Apache HIVE

QUALITY THOUGHT

INFOSYSTEMS (INDIA) PVT. LTD.

What is Hive?

- Hive is a subproject of the Apache Hadoop project that provides a data warehousing layer built on top of Hadoop
- Hive allows you to define a structure for your unstructured big data, simplifying the process of performing analysis and queries by introducing a familiar, SQL-like language called HiveQL
- Hive is for data analysts familiar with SQL who need to do ad-hoc queries, summarization and data analysis on their HDFS data

Hive is not…

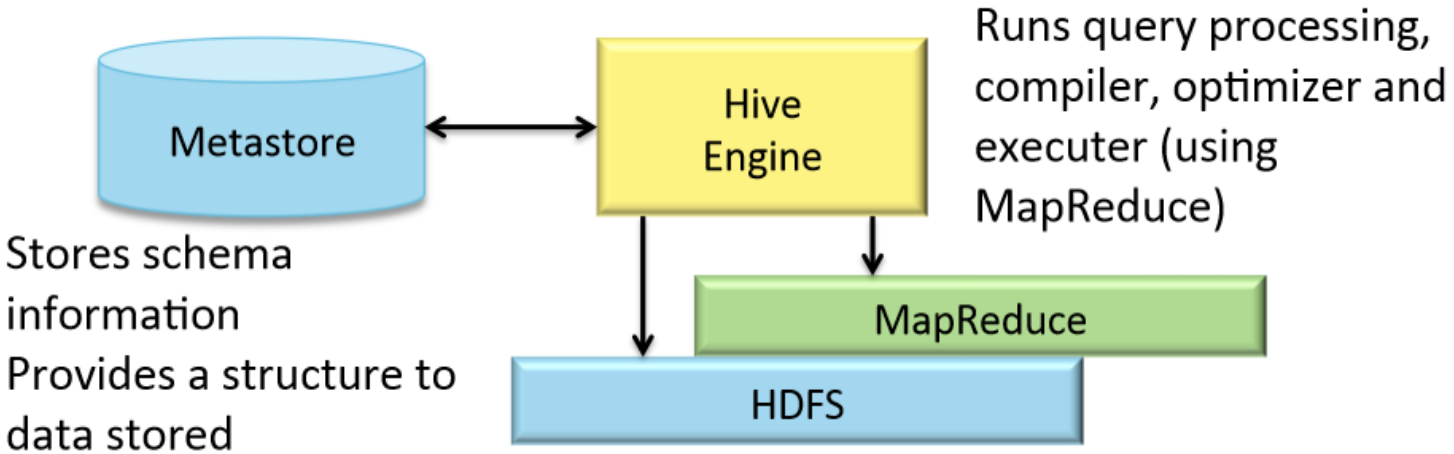
- Hive is not a relational database
- Hive uses a database to store metadata, but the data that Hive processes is stored in HDFS
- Hive is not designed for on-line transaction processing and does not offer real-time queries and row level updates

Pig vs. Hive

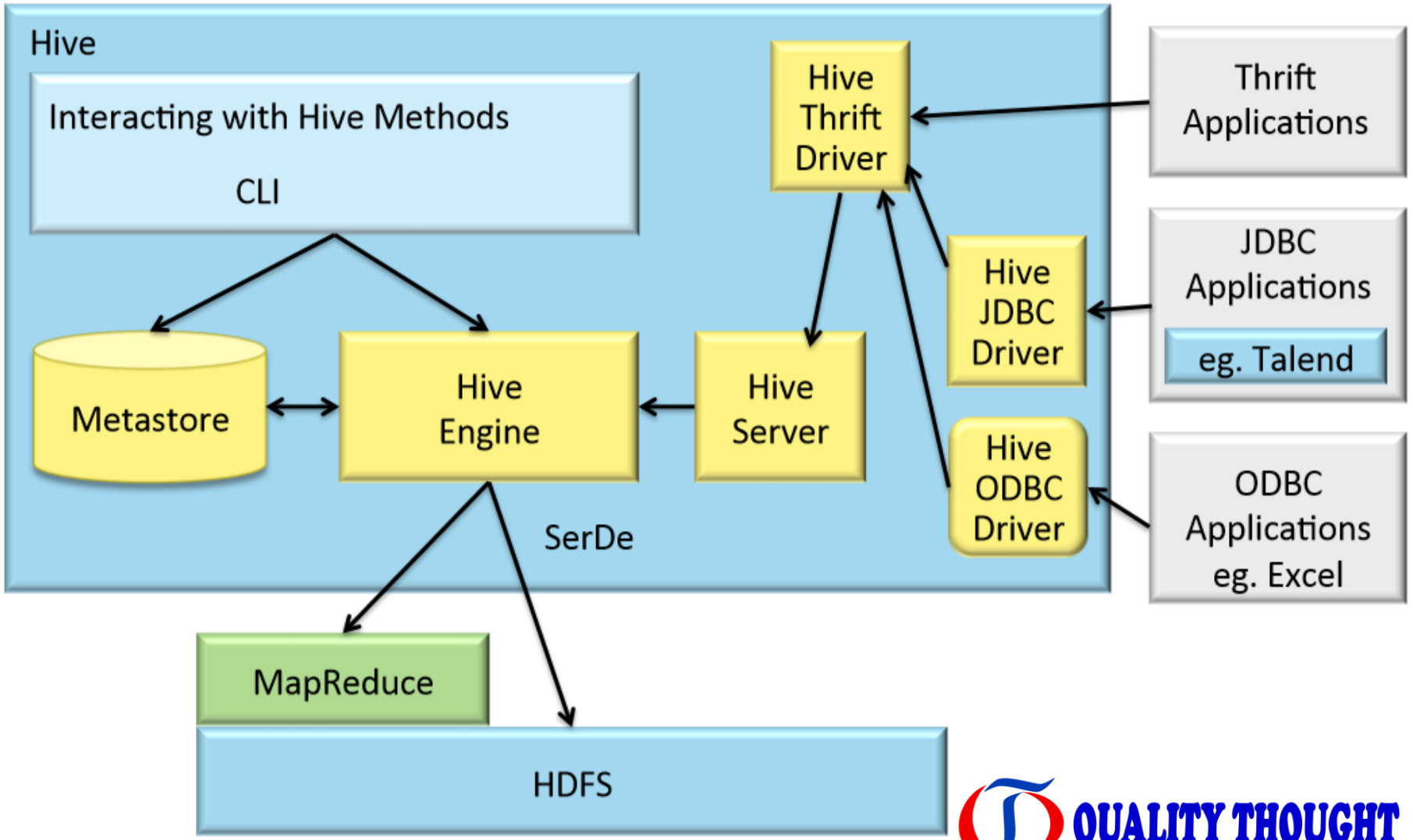
- Pig and Hive work well together
- Hive is a good choice:
  - when you want to query the data
  - when you need an answer to a specific questions
  - if you are familiar with SQL
- Pig is a good choice:
  - for ETL (Extract -> Transform -> Load)
  - preparing your data so that it is easier to analyze
  - when you have a long series of steps to perform
- Many businesses use both Pig and Hive together

Hive Basics

- Data Warehousing package built on top of Hadoop
- System for querying and managing structured data
  - Uses MapReduce for execution
  - Uses HDFS (or HBase) for storage



Hive Architecture



What is a Hive Table?

- A Hive table consists of:
  - Data: typically a file or group of files in HDFS
  - Schema: in the form of metadata stored in a relational database
- Schema and data are separate.
  - A schema can be defined for existing data
  - Data can be added or removed independently
  - Hive can be "pointed" at existing data
- You have to define a schema if you have existing data in HDFS that you want to use in Hive

Hive Shell

Running Jobs with the Hive Shell

- Primary way people use to interact with Hive

```
$ hive
hive>
```

- Can run in the shell in a non-interactive way

```
$ hive -f myhive.q
```

  - Use `-S` option to have only the results show

Hive Shell - Information

- At terminal enter:
  - `$ hive`
- List all properties and values:
  - `hive> set -v`
- List and describe tables
  - `hive> show tables;`
  - `hive> describe <tablename>;`
  - `hive> describe extended <tablename>;`
- List and describe functions
  - `hive> show functions;`
  - `hive> describe function <functionname>;`

Hive Shell – Querying Data

- Selecting Data
  - `hive> SELECT * FROM students;`
  - `hive> SELECT * FROM students`
    `WHERE gpa > 3.6 SORT BY gpa ASC;`

Table Operations

- Defining a table:

```
hive> CREATE TABLE mytable (name string, age
int)
        ROW FORMAT DELIMITED
        FIELDS TERMINATED BY ','
        STORED AS TEXTFILE;
```

- ROW FORMAT is a Hive-unique command that indicate that each row is comma delimited text
- HiveQL statements are terminated with a semicolon ';'
- Other table operations:
  - SHOW TABLES
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE

Managing Tables

- See current tables:
  - `hive> SHOW TABLES;`
- Check the schema:
  - `hive> DESCRIBE mytable;`
- Change the table name:
  - `hive> ALTER TABLE mytable RENAME to mt;`
- Add a column
  - `hive> ALTER TABLE mytable ADD COLUMNS (mycol STRING);`
- Drop a partition
  - `hive> ALTER TABLE mytable DROP PARTITION (age=17)`

Loading Data

- Use LOAD DATA to import data into a Hive table
- To load data indicate the path of the data
  - `LOAD DATA LOCAL INPATH 'input/mydata/data.txt' INTO TABLE myTable;`
- The files are not modified by Hive – they are loaded in as is
- Hive warehouse default location is:
  - `/apps/hive/warehouse`
- Hive can read all of the files in a particular directory
- Use the word OVERWRITE to write over a file of the same name
- The schema is checked when the data is queried. If a row does not match the schema, it will be read as null

INSERT

- Use INSERT statement to populate data into a table from another Hive table
- Since query results are usually large it is best to use an INSERT clause to tell Hive where to store your query
- Creating a table and inserting into it

```
hive> CREATE TABLE age_count  (name string, age int);
hive> INSERT OVERWRITE TABLE age_count
          SELECT age, COUNT(age)
          FROM mytable;
```

INSERT OVERWRITE

- OVERWRITE is used to replace the data in the table, otherwise the data is appended to the table.
  - Append happens by adding files to the directory holding the table data
  - `INSERT OVERWRITE TABLE newtable SELECT * FROM mytable;`
- Can write to a directory in HDFS
  - `INSERT OVERWRITE DIRECTORY '/hdfs/myresult' SELECT * FROM mytable;`
- Can write to a local directory (Linux typically)
  - `INSERT OVERWRITE LOCAL DIRECTORY...`

Tables

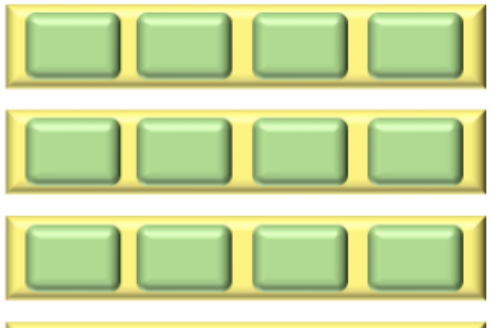
- A Hive table consists of the data stored and the metadata description of the data
  - The metadata is stored in a relational database
- Tables can be Hive-managed or external
  - Hive manages and can delete Hive-managed data
  - Hive can drop an external table, but it does not delete the data
  - Use external tables if other processes besides Hive will be accessing the data

Partitions

- A way to divide a table into parts based on the value of a partition column
  - Often date is used
- Queries that select a partitioned value are more efficient
- Subpartitions can be created
- Dynamic partitions are supported

Hash Buckets

- Tables or partitions can be further divided into buckets
- Bucketing divides data into a specified number of files based on the hash of the bucket column
- Each partition will have the specified number of files
- Buckets are useful to:
  - Make more efficient queries
  - Make sampling more efficient



Data partitioned by date, with each Date consisting of four buckets (arbitrary or by value)

Sampling Data

- Buckets – means to provide efficient queries on a random sample of data
- If there are 20 buckets then a Hive query can be run against only 1/20th of the data
- Specify the number of buckets in CREATE TABLE:

```
hive> CREATE TABLE mytable (USERID BIGINT, age
INT, gpa DOUBLE)
    PARTITION BY (age)
    CLUSTERED BY (userid) INTO 20 BUCKETS;
```

Hive Data Types - Scalar

- TINYINT – 1 byte integer
- BOOLEAN
- SMALLINT – 2 byte integer
- INT – 4 byte integer
- BIGINT – 8 byte integer
- DOUBLE – Double precision
- STRING – Sequence of characters
- TIMESTAMP
- BINARY
- DECIMAL

Hive Data Types – Complex

- STRUCT
  - Collection of named fields
  - Fields can be of different types
- MAP
  - Unordered collection of key-value pairs. Keys must be primitives; values may be any type
  - For a particular map the keys must be the same type and the values must be the same type
- ARRAY
  - An ordered collection of fields of the same type
- UNIONTYPE
  - at any one point hold exactly one of their specified data types

Operators

- Typical SQL operators supported
  - = for equality
  - IS NULL for testing whether null
  - LIKE for pattern matching
  - Arithmetic operators (+, -, etc.)
  - Logical operators
    - AND, OR, NOT
- Built-in Operators include:
  - Relational operators, arithmetic operators, logical operators, complex type constructors and operators on complex types

Some Hive Built-in Functions

- Mathematical Functions (`floor, rand, log`, etc.)
- Collection Functions (`size`)
  - Size of a map or an array
- Type Conversion Functions (`cast`)
- Date Functions (`unix_timestamp, day,` etc.)
- Conditional Functions (`if, case … when … then`, etc.)
- String Functions (`concat, substr, rpad,` etc.)
- Misc. Functions (`xpath`, etc.)
- Note: To see all functions in Hive run a **`SHOW FUNCTIONS`**
- Note: to get a description of a function enter:
  - **`DESCRIBE FUNCTION [function name]`**