Hive

A Warehousing Solution Over a Map-Reduce Framework

Agenda

- Why Hive?
- What is Hive?
- Hive Data Model
- Hive Commands
- Hive Shell Commands
- Hive Drivers
- HiveQL
- Pros and Cons

Challenges that Data Analysts faced

- Data Explosion

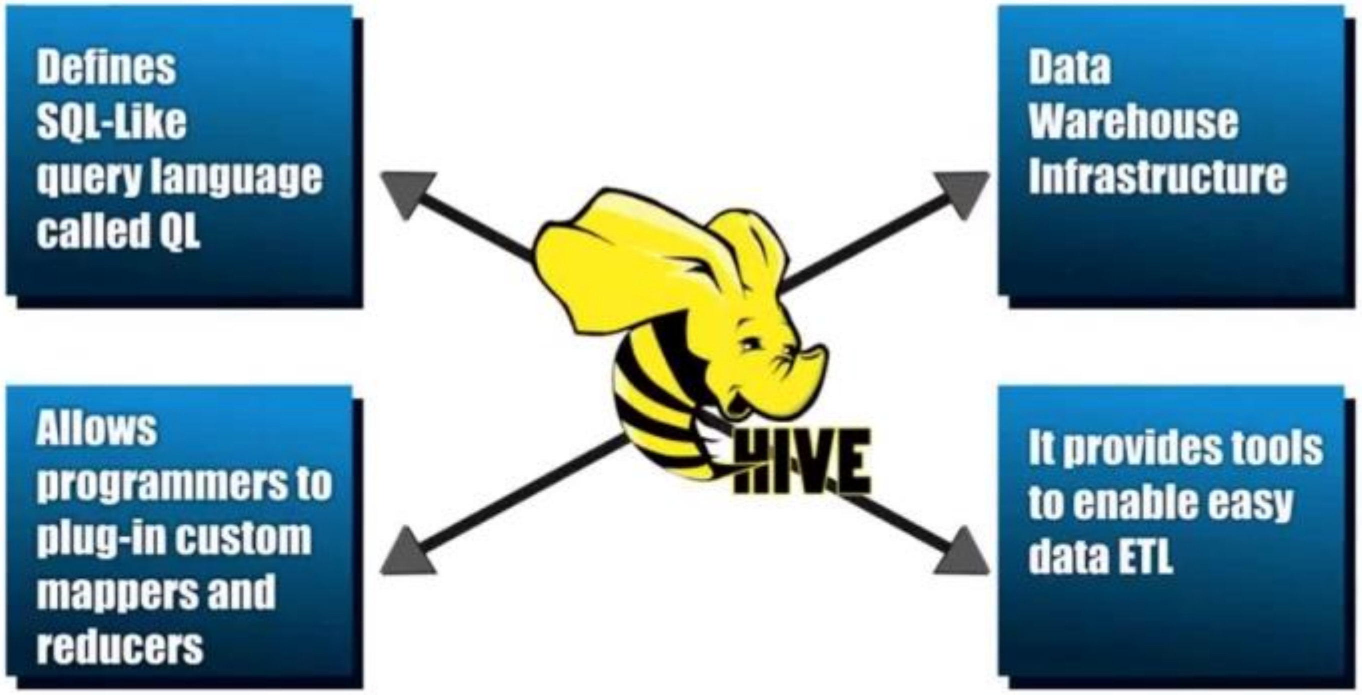
  - TBs of data generated everyday

Solution – HDFS to store data and Hadoop Map-Reduce framework to parallelize processing of Data

What is the catch?

- Hadoop Map Reduce is Java intensive
- Thinking in Map Reduce paradigm can get tricky

Hive Key Principles


Defines SQL-Like query language called QL
Data Warehouse Infrastructure
HIVE
Allows programmers to plug-in custom mappers and reducers
It provides tools to enable easy data ETL

HiveQL

DDL :

- CREATE DATABASE
- CREATE TABLE
- ALTER TABLE
- SHOW TABLE
- DESCRIBE

DML:

- INSERT

QUERY:

- SELECT
- GROUP BY
- JOIN

Hive Data Model

Data in Hive organized into :

- Tables
- Partitions

Hive Data Model Contd.

- Tables
  - Analogous to relational tables
  - Each table has a corresponding directory in HDFS
  - Data serialized and stored as files within that directory
  - External Vs Internal(Managed Tables) Table

Hive Data Types: Numeric

Type	Memory allocation
TINYINT	Its 1-byte signed integer (-128 to 127)
SMALLINT	2-byte signed integer (-32768 to 32767)
INT	4 –byte signed integer (-2,147,484,648 to 2,147,484,647)
BIGINT	8 byte signed integer
FLOAT	4 – byte single precision floating point number
DOUBLE	8- byte double precision floating point number
DECIMAL	We can define precision and scale in this Type

Hive Data Types: String

Type	Length
CHAR	255
VARCHAR	1 to 65355
STRING	We can define length here(No Limit)

Hive Data Types: Date/Date time

Type	Usage
Timestamp	Supports traditional Unix timestamp with optional nanosecond precision
Date	• It's in YYYY-MM-DD format.

> Select current_date(); -- gives current date

> Select current_timestamp(); -- gives current tmestamp

Hive Data Model Contd.

- Partitions
  - Each table can be broken into partitions
  - Partitions determine distribution of data within subdirectories

Hive Commands

#list all databases available

```
hive> show databases;
```

#Create a database

```
hive> create  database hadoop;
```

#Select a database

```
Hive> use hadoop;
```

# Show list of tables

```
hive> show tables;
```

Hive Commands – External Table

#create External table

```
CREATE External TABLE vbajaj.Sales (sale_id INT, amount FLOAT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ‘\t’
LINES TERMINATED BY ‘\n’ ;
```

#describe table structure

```
> desc sales;
```

#insert Values

```
> insert into table sales select *  from (select 1,111.01)a;
```

Hive Commands – External Table

#get hdfs location for table

```
> show create table sales;
```

#get hdfs files list

```
> hdfs dfs –ls /user/hive/warehouse/sales/
```

#cat the above file

```
hdfs dfs -cat /user/hive/warehouse/sales/000000_0
```

Hive Commands- Managed Tables

#create table

```
CREATE TABLE Sales_int (sale_id INT, amount FLOAT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ‘\t’
LINES TERMINATED BY ‘\n’
STORED AS TEXTFILE;
```

```
> insert into table sales_int select *  from (select
6,890.01000000676767576576576587586)a;
```

```
> select * from sales_int;
```

Hive Commands – Managed Tables

#create table

```
CREATE TABLE Sales_part (amount FLOAT)
PARTITIONED BY (sale_id INT )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

```
> set hive.exec.dynamic.partition.mode=nonstrict;
> insert into table sales_part partition(sales_id) select amt,id from (select 1 id ,111.01 as amt)a;
`
```

So each partition will be split out into different folders like

Sales_part/sales_id=1

Hive Shell Commands

```
hive –e “show databases;”

hive –e “show tables;”

hive –e “Select * from sales;”

hive –e “insert into sales select *  from (select 3,111.01)a;”
```

Hive Shell Commands

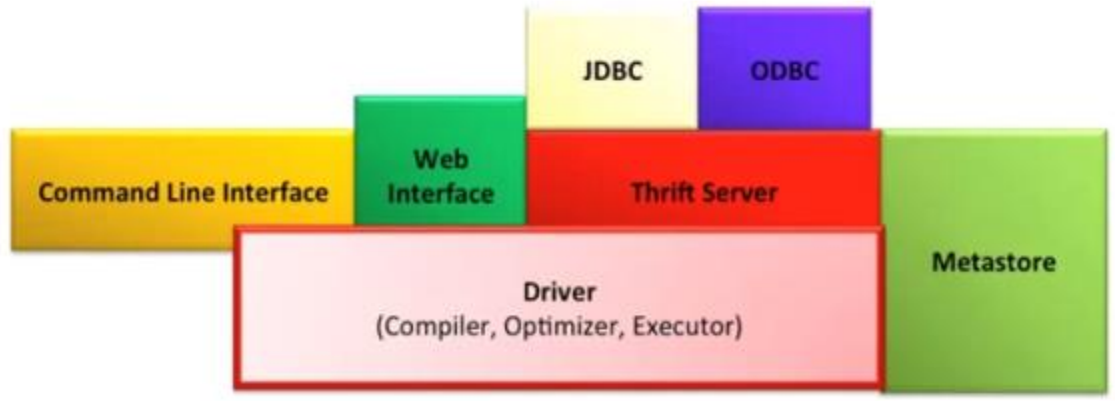
##for running files

```
CREATE TABLE hadoop.my_table (id INT, amount decimal(38,12))
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

insert into hadoop.my_table select * from (select 1,
44907375.5787794758857897598)a;
```

```
hive –f ‘create_my_table.sql’
```

Hive Driver



- **Driver** - Maintains the lifecycle of HiveQL statement
- **Query Compiler** – Compiles HiveQL in a DAG of map reduce tasks
- **Executor** - Executes the tasks plan generated by the compiler in proper dependency order. Interacts with the underlying Hadoop instance

Advantages

- Boon for Data Analysts
- Easy Learning curve
- Partitions(speed!)
- Flexibility to load data from localFS/HDFS into Hive Tables

Cons and Possible Improvements

- Extending the SQL queries support(Updates, Deletes)
- Explore methods for multi query optimization

REFERENCES

- https://hive.apache.org/
- https://cwiki.apache.org/confluence/display/Hive/Presentations
- https://developer.yahoo.com/blogs/hadoop/comparing-pig-latin-sql-constructing-data-processing-pipelines-444.html
- http://www.qubole.com/blog/big-data/hive-best-practices/