Hive

What is Hive ?

- Hive is a data warehouse tool
- It was developed at Facebook to manage their log files
- Hive uses following:
  - HDFS for storage
  - MapReduce for execution
  - Stores metadata in an RDBMS
- Hive queries are compiled into MapReduce jobs and then the job is run on Hadoop cluster.

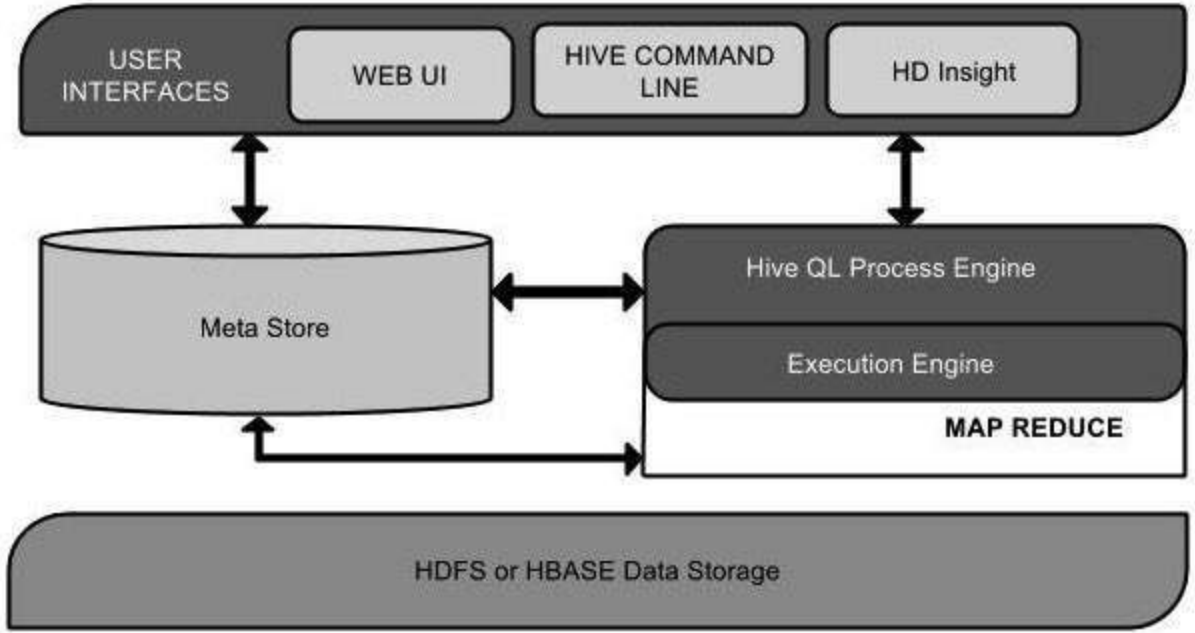
Features of Hive

- Hive provides HQL (Hive Query Language)
- HQL is similar to SQL
- Hive supports various data types including list, structs and map
- Hive supports group by, order by and filter clauses
- Custom types and custom functions can be defined

Hive data units

- Databases
- Tables
- Partitions
  - Static partition
  - Dynamic partition
- Buckets (or clusters)

Hive architecture



Hive Data types

- Primitive Data Types
  - Numeric Data Type
    - TINYINT
    - SMALLINT
    - INT
    - BIGINT
    - FLOAT
    - DOUBLE
  - String Data Type
    - STRING
    - VARCHAR
    - CHAR
  - Miscellaneous Data Type
    - BOOLEAN
    - BINARY
- Collection Data Types
  - STRUCT
  - MAP
  - ARRAY

Hive file format

- Text file
  - CSV, TSV, JSON, XML
- Sequential file
  - Stores binary key value pairs
  - Supports compression
- RCFile (Record Columnar File)
  - Split file horizontally first
  - Then convert columns to rows

HQL

Hive Query Language

List all databases

```
SHOW DATABASES;
```

Create database

```
CREATE DATABASE IF NOT EXISTS STUDENTS;
```

Describe a database

```
DESCRIBE DATABASE STUDENTS;
```

Use a database

```
USE STUDENTS;
```

Drop database

```
DROP DATABASE STUDENTS;
```

Create table

CREATE TABLE IF NOT EXISTS STUDENT(rollno INT, name STRING, gpa FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

Describe table

DESCRIBE STUDENT;

Loading data into table from file

LOAD DATA LOCAL INPATH ‘file.tsv’ OVERWRITE INTO TABLE STUDENT;

Collection data types

CREATE TABLE STUDENT_INFO (rollno INT, name STRING, sub ARRAY<STRING>, marks MAP<STRING, INT>)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’

COLLECTION ITEMS TERMINATED BY ‘:’

MAP KEYS TERMINATED BY ‘!’

Querying table

SELECT * from STUDENT;

Static partition

- Create table if not exists static_part_student (rollno int, name string) partitioned by (gpa float) row format delimited fields terminated by '\t';
- Insert overwrite table static_part_student partition(gpa = 4) select rollno, name from student where gpa = 4;
- Alter table static_part_student add partition (gpa = 3);
- Insert overwrite table static_part_student partition (gpa = 3) select rollno, name from student where gpa = 3;

Dynamic partition

- Create table if not exists dynamic_part_student (roll int, name string) partitioned by(gpa float) row format delimited fields terminated by '\t';
- SET hive.exec.dynamic.partition = true;
- SET hive.exec.dynamic.partition.mode = nonstrict;
- Insert overwrite table dynamic_part_student partition(gpa) select roll, name, gpa from student;

Bucketing

- A bucket is a file whereas a partition is a directory
- Set hive.enforce.bucketing=true;
- Create table if not exists student_bucket (rollno int, name string, gpa float) clustered by (gpa) into 3 buckets;
- From student insert overwrite table student_bucket select rollno, name, gpa;

To display:

- Select distinct gpa from student_bucket tablesample(bucket 1 out of 3 on gpa);

Views

Create view student_view as select rollno, name from student

Select * from student_view limit 4;

Drop view student_view;

Word count in hive

Create table docs (line string);

Load data local inpath ‘filename.txt’ overwrite into table docs;

Create table word_count as

Select word, count(1) as count from

(Select explode(split(line , ‘ ‘)) as word from docs) w

Group by word order by word;

Select * from words;

RCFile Implementation

- Create table student_rc (rollno int, name string gpa float) stored as rcfile;
- Insert overwrite table student_rc select * from students;
- Select sum(gpa) from student_rc;

XML data

- Create table xmlsample(xmldata string);
- Load data local inpath ‘file.xml’ into table xmlsample;

Create table xpath_table as

Select xpath_int(xmldata, ‘employee/empid’),

xpath_string(xmldata, ‘employee/name’),

xpath_string(xmldata, ‘employee/designation’)

From xmlsample;

Select * from xpath_table;

UDF

```
package com.example.hive.udf;
import org.apache.hadoop.hive.ql.exec.Description;
import org.apache.hadoop.hive.ql.exec.UDF;
@Description(
  name="SimpleUDFExample")
```

```
public final class MyLowerCase extends UDF {
  public String evaluate(final String word) {
   return word.toLowerCase();
  }
}
```

Note: Convert this Java Program into Jar.

```
ADD JAR /root/hivedemos/UpperCase.jar;
CREATE TEMPORARY FUNCTION touppercase AS 'com.example.hive.udf.MyUpperCase';
SELECT TOUPPERCASE(name) FROM STUDENT;
```

Outcome:

```
hive> ADD JAR /root/hivedemos/UpperCase.jar;
Added [/root/hivedemos/UpperCase.jar] to class path
Added resources: [/root/hivedemos/UpperCase.jar]
hive> CREATE TEMPORARY FUNCTION touppercase AS 'com.example.hive.udf.MyUpperCase';
OK
Time taken: 0.014 seconds
hive>
```