Lecture 7 – Apache Hive

BDAT 1002

Introduction to Hive

Introduction

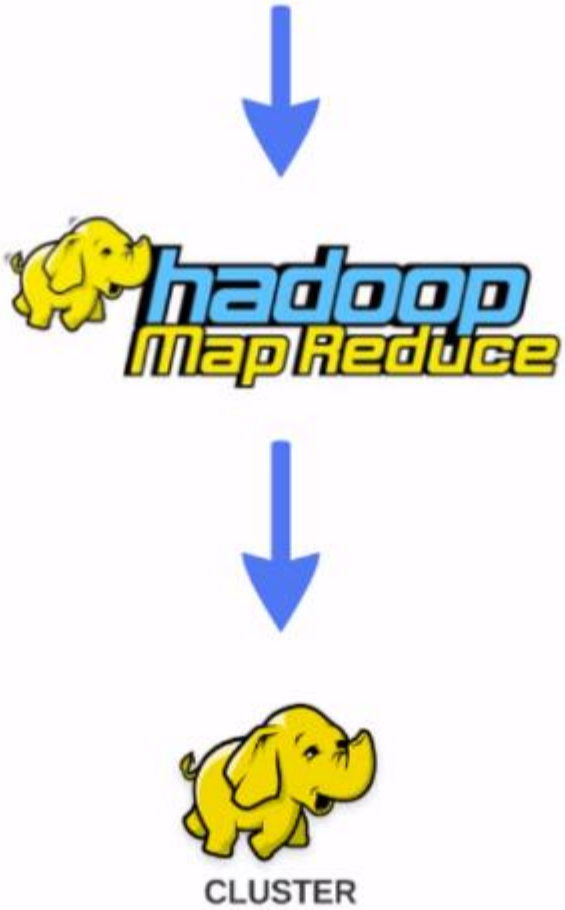
- So far we know:
  - How to write MapReduce programs with Java
  - Apache Pig to make writing MapReduce programs easier
- Hive makes it even more easier to work with Hadoop

Introduction to Hive

- It is very natural to think of data in a table format
- Hive attempts to treat data as tables and use traditional SQL queries to analyze
- It was developed by Facebook
- Widely used in industry
- Also a client tool like Pig
- Learning curve is much smaller than MapReduce

Process

select symbol, avg(volume) from stocks
group by symbol



Pig vs Hive

- Why do we need two tools doing similar things?
  - Pig and Hive were created by different companies around the same time to solve the same problem
  - The capabilities of each tool was not fully transparent to each company at early stages
  - This resulted in overlap

Pig vs Hive

- Do companies use both Pig and Hive?
  - Answer is yes
  - Use Pig for nightly extract transform and load (ETL)
  - Use Hive by developers by analysts for ad-hoc analysis of data

Hive vs. RDBMS

- Essentially, because Hive is using MapReduce behind the scenes, there is no bells and whistles of traditional RDBMS
- *No pointed updates and deletes*
- *Limited support for indexing*
- *Very high level transactional support*
  - *Introduced recently*
- *No triggers*

Hive vs. RDBSM

- Take away:
  - Don't look at Hive as an RDBMS system
  - Look at it as a tool that helps developers write MapReduce jobs in Hadoop

Our Approach

- Start by creating a database
- Create a Hive table
- Query our stocks dataset

Creating a Database and Table

- To create a database

```
hive> CREATE DATABASE stocks_db;
```

- To create objects in the database, we need to switch to the database

```
hive> USE stocks_db;
```

Creating a Database and Table

- Here is the syntax to create a table in the database

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS stocks_tb (
exch STRING,
symbol STRING,
ymd STRING,
price_open FLOAT,
price_high FLOAT,
price_low FLOAT,
price_close FLOAT,
volume INT,
price_adj_close FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/BDAT1002/hive';
```

This means the table stores on the HDFS.

like this

Creating a Database and Table

- We will do a simple select of all the columns and limit the result to 100

```
hive> SELECT * FROM stocks_db
      LIMIT 100;
```

Hive Table in Details

Hive Table in Details

- Whenever you are in the Hive environment, you need to be specific on the database you are using
- You can check out all the databases using the command:

```
hive> SHOW DATABASES;
```

- You can specify which database to work on with the command:

```
hive> USE stocks_db;
```

Hive Table in Details

- You can drop a database with following command:

```
hive> DROP DATABASE stocks_db;
```

- You can drop a table with

```
hive> DROP TABLE stocks;
```

Hive Table in Details

- However, you can only drop a database if a database is empty
- If the database already has tables you will either have to drop the tables first or
- Do a cascade command

```
hive> DROP DATABASE stock_db CASCADE;
```

Loading Hive Tables

Loading Hive Tables

- We'll learn four methods to load Hive tables
  - LOAD command
  - CTAS
  - INSERT … SELECT
  - LOCATION

Loading Hive Tables

- What do we need to load a Hive table?
- Two things
  - Table itself (the shell or columns)
  - Dataset (the actual column values)
- The shell is created much same way as Pig, you specify column names and types

Loading Hive Tables

- Here is a simple load command

```
hive> LOAD DATA INPATH '/BDAT1002/hive/stocks'
INTO TABLE stocks;
```

- This command will ***move*** the dataset from HDFS to the location mentioned in the table's *location attribute*
- You can check that the file has disappeared from the original location

Loading Hive Tables

- The dataset should be under location attribute folder
- Check to make sure it is there

Loading Hive Tables

- Now do a simple query

```
hive> SELECT * FROM stocks LIMIT 100;
```

Loading Hive Tables

- Another way to create a table is to simply use another table as a reference

```
hive> CREATE TABLE stocks_ctas
      AS SELECT * FROM stocks;
```

CTAS.

- Note that this command will start a MapReduce job to accomplish the task

Loading Hive Tables

- Let's look at the location attribute of this new table

```
hive> DESCRIBE FORMATTED stocks_ctas;
```

- Check what is under the location

```
hive> DESCRIBE FORMATTED stocks_ctas;
hive> !hadoop fs -ls
/user/hive/warehouse/stocks_db.db/stocks_ctas;
```

HDFS에 datasets을 Load한 경우 hive의 database 하위 (HDFS)에 생성된다.

This is the database name. stocks_db

Loading Hive Tables

- LOAD Command 
- CTAS 
- INSERT...SELECT
- LOCATION

Loading Hive Tables

- Let's look at how to use an INSERT/SELECT

일반 SQL에서는 사용하지 않는다.

```
hive> INSERT INTO TABLE stocks_ctas
SELECT s.* FROM stocks s;
```

→ same to SQL syntax

- This query will run a MapReduce job, take a look at the location

Loading Hive Tables

- INSERT INTO will append data to the table
- If you want to overwrite a table use:

```
hive> INSERT OVERWRITE TABLE stocks_ctas
SELECT s.* FROM stocks s;
```

- This query will also run a MapReduce job, take a look at the location
- The contents of the location are now overwritten (check)

Loading Hive Tables

- We have seen a few ways to load a table
- But to load a table you don't always need to use load command or even the insert command
- Whenever you are creating a Hive table, Hive will simply load the dataset that resides in the location pointed by the location attribute

Loading Hive Tables

- So this means, if you have a dataset that you would like to query using the Hive table, simply copy the dataset to the location
- You can either use the default directory or use a specific location

Creating a Database and Table

- Here is the syntax:

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS stocks_loc(
exch STRING,
symbol STRING,
ymd STRING,
price_open FLOAT,
price_high FLOAT,
price_low FLOAT,
price_close FLOAT,
volume INT,
price_adj_close FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/BDAT1002/hive/stocks_loc'
```

Loading Hive Tables

- Do a describe to make sure location is correct
- Query the table

Some final technical jargon

- Traditional RDBMS (MySQL, Oracle, etc), once data is loaded, it is controlled by the database engine
- With Hive you have complete control over your data
  - It resides in an HDFS location
  - Loading and inserting data using Hive Queries merely copies or moves the files into the HDFS location configured for the table

Hive Simple Selects

Hive Simple Queries

- Want to do simple queries
- Very similar to RDBSM queries

Hive Simple Queries

- When you write a query, Hive will look at it, analyze it, optimize it and create a MapReduce job

Hive Simple Queries

- Let's look at the first three statements

```
hive> SELECT * FROM stocks
WHERE symbol = 'GEL';

hive> SELECT * FROM stocks
WHERE symbol IN ('GEL', 'B3B');

hive> SELECT * FROM stocks
WHERE exch LIKE 'ABC%' AND symbol LIKE 'B%B';
```

Hive Simple Queries

- Let's look at a CASE statement query

```
hive> SELECT symbol, price_open, price_close, volume,
CASE
 WHEN volume < 20000 THEN 'low'
 WHEN volume >= 20000 AND volume < 40000 THEN 'middle'
 WHEN volume >= 40000 AND volume < 60000 THEN 'high'
 ELSE 'very high'
END AS volume_level
FROM stocks
WHERE symbol = 'GEL';
```

or ANSI Standard SQL?

Hive Simple Queries

- Bring a list of distinct symbols and exchanges

```
hive> SELECT DISTINCT exch, symbol FROM stocks;
```

Hive Simple Queries

- Sometimes you want to limit the number of records that you see

```
hive> SELECT * FROM stocks LIMIT 10;
```

- Note that this query did not result in a MapReduce job
- Simple queries without computations are run without MapReduce jobs

Hive Simple Queries

- This query will get us the average volume of stocks by year and symbol

```
hive> SELECT year(ymd), symbol, avg(volume) FROM
stocks GROUP BY year(ymd), symbol;
```

Hive Simple Queries

- This query is a slight variation where we further filter the records based on volume

```
hive> SELECT year(ymd), symbol, avg(volume) FROM
stocks GROUP BY year(ymd), symbol
HAVING avg(volume) > 400000;
```

Hive Simple Queries

- Sometime it is not practical to show your output to the screen
- Perhaps you want to save it for another job or for viewing later
- In such cases, you can save your queries to a file either locally or to HDFS

```
hive> INSERT OVERWRITE LOCAL DIRECTORY '/'
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,'
SELECT year(ymd), symbol, avg(volume) FROM stocks
GROUP BY year(ymd), symbol
HAVING avg(volume) > 400000;
```

Hive Tables

Hive Tables

- There are two types of tables in Hive
  - Managed table
  - External table
- Managed table has full control over it's data
  - When you drop the table, the table's dataset will also be deleted from HDFS

Hive Tables

- External table does not have full control over it's dataset
- When you drop the table, the dataset is not deleted from HDFS
- When do we use managed table, and when do we use external table?

Hive Tables

- Use managed table if Hive is the only application using the dataset
- Use external table if the dataset is shared between applications
- Let's see how to create each type of table

Hive Tables

- By default, when you create a table it is a managed table
- If you want to create an external table you have to specify the keyword "EXTERNAL"

Hive Tables

- Create the stocks table if you haven't done so
- Do a describe on it

```
hive> DESCRIBE FORMATTED stocks;
```

Hive Tables

- Check out the location of the table

```
hive> !hadoop fs -ls
/user/hive/warehouse/stocks_db.db/stocks;
```

- Now drop the table and check out the location again

```
hive> DROP TABLE stocks_db.stocks;
hive> !hadoop fs -ls
/user/hive/warehouse/stocks_db.db/stocks;
```

This is source of external table

Hive Tables

- Now create an external table and do a describe on it

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS stocks_ext(
exch STRING,
symbol STRING,
ymd STRING,
price_open FLOAT,
price_high FLOAT,
price_low FLOAT,
price_close FLOAT,
volume INT,
price_adj_close FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
```

Hive Tables

- Now load the table

```
hive> LOAD DATA INPATH ‘/BDAT1002/hive/stocks_db'
INTO TABLE stocks_ext;
```

Hive Tables

- Now let's look at the file location before and after the table is dropped.
- You should see the data there in both cases

Ordering Records in Hive

Hive Ordering

- Let's execute the following command:

```
hive> SELECT * FROM stocks_loc
ORDER BY price_close DESC;
```

- Notice the log file number of reducers is one
- Why?

Hive Ordering

- Let's increase the number of reducers

```
hive> SET mapreduce.job.reduces=3;
```

- Run the query again. Notice the log file number of reducers is ***still*** one
- Why?

Hive Ordering

- So what is a real solution? Use SORT BY
- When you use SORT BY, it uses multiple reducers.
- Look at this query:

```
hive> SELECT ymd, symbol, price_close
FROM stocks WHERE year(ymd) = '2003'
SORT BY symbol ASC, price_close DESC;
```

Hive Ordering

- We can also save the result to our local file system
- This will demonstrate a problem with SORT BY
- Make sure the number of reducers is set to 3 and run:

```
hive> INSERT OVERWRITE LOCAL DIRECTORY
'/home/cloudera/hive'
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
SELECT ymd, symbol, price_close
FROM stocks WHERE year(ymd) = '2003'
SORT BY symbol ASC, price_close DESC;
```

Hive Ordering

- If you look at the local directory, you see 3 files
  - One for each reducer
- There is a problem with this setup

Hive Ordering

- How do we make all the record from the same symbol go to the same reducer and end up in the same file
  - Use DISTRIBUTE BY

```
hive> INSERT OVERWRITE LOCAL DIRECTORY
'/home/cloudera/hive'
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
SELECT ymd, symbol, price_close
FROM stocks WHERE year(ymd) = '2003’
DISTRIBUTE BY symbol
SORT BY symbol ASC, price_close DESC;
```

Hive Ordering

- If we have the same set of columns in SORT BY and DISTRIBUTE BY, and you are sorting the records in ascending order, you can use CLUSTER BY:

```
hive> SELECT ymd, symbol, price_close
FROM stocks WHERE year(ymd) = '2003'
DISTRIBUTE BY symbol
SORT BY symbol ASC;
```

is equivalent to:

```
hive> SELECT ymd, symbol, price_close
FROM stocks WHERE year(ymd) = '2003'
CLUSTER BY symbol
```