AN INTRODUCTION TO HIVE

Overview

Hive is very similar to Apache Pig. What it does is let you create tables and load external files into tables using SQL. Then it creates MapReduce jobs in Java. Java is a very wordy language so using Pig and Hive is simpler.

Some have said that Hive is a data warehouse tool (Bluntly put, that means an RDBMS used to do analytics before Hadoop was invented.). In fact you can use Apache Sqoop to load data into Hive or Hadoop from a relational database.

(This article is part of our Hadoop Guide. Use the right-hand menu to navigate.)

Installation

In this document we will introduce Hive mainly by using examples. You need to install Apache Hadoop first. Then you can install MySQL to store metadata. (Or you could use Derby. Here we use MySQL). Hive will create data in Hadoop.

So first install Hadoop. Then execute these steps:

```
apt-get install mysql-server
apt-get install libmysql-java
```

Download and extract Hive to /usr/local/hive. Then in .bashrc set:

```
export HIVE_HOME=/usr/local/hive/apache-hive-2.0.1-bin
export PATH=$PATH:$HIVE_HOME/bin
```

Make this softlink:

```
ln -s /usr/share/java/mysql-connector-java.jar $HIVE_HOME/lib/mysql-connector-java.jar
```

Then open MySQL to create the schema,etc:

```
mysql -u root -p
mysql> CREATE DATABASE metastore;
mysql> USE metastore;
```

Here use the schema that most nearly matches the version you are using. (In this tutorial we are using Hive 2.0.1 but there is no 2.0.1 schema. So we used 2.0.0.)

```
mysql> SOURCE /usr/local/hive/apache-hive-2.0.1-bin/scripts/metastore/upgrade/mysql/hive-schema-2.0.0.mysql.sql

mysql> CREATE USER 'hiveuser'@'%' IDENTIFIED BY password';

mysql> GRANT all on *.* to 'hiveuser'@localhost identified by 'password';
mysql>  flush privileges;
```

Now edit $HIVE_HOME/conf/hive-site.xml:

```
        javax.jdo.option.ConnectionURL
        jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true
        metadata is stored in a MySQL server
        javax.jdo.option.ConnectionDriverName
        com.mysql.jdbc.Driver
        MySQL JDBC driver class
        javax.jdo.option.ConnectionUserName
        hiveuser
        user name for connecting to mysql server
        javax.jdo.option.ConnectionPassword
        password
        password for connecting to mysql server
```

If you have set up Hadoop as a cluster you might need to run the next command. You will know if that is the case if you get an error about that.

```
$HADOOP_HOME/bin/hadoop dfsadmin -safemode leave
```

Also if you have installed Spark you could choose to execute this to get rid of a warning message-

```
unset SPARK_HOME
```

Now run the Hive shell:

```
hive
```

Load external CSV File

Much of what you do with Hive is load external files stored in Hadoop so that you can use SQL to work with them. So here is an example.

Download this file (It is an ssh log.) then open it with Google sheets or Excel or use sed and vi to replace the spaces with commas so that it will be comma-delimited.

Then the top of the file looks like this:

```
1331901012,CTHcOo3BARDOPDjYue,192.168.202.68,53633,192.168.28.254,22,failure,
INBOUND,SSH-2.0-OpenSSH_5.0,SSH-1.99-Cisco-1.25,-,-,-,-,-
1331901030,CBHpSz2Zi3rdKbAvwd,192.168.202.68,35820,192.168.23.254,22,failure,
INBOUND,SSH-2.0-OpenSSH_5.0,SSH-1.99-Cisco-1.25,-,-,-,-,-
```

Now copy the file to Hadoop:

```
hadoop fs -put /home/walker/Downloads/ssh.csv /data/ssh/
```

Notice that you put the directory name and not the name of the file in Hadoop.
Then we load the file and create the schema all in one step:

```
CREATE EXTERNAL TABLE IF NOT EXISTS sshlog(
sshtime STRING,
sshkey STRING,
sourceIP STRING,
socket STRING,
targetIP STRING,
port STRING,
status STRING,
direction STRING,
software STRING,
device STRING,
junk1 STRING,
junk2 STRING,
junk3 STRING,
Junk4 STRING,
Junk5 STRING )
   COMMENT 'ssh log'
    ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    STORED AS TEXTFILE
    location '/data/ssh/';
```

Python User Defined Function (UDF)

Before we show some SQL commands (Technically they are called HQL, Hive Query Language.) we

will show how to write a Python UDF:

Create the Python program ssh.py. Below we explain how it works.

```
import sys
        import datetime
        import re

def bi(s):
    r='(\d+)\.(\d+)\.(\d+)\.(\d+)'
    g=re.search('(\d+)\.(\d+)\.(\d+)\.(\d+)',s)
    if g:
        return bin(int(g.group(1))) + bin(int(g.group(2))) +
bin(int(g.group(3)))

for line in sys.stdin:
        x, y, z = line.strip().split('\t')
        print x,bi(y),bi(z)
```

Add it to make it available for Hive to execute this command in the Hive shell.

```
add file /home/walker/Downloads/ssh.py
```

Then run this transform operation calling the Python UDF:

```
SELECT TRANSFORM (sshtime, sourceip, targetip) USING 'python
/home/walker/Downloads/ssh.py' as (sshtime, sourceIP, targetip) FROM sshlog;

1332017778 0b110000000b101010000b11001010 0b110000000b101010000b10101   NULL
NULL
1332017793 0b110000000b101010000b11001010 0b110000000b101010000b10101   NULL
NULL
```

The Python function works off stdin and stdout. We split the input into a time and then two IP address fields. Then we run a function to convert the IP address to bit format. The TRANSFORM statement means pass those three values to the UDF.

Exercise: why does it return 2 null columns?

Create table insert data

No we go back and do something simpler - create a table and insert some data into it to illustrate HQL functions.

First create a table. People who know SQL will see that it is almost the same syntax.

```
create table students (student string, age int);
```

Then add some data into it:

```
insert into table students values('Walker', 33);
insert into table students values('Sam', 33);
```

```
insert into table students values('Sally', 33);
insert into table students values('Sue', 72);
insert into table students values('George', 56);
insert into table students values('William', 64);
insert into table students values('Ellen', 24);
insert into table students values('Jose', 72);
insert into table students values('Li', 56);
insert into table students values('Chris', 64);
insert into table students values('Ellen', 24);
insert into table students values('Sue', 72);
insert into table students values('Ricardo', 56);
insert into table students values('Wolfgang', 64);
insert into table students values('Melanie', 24);
insert into table students values('Monica', 36);
```

Select

Now we can run regular SQL commands over that. Remember this is not a relational database. Hadoop as you will recall does not allow updating files: only adding and deleting them. So it creates new files for every operation.

```
select count(*) from students;
```

Here we can find all students whose birthday is a multiple of 3 by using the modulo (is divisible by) function. (Hive has many math functions.)

```
select age,pmod(age,3) from students where pmod(age,3) = 0;
```

Now, without discriminating against older people, we create two new tables: one with people older than 45 and one for people younger than that:

```
create table old as select * from students where age > 45;

eate table young as select * from students where age < = 45;
```

Join

Now we can make the intersection of two tables by showing students from the old and new table whose name is 3 letters long:

```
select young.student, old.student from young join old on
(length(young.student) = 3) and (length(old.student) = 3);
```

It responds:

```
OK
Sam     Sue
Sam     Sue
```

Maps

Hive supports maps, structs, and array complex types. But it does not support the ability to add data to those with SQL yet. So we show the sort of awkward way of doing that below.

First, create a table with a map column, meaning a (key->value) column:

```
create table prices(product map);
```

Now we use the students table we created above to stand in as a proxy for this insert operation. You can use any table for that

```
insert into prices select map("abc", 1) from students limit 1;
```

Now you can see the data we just inserted:

```
select * from prices;
OK
{"abc":1}
```

Where to go next

From here there are many areas where you could focus your learning as Hive has many features. For example you can learn about partitions, the decimal data type, working with dates, using Hive on Amazon, and using Hive with Apache Spark. You could learn about Beeline which is a newer Hive command line interface. (Beeline will replace the Hive cli in the future.) And you can dig into architectural like topics like SerDe, which is the Hive serializer-deserializer and Hive file storage formats.