BUAN 6346 BIG DATA

DR. JERRY F. PEREZ

Assignment 2

06/16/2020

Jeffrey Chang / jlc190004

Table of Contents of Assignment 2

Lab-Chapter-06-impala-hive 2

Create and Query a Table in Impala or Hive 2

Use sqoop to impoet Directly into Hive and Impala 4

Lab-Chapter-07-file-format 7

Select a Format for a Data File 7

Optional & Optional Homework 11

Lab-Chapter-08-partitions 13

Partition Data in Impala or Hive 13

Lab-Chapter-09-flume 16

Create an HDFS directory for Flume to save ingested data 16

Create a local directory for web server log output 17

Configure Flume 17

Run the Agent 18

Simulate Apache web server output 19

Lab-Chapter-06-Impala-Hive

In this practice, We will define Impala and Hive tables to model view the data in HDFS.

Create and Query a Table in Impala or Hive

We can use both Impala or Hive to query the webpage data. We imported the data from the previous exercise for practice. There are couple ways to interact with Impala and Hive. Now we use Query Editor in Hue to do the Impala or Hive.

If use Hive rather than Impala, start the Hive server, by entering the following two commands in a terminal window.

```
$ sudo service zookeeper-server start
$ sudo service hive-server2 start
```

Visit the Hue page in Firefox. Open either the Impala query editor or Hive query editor, by selecting the editor of your choice from the Query Editors menu.

Enter a SQL command to create a table for the webpage data imported in the previous exercises in the query editor pane. We can see the column definitions in the webpage table on the left.

Using New Query button to test query by input [SELECT * FROM webpage WHERE name LIKE "ifruit%"], and the the result will show below.

Click on the Preview Sample Data icon to view a sampling of the table data.

	page_id	name	assoc_files
0	40	ifruit_4a_sales.html	theme.css,code.js,ifruit_4a.jpg
1	41	sorrento_f31l_sales.html	theme.css,code.js,sorrento_f31l.jpg
2	42	sorrento_f40l_sales.html	theme.css,code.js,sorrento_f40l.jpg
3	43	sorrento_f01l_sales.html	theme.css,code.js,sorrento_f01l.jpg
4	44	sorrento_f11l_sales.html	theme.css,code.js,sorrento_f11l.jpg
5	45	titanic_2400_sales.html	theme.css,code.js,titanic_2400.jpg
6	46	ronin_s4_sales.html	theme.css,code.js,ronin_s4.jpg
7	47	ronin_s1_sales.html	theme.css,code.js,ronin_s1.jpg
8	48	ronin_s3_sales.html	theme.css,code.js,ronin_s3.jpg
9	49	ronin_s2_sales.html	theme.css,code.js,ronin_s2.jpg
10	50	ifruit_5a_sales.html	theme.css,code.js,ifruit_5a.jpg
11	51	contactus.html	theme.css,code.js

Use Sqoop to Import Directly into Hive and Impala

In this part, we use sqoop to import data from MySQL into HDFS and automatically create the corresponding table in the Hive metastore.

- Import the *device* table directly into the Hive Metastore:

[$ sqoop import --connect jdbc:mysql://localhost/loudacre --username training --password training --fields-terminated-by '\t' --table device --hive-import]

```
[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/loudacre --username training --password training --fields-termina
ted-by '\t' --table device --hive-import
20/06/15 21:48:42 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5-cdh5.4.3
20/06/15 21:48:42 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
20/06/15 21:48:43 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
20/06/15 21:48:43 INFO tool.CodeGenTool: Beginning code generation
20/06/15 21:48:43 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `device` AS t LIMIT 1
20/06/15 21:48:43 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `device` AS t LIMIT 1
20/06/15 21:48:43 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-mapreduce
Note: /tmp/sqoop-training/compile/42106aaf462e4d8c4f862da75242a352/device.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
20/06/15 21:48:46 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/42106aaf462e4d8c4f862da75242a352/device.ja
r
20/06/15 21:48:46 WARN manager.MySQLManager: It looks like you are importing from mysql.
20/06/15 21:48:46 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
20/06/15 21:48:46 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
20/06/15 21:48:46 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
20/06/15 21:48:46 INFO mapreduce.ImportJobBase: Beginning import of device
20/06/15 21:48:46 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
20/06/15 21:48:46 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
20/06/15 21:48:47 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
20/06/15 21:48:47 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/06/15 21:48:49 INFO db.DBInputFormat: Using read commited transaction isolation
20/06/15 21:48:49 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`device_num`), MAX(`device_num`) FROM `device`
20/06/15 21:48:49 INFO mapreduce.JobSubmitter: number of splits:4
20/06/15 21:48:49 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1590875116123_0007
20/06/15 21:48:49 INFO impl.YarnClientImpl: Submitted application application_1590875116123_0007
20/06/15 21:48:49 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1590875116123_0007/
20/06/15 21:48:49 INFO mapreduce.Job: Running job: job_1590875116123_0007
20/06/15 21:48:58 INFO mapreduce.Job: Job job_1590875116123_0007 running in uber mode : false
20/06/15 21:48:58 INFO mapreduce.Job:  map 0% reduce 0%
```

```
                Input split bytes=464
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=302
                CPU time spent (ms)=2700
                Physical memory (bytes) snapshot=482361344
                Virtual memory (bytes) snapshot=3378053120
                Total committed heap usage (bytes)=191889408
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=2183
20/06/15 21:49:22 INFO mapreduce.ImportJobBase: Transferred 2.1318 KB in 34.8949 seconds (62.5593 bytes/sec)
20/06/15 21:49:22 INFO mapreduce.ImportJobBase: Retrieved 50 records.
20/06/15 21:49:22 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `device` AS t LIMIT 1
20/06/15 21:49:22 WARN hive.TableDefWriter: Column release_dt had to be cast to a less precise type in Hive
20/06/15 21:49:22 INFO hive.HiveImport: Loading uploaded data into Hive

Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-1.1.0-cdh5.4.3.jar!/hive-log4j.properties
OK
Time taken: 0.96 seconds
Loading data to table default.device
chgrp: changing ownership of 'hdfs://localhost:8020/user/hive/warehouse/device/part-m-00000': User does not belong to hive
chgrp: changing ownership of 'hdfs://localhost:8020/user/hive/warehouse/device/part-m-00001': User does not belong to hive
chgrp: changing ownership of 'hdfs://localhost:8020/user/hive/warehouse/device/part-m-00002': User does not belong to hive
chgrp: changing ownership of 'hdfs://localhost:8020/user/hive/warehouse/device/part-m-00003': User does not belong to hive
Table default.device stats: [numFiles=4, totalSize=2183]
OK
Time taken: 0.512 seconds
[training@localhost ~]$
[training@localhost ~]$ jlc190004
```

Using the HDFS command line, review the imported data files. The Hive import copies the data to the default Hive warehouse location: /user/hive/warehouse/device

```
[training@localhost ~]$ hdfs dfs -ls /user/hive/warehouse/device
Found 4 items
-rwxrwxrwx   1 training supergroup        540 2020-06-15 21:49 /user/hive/warehouse/device/part-m-00000
-rwxrwxrwx   1 training supergroup        561 2020-06-15 21:49 /user/hive/warehouse/device/part-m-00001
-rwxrwxrwx   1 training supergroup        527 2020-06-15 21:49 /user/hive/warehouse/device/part-m-00002
-rwxrwxrwx   1 training supergroup        555 2020-06-15 21:49 /user/hive/warehouse/device/part-m-00003
[training@localhost ~]$
[training@localhost ~]$ jlc190004
```

Using Hue to review the imported data files. I use Impala and refresh the Impala metadata cache by entering the command in the Hue Impala Query Editor: [INVALIDATE METADATA]

Viewing the columns by executing a test query with SQL command.

Lab-Chapter-07-file-format

In this part, we will use Avro to import data and create an Impala or Hive table to access it.

Select a Format for a Data File

1. Change directories to the exercise directory: [cd $DEV1/exercises/data-format/]
2. Import the accounts table to an Avro data format.

[$ sqoop import --connect jdbc:mysql://localhost/loudacre --username training --password training --table accounts --target-dir /loudacre/accounts_avro --null-non-string '\\N' --as-avrodatafile]

```
[training@localhost ~]$ cd $DEV1/exercises/data-format/
[training@localhost data-format]$ sqoop import --connect jdbc:mysql://localhost/loudacre --username training --password training --tabl
e accounts --target-dir /loudacre/accounts_avro --null-non-string '\\N' --as-avrodatafile
20/06/15 22:31:50 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5-cdh5.4.3
20/06/15 22:31:50 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
20/06/15 22:31:51 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
20/06/15 22:31:51 INFO tool.CodeGenTool: Beginning code generation
20/06/15 22:31:51 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `accounts` AS t LIMIT 1
20/06/15 22:31:51 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `accounts` AS t LIMIT 1
20/06/15 22:31:51 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-mapreduce
Note: /tmp/sqoop-training/compile/ccfb9b0d091a2a09af41bd134bad5322/accounts.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
20/06/15 22:31:54 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/ccfb9b0d091a2a09af41bd134bad5322/accounts.
jar
20/06/15 22:31:54 WARN manager.MySQLManager: It looks like you are importing from mysql.
20/06/15 22:31:54 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
20/06/15 22:31:54 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
20/06/15 22:31:54 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
20/06/15 22:31:54 INFO mapreduce.ImportJobBase: Beginning import of accounts
20/06/15 22:31:54 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
20/06/15 22:31:54 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
20/06/15 22:31:55 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `accounts` AS t LIMIT 1
20/06/15 22:31:55 INFO mapreduce.DataDrivenImportJob: Writing Avro schema file: /tmp/sqoop-training/compile/ccfb9b0d091a2a09af41bd134ba
d5322/sqoop_import_accounts.avsc
20/06/15 22:31:55 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
20/06/15 22:31:55 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/06/15 22:31:56 INFO db.DBInputFormat: Using read commited transaction isolation
20/06/15 22:31:56 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`acct_num`), MAX(`acct_num`) FROM `accounts`
20/06/15 22:31:56 INFO mapreduce.JobSubmitter: number of splits:4
20/06/15 22:31:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1590875116123_0008
20/06/15 22:31:57 INFO impl.YarnClientImpl: Submitted application application_1590875116123_0008
20/06/15 22:31:57 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1590875116123_0008/
```

```
                HDFS: Number of bytes written=12713387
                HDFS: Number of read operations=16
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=8
        Job Counters
                Launched map tasks=4
                Other local map tasks=4
                Total time spent by all maps in occupied slots (ms)=0
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=27352
                Total vcore-seconds taken by all map tasks=27352
                Total megabyte-seconds taken by all map tasks=7002112
        Map-Reduce Framework
                Map input records=129767
                Map output records=129767
                Input split bytes=470
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=520
                CPU time spent (ms)=12700
                Physical memory (bytes) snapshot=555237376
                Virtual memory (bytes) snapshot=3383017472
                Total committed heap usage (bytes)=191889408
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=12713387
20/06/15 22:32:40 INFO mapreduce.ImportJobBase: Transferred 12.1244 MB in 44.5891 seconds (278.4409 KB/sec)
20/06/15 22:32:40 INFO mapreduce.ImportJobBase: Retrieved 129767 records.
[training@localhost data-format]$
[training@localhost data-format]$ jlc190004
```

3. View the files imported by Sqoop into HDFS by HDFS command.

Using Hue.

But if we use -cat to view the data, it will come out garbled text.

[hdfs dfs -cat /loudacre/accounts_avro/part-m-00000.avro]

1. Sqoop generated a schema named *sqoop_import_accounts.avsc* in the current directory.

Copy *sqoop_import_accounts.avsc* to the /loudacre directory in HDFS.

2. In Impala or Hive, create a table using this schema:

[CREATE EXTERNAL TABLE accounts_avro
STORED AS AVRO
LOCATION '/loudacre/accounts_avro'
TBLPROPERTIES ('avro.schema.url'='hdfs:/loudacre/sqoop_import_accounts.avsc');]

Confirm correct creation of the table by issuing a query such as:

Optional:

Use the DESCRIBE or DESCRIBE FORMATTED command to list the columns and data types of the accounts_avro table created from the Avro schema.

a. describe accounts_avro;

b. describe formatted accounts_avro;

Optional Homework

Create a new table based on the existing *accounts_avro* table data, using **Parquet** for the storage format.

Lap-Chapter-08-Partitions

Partition Data in Impala or Hive

I will establish and load an Impala (or Hive) table with account data, divided by area code. In the last part, I imported data from the accounts table using Sqoop, into a table called *accounts_avro*. In this part, I will establish a new table with some of the account data, partitioned by area code (the first three digits of the phone number)

1. Create a new, empty table in Impala (or Hive):

```
CREATE EXTERNAL TABLE accounts_by_areacode (
acct_num INT,
first_name STRING,
last_name STRING,
phone_number STRING)
PARTITIONED BY (areacode STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/loudacre/accounts_by_areacode';
```

2. In order to populate the new table, you will need to extract the area code from the phone number. Try executing the following query to demonstrate:

SELECT acct_num, first_name, last_name, phone_number,
SUBSTR(phone_number,1,3) AS areacode
FROM accounts_avro

3. Use the *SELECT* statement above in an *INSERT INTO TABLE* command to copy the specified columns to the new table, dynamically partitioning by area code.

4. Execute a simple query to confirm that the table was populated correctly, such as:
   SELECT * FROM accounts_by_areacode LIMIT 10

5. Using Hue or the *hdfs* command line interface, confirm that the directory structure of the *accounts_by_areacode* table includes partition directories. Review the data in the directories to verify that the partitioning is correct.

Lap-Chapter-09-Flume

Apache Flume Introduction

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application. *(From Open Source Apache Flume website: https://flume.apache.org/)*

In this part, I will configure Flume to ingest web log data from a local directory to HDFS. Apache web server logs are generally stored in files on the local machines running the server. In this part, I will simulate an Apache server by placing provided web log files into a local spool directory, and then use Flume to collect the data.

Both the local and HDFS directories must exist before using the spooling directory source.

Create an HDFS directory for Flume to save ingested data

1. Create a directory in HDFS called */loudacre/weblogs* to hold the data files Flume ingests.
   [$ hdfs dfs -mkdir /loudacre/weblogs]

Create a local directory for web server log output

2. Let's Creating the spool directory into which our web log simulator will store data files for Flume to ingest.

On the local filesystem create */flume/weblogs_spooldir*.

[$ sudo mkdir -p /flume/weblogs_spooldir]

We set all users the permissions to write to the */flume/weblogs_spooldir* directory.

[$ sudo chmod a+w -R /flume]

```
[training@localhost ~]$ hdfs dfs -mkdir /loudacre/weblogs
[training@localhost ~]$ hdfs dfs -ls /loudacre/
Found 8 items
drwxrwxrwx   - training supergroup          0 2020-05-31 16:09 /loudacre/accounts
drwxrwxrwx   - training supergroup          0 2020-06-16 00:11 /loudacre/accounts_avro
drwxrwxrwx   - impala   supergroup          0 2020-06-16 00:49 /loudacre/accounts_by_areacode
-rw-rw-rw-   1 training supergroup      15191 2020-05-30 19:30 /loudacre/base_stations.tsv
drwxrwxrwx   - training supergroup          0 2020-05-30 18:34 /loudacre/kb
-rw-rw-rw-   1 training supergroup       1574 2020-06-15 23:32 /loudacre/sqoop_import_accounts.avsc
drwxrwxrwx   - training supergroup          0 2020-06-16 01:11 /loudacre/weblogs
drwxrwxrwx   - training supergroup          0 2020-05-31 16:27 /loudacre/webpage
[training@localhost ~]$
[training@localhost ~]$ sudo mkdir -p /flume/weblogs_spooldir
[training@localhost ~]$ sudo chmod a+w -R /flume
[training@localhost ~]$ hdfs dfs -ls /loudacre/weblogs
[training@localhost ~]$ hdfs dfs -ls /
Found 7 items
drwxrwxrwx   - hbase    supergroup          0 2015-08-27 07:37 /hbase
drwxrwxrwx   - training supergroup          0 2020-06-16 01:11 /loudacre
drwxrwxrwx   - training supergroup          0 2020-06-01 21:19 /optional_HW
drwxrwxrwx   - solr     supergroup          0 2015-08-27 07:32 /solr
drwxrwxrwt   - hdfs     supergroup          0 2020-06-15 21:19 /tmp
drwxrwxrwx   - hdfs     supergroup          0 2020-05-30 19:24 /user
drwxrwxrwx   - hdfs     supergroup          0 2015-08-27 07:31 /var
[training@localhost ~]$ jlc190004
```

Configure Flume

I follow the instruction, there are my steps of my work.

1. I create the configuration file called = ***spooldir.conf*** under ***/home/training/training_materials/dev1/exercises/flume***
2. ***spooldir.conf*** contents and characteristics are listed below:

```
# Name the components on this agent
agent1.sources = webserver-log-source
agent1.sinks = hdfs-sink
agent1.channels = memory-channel

# Describe/configure the source
agent1.sources.webserver-log-source.type = spooldir
agent1.sources.webserver-log-source.spoolDir = /flume/weblogs_spooldir
agent1.sources.webserver-log-source.channels = memory-channel
```

```
# Describe the sink
agent1.sinks.hdfs-sink.type = hdfs
agent1.sinks.hdfs-sink.hdfs.path = /loudacre/weblogs/
agent1.sinks.hdfs-sink.channel = memory-channel
agent1.sinks.hdfs-sink.hdfs.rollInterval = 0
agent1.sinks.hdfs-sink.hdfs.rollSize = 524288
agent1.sinks.hdfs-sink.hdfs.rollCount = 0
agent1.sinks.hdfs-sink.hdfs.fileType = DataStream

# Use a channel which buffers events in memory
agent1.channels.memory-channel.type = memory
agent1.channels.memory-channel.capacity = 10000
agent1.channels.memory-channel.transactionCapacity = 10000
```

Run the Agent

Use the commands:

[$ flume-ng agent --conf /etc/flume-ng/conf --conf-file spooldir.conf --name agent1 -Dflume.root.logger=INFO,console]

Simulate Apache web server output

- Open a separate terminal window, and change to the exercise directory. Run the script to place the web log files in the */flume/weblogs_spooldir* directory

Return to the terminal that is running the Flume agent and watch the logging output.

- The output will give information about the files Flume is putting into HDFS.
- Once the Flume agent has finished, enter CTRL+C to terminate the process.

Using the hdfs command line or Hue File Browser, list the files in HDFS that were added by the Flume agent. Note that the files that were imported are tagged with a Unix timestamp corresponding to the time the file was imported, e.g. FlumeData.1427214989392