can be created on pre-existing files or directories in HDFS by providing the appropriate location to the table creation.

Two Types of tables in HIVE

Managed Tables

External Tables

HIVE : DATA MODELS TABLE

HQL syntax to create Managed Tables:

```
CREATE [TEMPORARY] [EXTERNAL] TABLE IF NOT EXISTS [db_name.]
tab_name[(col_name data_type [COMMENT col_comment], . . . )]
[COMMENT table_comment]
[ROW FORMAT row_format_type]
[STORED AS file_format_type]
COMMENT ‘db_details’ ROW FORMAT DELIMITED
FIELDS TERMINATED BY ‘\t’
LINES TERMINATED BY ‘\n’
STORED AS file_format_type
```

IT SORES TABLES IN HIVE HDFS WAREHOUSE . . . .

HIVE : DATA MODELS EXTERNAL TABLE

- The EXTERNAL keyword lets you create a table and provide a LOCATION so that Hive does not use a default location for this table. This comes in handy if you already have data generated.
- Dropping an EXTERNAL table, data in the table is *NOT* deleted from the file system.
- An EXTERNAL table points to any HDFS location for its storage, rather than being stored in a folder specified by the configuration property.
- HQL Command to create External Commands:

```
CREATE EXTERNAL TABLE weatherext ( wban INT, date STRING)
COMMENT ‘this is external table view’
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
STORED AS TEXTFILE LOCATION ‘ /hive/data/weatherext’;
```

HIVE : PARTITIONING TABLES

- Hive stores tables in partitions. Partitions are used to divide the table into related parts. Partitions make data querying more efficient. For example in the above weather table the data can be partitioned on the basis of year and month and when query is fired on weather table this partition can be used as one of the column.
- HQL syntax to create the Partitioning the table:

```
CREATE EXTERNAL TABLE IF NOT EXSISTS weatherext
 (wban INT, date STRING)
PARTITIONED BY (year INT, month STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
LOCATION ‘location of text_file’;
```

HAPPY LEARNING