Apache Hive

INF 551

Wensheng Wu

Hive

- Big data analytics (warehousing) software
  - For analyzing a large scale of data
  - Data may reside on HDFS
- HiveQL: SQL-like ad-hoc query language
- Hive turns query into MapReduce jobs

History

- 2007: Facebook data warehousing project
  - Developed for analyzing petabytes of data
- Used to be part of Hadoop project
  - Now a top-level project at Apache
- Used in Netflix
- Included in Amazon Elastic MapReduce (EMR) service

Workload

- Hive
  - Analytical workload (~ OLAP)
  - A query may need to process terabytes of data
- Cassandra & DynamoDB
  - Key-based (~ OLTP)
  - Processing a small amount of data per query

Installation on EC2

- Download binary at:
  - https://hive.apache.org/
- Or directly at:
  - http://apache.mirrors.lucidnetworks.net/hive/hive-2.1.1/apache-hive-2.1.1-bin.tar.gz

Installation on EC2

- tar xvf apache-hive-2.1.1-bin.tar.gz

- Move apache-hive-2.1.1-bin directory to your home
  - mv apache-hive-2.1.1-bin ~

Setting up

- Add these to ~/.bashrc
  - export HIVE_HOME=/home/ec2-user/apache-hive-2.1.1-bin
  - export PATH=$HIVE_HOME/bin:$PATH
  - export HADOOP_HOME=/home/ec2-user/hadoop-2.7.3
- Note: may need to change Hive and Hadoop installation directories to ones you use

Reduce heap size of Hadoop

- Default is 1GB: too big on EC2 free account
- Reduce it to 256MB
  - Modify hadoop-env.sh under $HADOOP_HOME/etc/hadoop directory
- This will prevent Hive's errors in performing join

```
# The maximum amount of heap to use, in MB. Default is 1000.
export HADOOP_HEAPSIZE=256
#export HADOOP_NAMENODE_INIT_HEAPSIZE=""
```

HDFS

- Start HDFS
  - Go to your Hadoop installation directory
  - Execute "sbin/start-dfs.sh"

Setting up temp & warehouse dirs

- $HADOOP_HOME/bin/hadoop fs -mkdir /tmp
- $HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
- $HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp
- $HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

Initializing metastore database

- Metastore is used by hive to store metadata, e.g., how tables are partitioned over nodes

- cd $HIVE_HOME
- bin/schematool -initSchema -dbType derby

- Note: if hive fails to start even after the initialization of metastore
  - Remove metastore_db diretory (under $HIVE_HOME)
  - Run the schematool again to initialize the schema

HiveQL

- Make sure hdfs is running
  - $HADOOP_HOME/sbin/start-dfs.sh

- Shell
  - bin/hive (executed under Hive installation dir)
  - Or $HIVE_HOME/bin/hive

Databases

- create database inf551;
  - Equivalent to "create schema inf551"

```
drwxrwxr-x   - vincent supergroup          0 2016-11-26 22:12 /user/hive/warehou
se/inf551.db
```

- drop database inf551;
- show databases;

- use inf551;

Databases

- set hive.cli.print.current.db=true;
  - Prompt will now show the current database

Create table

- create table purchase (buyer string,
  seller string,
  store string,
  product string,
  year int)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  STORED AS TEXTFILE;

One row per line

Another format is Hadoop sequence file

Describe & drop table

- desc purchase;

```
hive> desc purchase;
OK
buyer                   string
seller                  string
store                   string
product                 string
year                    int
Time taken: 0.049 seconds, Fetched: 5 row(s)
hive>
```

- drop table purchase;

purchase_year.txt

mary,john,Amazon,iphone 6,2016

mary,david,Ebay,ms office 2013,2016

mary,john,Amazon,thinkpad t460s,2015

mary,david,Ebay,db2 v9,2014

bill,john,Amazon,iphone 6,2015

bill,john,Amazon,thinkpad t460s,2015

bill,david,Ebay,ms office 2010,2016

…

Loading data

- load data local inpath './purchase_year.txt' overwrite into table purchase;
  - Table content is stored as a text file in HDFS

```
[ec2-user@ip-172-31-52-194 hadoop-2.7.3]$ bin/hadoop fs -ls /user/hive/w
arehouse/inf551.db/purchase
Found 1 items
-rwxrwxr-x   1 ec2-user supergroup        657 2016-11-12 19:11 /user/hiv
e/warehouse/inf551.db/purchase/purchase_year.txt
[ec2-user@ip-172-31-52-194 hadoop-2.7.3]$ bin/hadoop fs -cat /user/hive/
warehouse/inf551.db/purchase/purchase_year.txt
mary,john,Amazon,iphone 6,2016
mary,david,Ebay,ms office 2013,2016
mary,john,Amazon,thinkpad t460s,2015
mary,david,Ebay,db2 v9,2014
bill,john,Amazon,iphone 6,2015
bill,john,Amazon,thinkpad t460s,2015
bill,david,Ebay,ms office 2010,2016
```

Partitioning in Hive

- Consider purchase table
  - Recording sales data in the past several years
- Consider queries such as:
  - find total sales amount of last year
  - find total sales amount since 2015
- These queries examine only part of table

Partitioning in Hive

- create table purchase_part (
  buyer string,
  seller string,
  store string,
  product string)
  partitioned by (year int)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  STORED AS TEXTFILE;

Describe table

- desc purchase_part;

Note year added as last column

```
hive> desc purchase_part;
OK
buyer                   string
seller                  string
store                   string
product                 string
year                    int

# Partition Information
# col_name              data_type               comment

year                    int
Time taken: 0.045 seconds, Fetched: 10 row(s)
```

Loading data by partition

- load data local inpath './purchase_year-2014.txt' overwrite into table purchase_part partition (year = 2014);
  - Input file (purchase_year-2014.txt) contains only year 2014 data
  - Also need to specify which partition (year = 2014) to load into

purchase_year-2014.txt

mary,david,Ebay,db2 v9

mark,john,Amazon,db2 v9

mark,david,Ebay,information server

Loading data by partition

- load data local inpath './purchase_year-2015.txt' overwrite into table purchase_part partition (year = 2015);

- load data local inpath './purchase_year-2016.txt' overwrite into table purchase_part partition (year = 2016);

Loading data by partition

- Partition data stored as a file under a subdirectory (e.g., year=2014)

Table

Partition

Content file

```
[ec2-user@ip-172-31-52-194 hadoop-2.7.3]$ bin/hadoop fs -ls /user/hive/w
arehouse/inf551.db/purchase_part/year=2014/
Found 1 items
-rwxrwxr-x   1 ec2-user supergroup        116 2016-11-12 23:06 /user/hiv
e/warehouse/inf551.db/purchase_part/year=2014/purchase_year-2014.txt
[ec2-user@ip-172-31-52-194 hadoop-2.7.3]$ bin/hadoop fs -cat /user/hive/
warehouse/inf551.db/purchase_part/year=2014/purchase_year-2014.txt
mary,david,Ebay,db2 v9
mark,john,Amazon,db2 v9
mark,david,Ebay,information server
steve,david,Bestbuy,ms office 2013[ec2-user@ip-172-31-52-194 hadoop-2.7.
```

Hive query

- SQL-like, e.g.,

  select * from purchase_part

  where seller = 'john';

```
hive>     select * from purchase_part
    >     where seller = 'john';
OK
mark    john    Amazon  db2 v9  2014
mary    john    Amazon  thinkpad t460s  2015
bill    john    Amazon  iphone 6        2015
bill    john    Amazon  thinkpad t460s  2015
mark    john    Amazon  iphone 5        2015
steve   john    Bestbuy iphone 6        2015
steve   john    Bestbuy thinkpad t460s  2015
steve   john    Bestbuy iphone 5        2015
mary    john    Amazon  iphone 6        2016
Time taken: 0.312 seconds, Fetched: 9 row(s)
```

Write result to local directory

Change this to other directory if needed
(need absolute path)

- insert overwrite

```
local directory '/home/ec2-user/local-out'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
select * from purchase_part
where seller = 'john';
```

MapReduce job generated

```
hive>
    > insert overwrite
    > local directory '/home/ec2-user/local-out'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > select * from purchase_part
    > where seller = 'john';
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in
the future versions. Consider using a different execution engine (i.e. t
ez, spark) or using Hive 1.X releases.
Query ID = ec2-user_20161112235457_106d1cfa-0e8c-4341-8e9f-e22f499c68d6
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2016-11-12 23:54:58,924 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_local661421037_0002
Moving data to local directory /home/ec2-user/local-out
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 3312 HDFS Write: 918 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 1.917 seconds
```

Write result to local directory

- Output file generated by map task

```
[ec2-user@ip-172-31-52-194 local-out]$ ls
000000_0
[ec2-user@ip-172-31-52-194 local-out]$ cat 000000_0
mark,john,Amazon,db2 v9,2014
mary,john,Amazon,thinkpad t460s,2015
bill,john,Amazon,iphone 6,2015
bill,john,Amazon,thinkpad t460s,2015
mark,john,Amazon,iphone 5,2015
steve,john,Bestbuy,iphone 6,2015
steve,john,Bestbuy,thinkpad t460s,2015
steve,john,Bestbuy,iphone 5,2015
mary,john,Amazon,iphone 6,2016
[ec2-user@ip-172-31-52-194 local-out]$
```

Aggregation

- select year, count(*)
  from purchase
  group by year;

```
hive> select year, count(*)
    >        from purchase
    >        group by year;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in
the future versions. Consider using a different execution engine (i.e. t
ez, spark) or using Hive 1.X releases.
Query ID = ec2-user_20161113002832_01f3edd2-956b-4df9-8a55-a0b009622ff7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2016-11-13 00:28:33,717 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local526165384_0011
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 25352 HDFS Write: 4298 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
2014    4
2015    9
2016    6
Time taken: 1.38 seconds, Fetched: 3 row(s)
```

# of reduce tasks = 1

Join

- create table person(
  name string,
  phone string,
  city string)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ',' ;

- load data local inpath './person.txt' overwrite into table person;

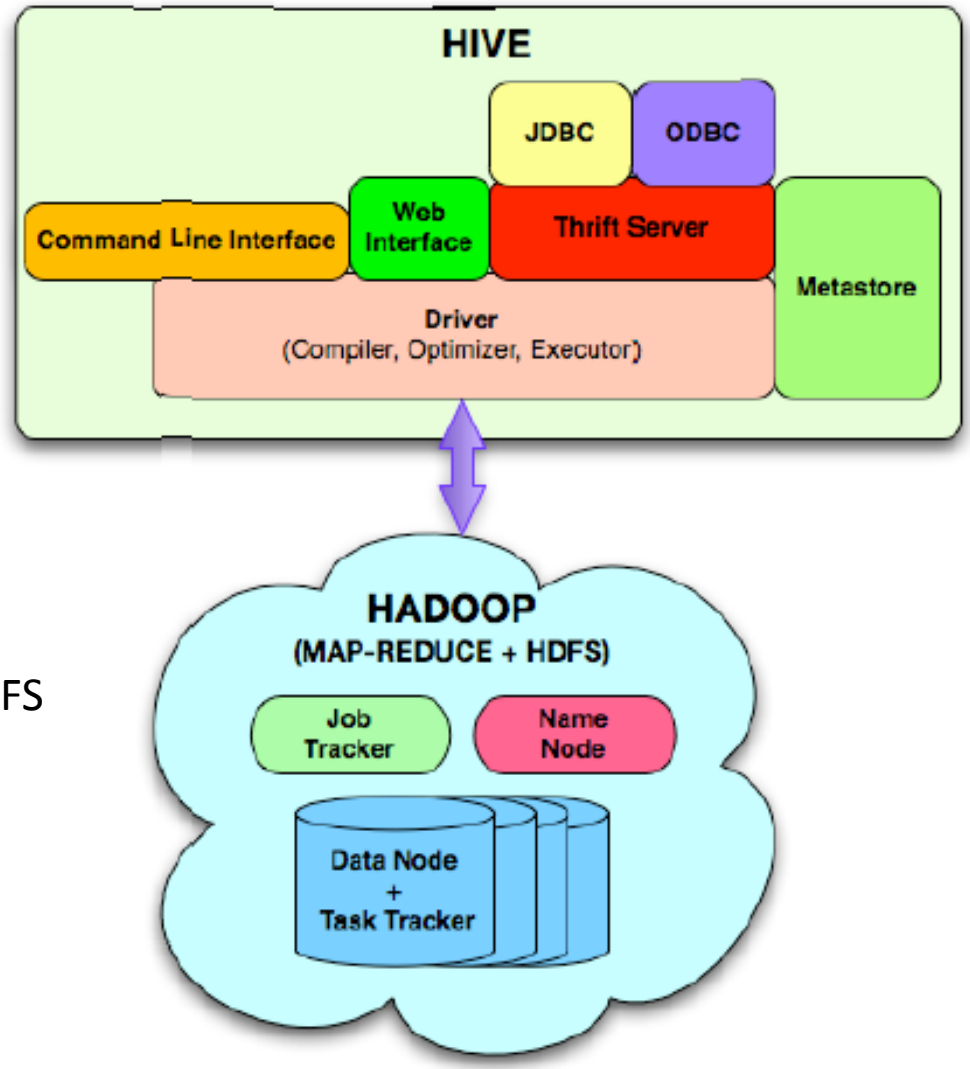
Join

- select buyer, city, product, year
  from person join purchase
  on person.name = purchase.buyer;

Join

```
Total MapReduce CPU Time Spent: 0 msec
OK
mary    los angeles     iphone 6        2016
mary    los angeles     ms office 2013  2016
mary    los angeles     thinkpad t460s  2015
mary    los angeles     db2 v9  2014
bill    los angeles     iphone 6        2015
bill    los angeles     thinkpad t460s  2015
bill    los angeles     ms office 2010  2016
bill    los angeles     windows 10      2016
bill    los angeles     ideapad s50     2015
mark    los angeles     iphone 5        2015
mark    los angeles     ms office 2010  2016
mark    los angeles     ms office 2013  2015
mark    los angeles     db2 v9  2014
mark    los angeles     information server      2014
steve   alhambra        iphone 6        2015
steve   alhambra        ms office 2013  2014
steve   alhambra        thinkpad t460s  2015
steve   alhambra        windows 10      2016
steve   alhambra        iphone 5        2015
Time taken: 11.002 seconds, Fetched: 19 row(s)
```

Architecture



Hive stores data in HDFS

Hive query processing

- Queries compiled into logical data plan
- Logical plan optimized by a rule-based optimizer into a DAG of MapReduce and HDFS (loading and storing data) tasks
- Tasks are executed by executor

Metadata store

- An embedded Apache Derby database
  - an RDBMS implemented in Java
  - stores information about tables, etc.

- E.g., /home/ec2-user/apache-hive-2.1.1-bin/metastore_db

Explain command

```
hive> explain select year, count(*) from purchase group by year;
OK
STAGE DEPENDENCIES:
  Stage-1 is a root stage
  Stage-0 depends on stages: Stage-1

STAGE PLANS:
  Stage: Stage-1
    Map Reduce
      Map Operator Tree:
          TableScan
            alias: purchase
            Statistics: Num rows: 165 Data size: 662 Basic stats: COMPLE
TE Column stats: NONE
            Select Operator
              expressions: year (type: int)
              outputColumnNames: year
              Statistics: Num rows: 165 Data size: 662 Basic stats: COMP
LETE Column stats: NONE
              Group By Operator
                aggregations: count()
                keys: year (type: int)
                mode: hash
                outputColumnNames: _col0, _col1
                Statistics: Num rows: 165 Data size: 662 Basic stats: CO
MPLETE Column stats: NONE
                Reduce Output Operator
                  key expressions: _col0 (type: int)
```

References

- Hive – A Petabyte Scale Data Warehouse Using Hadoop. Thusoo et. al., ICDE 2010.
  - http://infolab.stanford.edu/~ragho/hive-icde2010.pdf

- Hive: getting start guide
  - https://cwiki.apache.org/confluence/display/Hive/GettingStarted