SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

Department of MCA

Topic: Hive

Course

19CAT702
Big Data Analytics

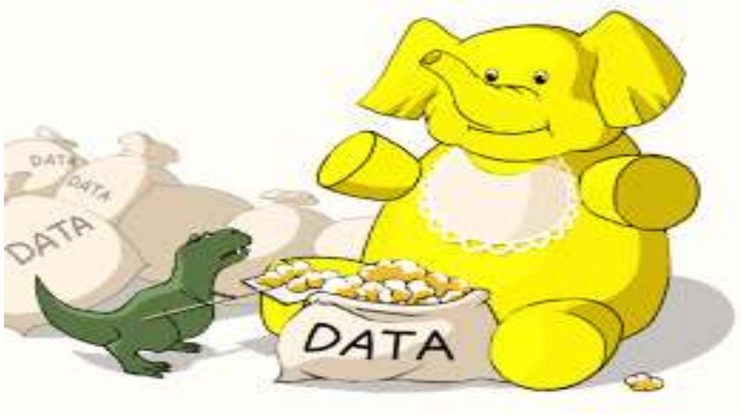
Unit IV

Hadoop

Elective

III Semester / II MCA

Session Objectives



- ❑ Understand the components which constitutes the architecture of Hive and its roles
- ❑ Demonstrate the SQL job query submission through Hive

Motivation

- Limitation of Hadoop like
  - Use Map/Reduce model of (low-level) programming
  - Not reusable
  - Error prone
  - More number of stages during transformation

What is Hive?

A Data warehouse software sYSTEMbuilt on top of Apache Hadoop for providing data summarization, query, and analysis

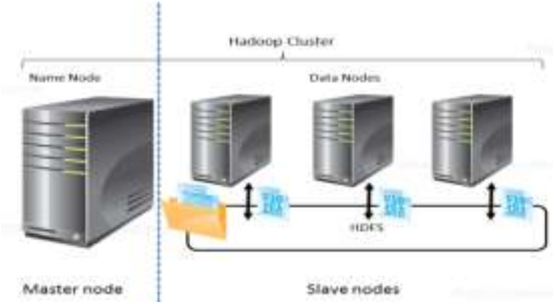
Hive

- Initially it was developed by Facebook and later Apache foundation took it up
- It abstracts the complexity of Hadoop MapReduce
- It supports queries expressed in SQL-like language called HiveQL which are compiled into MR jobs that are executed on Hadoop
- Supports Data Definition Language (DDL), Data Manipulation Language (DML) and User Defined Functions (UDF)
- Tables stored on HDFS as flat files

Hive Architecture

User Submit SQL queries

Query is converted into MapReduce Jobs



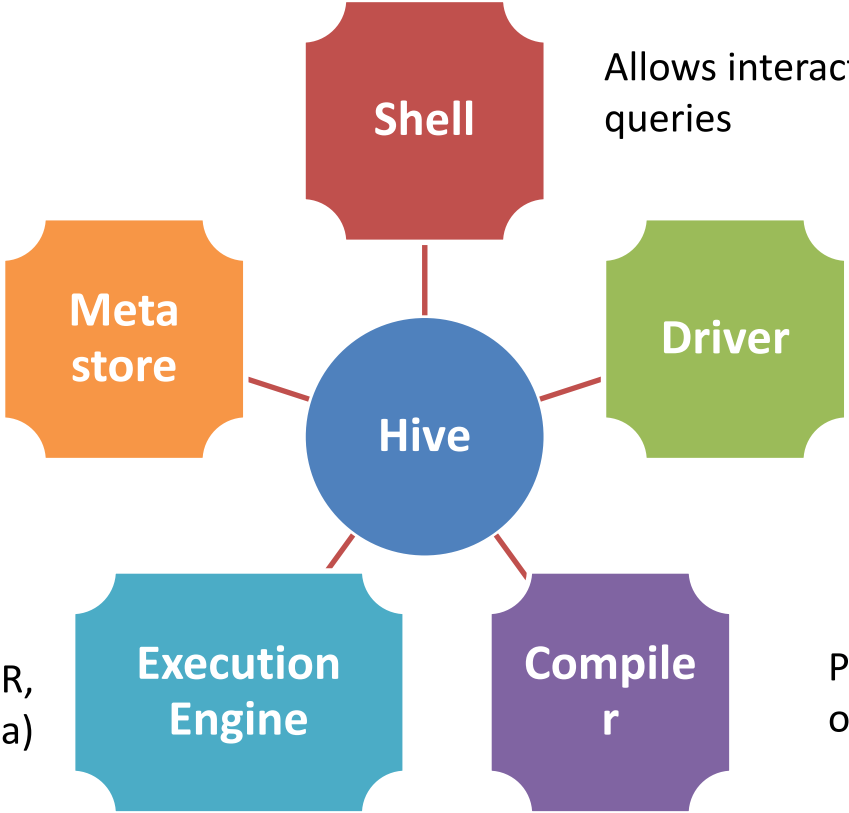
Characteristics

- ❑ Execution is like a series of MapReduce jobs that are generated automatically
- ❑ Similar to SQL handles structured data, It structures the unstructured data before querying it
- ❑ Warehouse generates the tables and databases before adding the data to them
- ❑ While executing a query, Hive uses the partition and bucket concept
- ❑ We can create user-defined functions to perform certain tasks such as filtering, data cleaning
- ❑ Schema information is stored in the traditional relational database

Difference between Hive & Database

Hive	Database
Schema on read	Schema on write
Not allowed	Record level Updates, transactions, and indexes exist
Write once, Read many times	Read and Write many times
Max data size: 100’s of Terabyte	Max data size: 10’s of Terabyte
Doesn’t Support OLTP	Supports OLTP

Hive Components



Allows interactive queries

Schema, location in HDFS, SerDe3

Session handles, fetch, execute

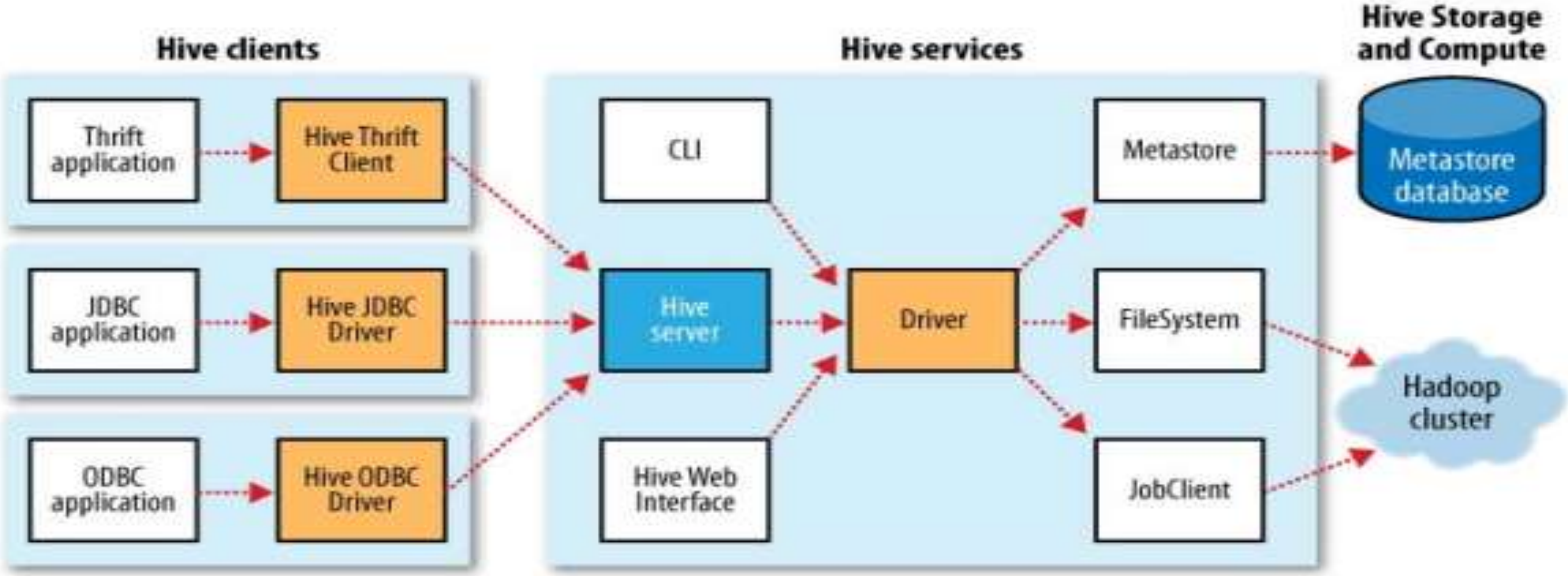
DAG of stages (MR, HDFS, metadata)

Parse, plan, optimize

Hive Services

- ❑ **Metastore**: stores system catalog
- ❑ **Driver**: manages life cycle of HiveQL query as it moves thru' HIVE; also manages session handle and session statistics
- ❑ **Query compiler**: Compiles HiveQL into a directed acyclic graph of map/reduce tasks
- ❑ **Execution engines**: The component executes the tasks in proper dependency order; interacts with Hadoop
- ❑ **HiveServer**: provides Thrift interface and JDBC/ODBC for integrating other applications.
- ❑ **Client components**: CLI, web interface, jdbc/odbc inteface

Hive Architecture



Hive

- ❑ Extensibility interface include Server, User Defined Functions and User Defined Aggregate Function
- ❑ Cli -The command line interface to Hive
- ❑ Hwi - The Hive Web Interface
- ❑ Thrift Client - makes it easy to run Hive commands from a wide range of programming languages
- ❑ JDBC Driver - Hive provides a Type 4 (pure Java) JDBC driver, defined in the class org.apache.hadoop.hive. jdbc.HiveDriver
- ❑ ODBC Driver- allows applications that support the ODBC protocol to connect to Hive

Hive Data Model

- ❑ Tables
  - Typed columns (int, float, string, boolean)
  - Also, list: map (for JSON-like data)
- ❑ Partitions
  - For example, range-partition tables by date
- ❑ Buckets
  - Hash partitions within ranges (useful for sampling, join optimization)

Hive Data Model

- Typed columns (int, float, string, boolean)
- Also, list: map (for JSON-like data)

Table

Partiti on

Table

Hash partitions within ranges (useful for sampling, join optimization)

For example, range-partition tables by date

Hive Data Model - Query

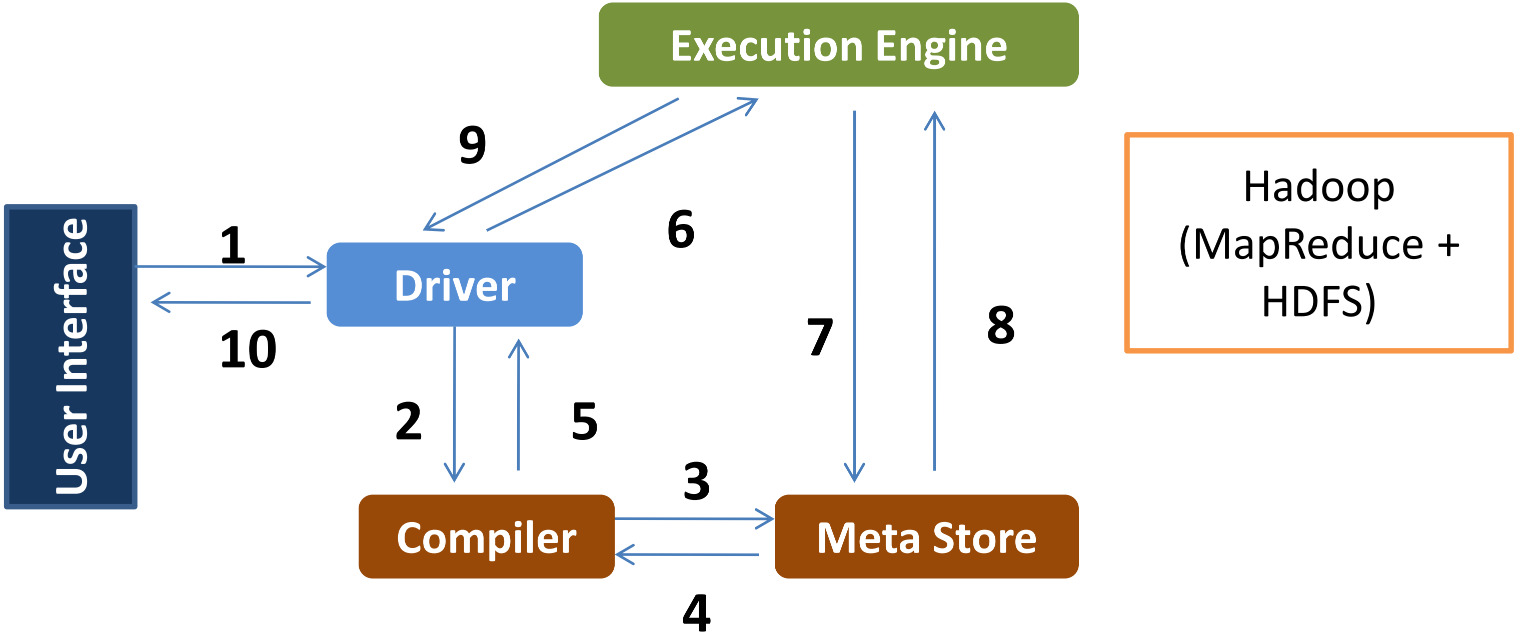
- ❑ *CREATE TABLE sales(id INT, items ARRAY<STRUCT <id:INT,name: STRING>)* ***PARITIONED BY (ds STRING) CLUSTERED BY (id) INTO 32 BUCKETS***

- ❑ *SELECT id FROM sales* ***TABLESAMPLE (BUCKET 1 OUT OF 32)***

Physical Layout

- ❑ Warehouse directory in HDFS
  - ▪ E.g., /user/hive/warehouse
- ❑ Tables stored in subdirectories of warehouse
  - ▪ Partitions form subdirectories of tables
- ❑ Actual data stored in flat files
  - ▪ Control char-delimited text, or SequenceFiles
  - ▪ With custom SerDe, can use arbitrary format

Data flow in Hive



Data flow in Hive

Process

1 UI sends the query to Driver

2 Driver uses Compiler to check query syntax and for query plan

3 Compiler sends metadata request to Metastore

4 Metastore sends metadata as a response

5 Compiler resends the plan to the drive

6 Driver sends the execute plan to the execution engine

7 Execution engine sends the job to JobTracker of Hadoop

8 Execution engine receives the results from Data nodes

9 Execution engine sends those result to the driver

10 Driver sends the results to Hive Interfaces.

HiveQL

❑ HiveQL / HQL provides the basic SQL-like operations:

- Query using SELECT
- Filtering rows by WHERE clause
- JOINing tables
- Aggregate using GROUP BY
- Store query results into another table
- Download results to a local directory
- Manage tables and queries with CREATE, DROP, and ALTER

Physical Layout

- ❑ **Data Types**
- Supports both primitive and complex data types
- Primitive types: TINYINT , SMALLINT , INT , BIGINT, FLOAT, DOUBLE BOOLEAN, and a STRING
- Complex data types: It includes ARRAY, MAP and STRUCT
- ❑ **Operators ad functions**
- Relational, arithmetic and logical operators
- Mathematical and statistical functions, string functions, date functions conditional functions, aggregate functions, and functions for working with XML (using the xpath function) and JSON.

Hive QL

- ❑ **Create Table**

```
CREATE TABLE table_name
    (col1 data_type,
       col2 data_type,
       col3 data_type,
       col4 datatype )
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    STORED AS format_type;
```

```
CREATE TABLE employees  (
     (name STRING,
       salary FLOAT,
       subordinates ARRAY<STRING>,
       deductions MAP<STRING, FLOAT>,
       address STRUCT<street:STRING,
                                  city:STRING,
                                  state:STRING,
                                  zip:INT>)
   ROW FORMAT DELIMITED
   FIELDS TERMINATED BY '\t'
   STORED AS TEXTFILE;
```

Hive QL

❑ **Partitioning Table**

- Divide data based on partition column
- make some queries run faster
- Use PARTITION BY clause when creating table
- Use PARTITION clause when loading data
- SHOW PARTITIONS will show a table’s partition

Hive QL

❑ **Buckets**

- Can speed up queries that involve sampling the data
- Use CLUSTERED BY when creating table
  - For sorted buckets, add SORTED BY
- To query a sample of your data, use TABLESAMPLE

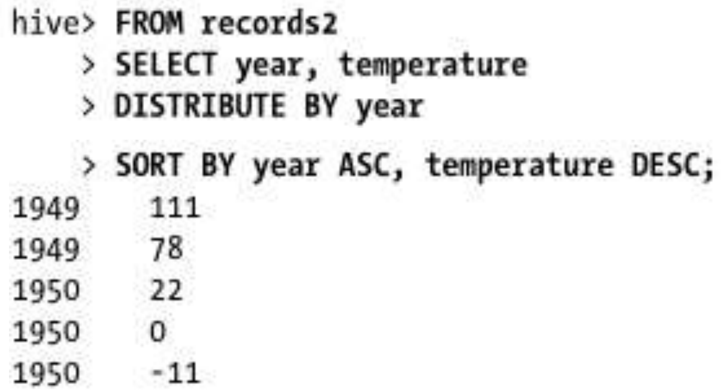
Hive QL

❑ Loading data

- Can speed up queries that involve sampling the data
- Use CLUSTERED BY when creating table
  - For sorted buckets, add SORTED BY
- To query a sample of your data, use TABLESAMPLE

Hive QL

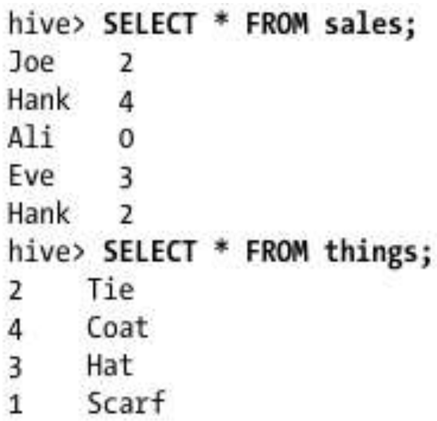
- **Sorting and Aggregating**
- Sorting data in Hive can be achieved by use of a standard ORDER BY clause
- But to do so it sets the number of reducers to one
- SORT BY produces a sorted file per reducer
- DISTRIBUTE BY clause does aggregation



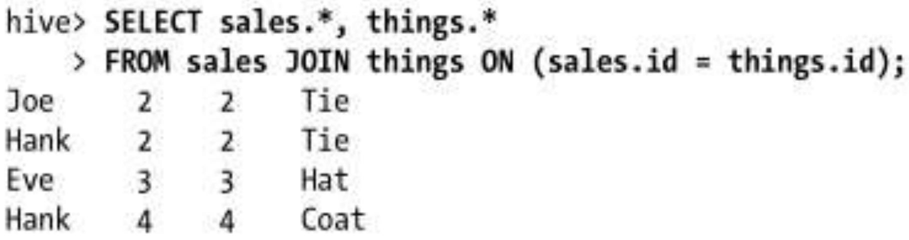
```
hive> FROM records2
    > SELECT year, temperature
    > DISTRIBUTE BY year

    > SORT BY year ASC, temperature DESC;
1949    111
1949    78
1950    22
1950    0
1950    -11
```

Hive QL

- **Inner Joins**
- where each match in the input tables results in a row in the output



```
hive> SELECT * FROM sales;
Joe     2
Hank    4
Ali     0
Eve     3
Hank    2
hive> SELECT * FROM things;
2       Tie
4       Coat
3       Hat
1       Scarf
```



```
hive> SELECT sales.*, things.*
    > FROM sales JOIN things ON (sales.id = things.id);
Joe     2     2     Tie
Hank    2     2     Tie
Eve     3     3     Hat
Hank    4     4     Coat
```

Hive QL

❑ **Outer Joins**

- allow you to find nonmatches in the tables being joined

SELECT sales., things.* FROM sales LEFT OUTER JOIN things ON (sales.id = things.id);*

Hive QL

- **Views**
- A view is a sort of “virtual table” defined by a SELECT
- Used to present data to users in a different way to the way it is actually stored on disk

```
CREATE VIEW valid_records AS SELECT * FROM records2 WHERE
temperature !=9999 AND
(quality = 0 OR quality = 1 OR quality = 4 OR quality = 5 OR quality = 9);
```

References

- ❑ Tom White, “ Hadoop: The Definitive Guide” Third Edition, O’reilly Media, 4th Edition, 2012
- ❑ https://www.guru99.com/hive-tutorials.html
- ❑ **https://data-flair.training/blogs/apache-hive-tutorial/**
- ❑ **https://www.simplilearn.com/tutorials/hadoop-tutorial/hive**

Assessment

Assessment

THANK YOU!