Apache Hadoop –

A course for undergraduates

Homework Labs, Lecture 9

Notes for Upcoming Labs

VM Services Customization

For the remainder of the labs, there are services that must be running in your VM, and others that are optional. It is strongly recommended that you run the following command whenever you start the VM:

```
$ ~/scripts/analyst/toggle_services.sh
```

This will conserve memory and increase performance of the virtual machine. After running this command, you may safely ignore any messages about services that have already been started or shut down.

Data Model Reference

For your convenience, you will find a reference document depicting the structure for the tables you will use in the following labs. See file: Homework_DataModelRef.docx

Regular Expression (Regex) Reference

For your convenience, you will find a reference document describing regular expressions syntax. See file: Homework_RegexRef.docx

Lab: Data Ingest With Hadoop Tools

In this lab you will practice using the Hadoop command line utility to interact with Hadoop's Distributed Filesystem (HDFS) and use Sqoop to import tables from a relational database to HDFS.

Prepare your Virtual Machine

Launch the VM if you haven't already done so, and then run the following command to boost performance by disabling services that are not needed for this class:

```
$ ~/scripts/analyst/toggle_services.sh
```

Step 1: Setup HDFS

1. Open a terminal window (if one is not already open) by double-clicking the Terminal icon on the desktop. Next, change to the directory for this lab by running the following command:

```
$ cd $ADIR/exercises/data_ingest
```

2. To see the contents of your home directory, run the following command:

```
$ hadoop fs -ls /user/training
```

3. If you do not specify a path, `hadoop fs` assumes you are referring to your home directory. Therefore, the following command is equivalent to the one above:

```
$ hadoop fs -ls
```

4. Most of your work will be in the `/dualcore` directory, so create that now:

```
$ hadoop fs -mkdir /dualcore
```

Step 2: Importing Database Tables into HDFS with Sqoop

Dualcore stores information about its employees, customers, products, and orders in a MySQL database. In the next few steps, you will examine this database before using Sqoop to import its tables into HDFS.

1. Log in to MySQL and select the `dualcore` database:

```
$ mysql --user=training --password=training dualcore
```

2. Next, list the available tables in the `dualcore` database (`mysql>` represents the MySQL client prompt and is not part of the command):

```
mysql> SHOW TABLES;
```

3. Review the structure of the `employees` table and examine a few of its records:

```
mysql> DESCRIBE employees;
mysql> SELECT emp_id, fname, lname, state, salary FROM
employees LIMIT 10;
```

4. Exit MySQL by typing `quit`, and then hit the enter key:

```
mysql> quit
```

5. Next, run the following command, which imports the `employees` table into the `/dualcore` directory created earlier using tab characters to separate each field:

```
$ sqoop import \
 --connect jdbc:mysql://localhost/dualcore \
 --username training --password training \
 --fields-terminated-by '\t' \
 --warehouse-dir /dualcore \
 --table employees
```

6. Revise the previous command and import the `customers` table into HDFS.

7. Revise the previous command and import the `products` table into HDFS.

8. Revise the previous command and import the `orders` table into HDFS.

9. Next, you will import the `order_details` table into HDFS. The command is slightly different because this table only holds references to records in the `orders` and `products` table, and lacks a primary key of its own. Consequently, you will need to specify the `--split-by` option and instruct Sqoop to divide the import work among map tasks based on values in the `order_id` field. An alternative is to use the `-m 1` option to force Sqoop to import all the data with a single task, but this would significantly reduce performance.

```
$ sqoop import \
 --connect jdbc:mysql://localhost/dualcore \
 --username training --password training \
 --fields-terminated-by '\t' \
 --warehouse-dir /dualcore \
 --table order_details \
 --split-by=order_id
```

This is the end of the lab.

Lab: Using Pig for ETL Processing

In this lab you will practice using Pig to explore, correct, and reorder data in files from two different ad networks. You will first experiment with small samples of this data using Pig in local mode, and once you are confident that your ETL scripts work as you expect, you will use them to process the complete data sets in HDFS by using Pig in MapReduce mode.

IMPORTANT: Since this lab builds on the previous one, it is important that you successfully complete the previous lab before starting this lab.

Background Information

Dualcore has recently started using online advertisements to attract new customers to its e-commerce site. Each of the two ad networks they use provides data about the ads they've placed. This includes the site where the ad was placed, the date when it was placed, what keywords triggered its display, whether the user clicked the ad, and the per-click cost.

Unfortunately, the data from each network is in a different format. Each file also contains some invalid records. Before we can analyze the data, we must first correct these problems by using Pig to:

- Filter invalid records
- Reorder fields
- Correct inconsistencies
- Write the corrected data to HDFS

Step #1: Working in the Grunt Shell

In this step, you will practice running Pig commands in the Grunt shell.

1. Change to the directory for this lab:

```
$ cd $ADIR/exercises/pig_etl
```

2. Copy a small number of records from the input file to another file on the local file system. When you start Pig, you will run in local mode. For testing, you can work faster with small local files than large files in HDFS.

It is not essential to choose a random sample here – just a handful of records in the correct format will suffice. Use the command below to capture the first 25 records so you have enough to test your script:

```
$ head -n 25 $ADIR/data/ad_data1.txt > sample1.txt
```

3. Start the Grunt shell in local mode so that you can work with the local `sample1.txt` file.

```
$ pig -x local
```

A prompt indicates that you are now in the Grunt shell:

```
grunt>
```

4. Load the data in the `sample1.txt` file into Pig and dump it:

```
grunt> data = LOAD 'sample1.txt';
grunt> DUMP data;
```

You should see the 25 records that comprise the sample data file.

5. Load the first two columns' data from the sample file as character data, and then dump that data:

```
grunt> first_2_columns = LOAD 'sample1.txt' AS
       (keyword:chararray, campaign_id:chararray);
grunt> DUMP first_2_columns;
```

6. Use the `DESCRIBE` command in Pig to review the schema of `first_2_cols`:

```
 grunt> DESCRIBE first_2_columns;
```

   The schema appears in the Grunt shell.

   Use the `DESCRIBE` command while performing these labs any time you would like to review schema definitions.

7. See what happens if you run the `DESCRIBE` command on `data`. Recall that when you loaded `data`, you did *not* define a schema.

```
grunt> DESCRIBE data;
```

8. End your Grunt shell session:

```
grunt> QUIT;
```

Step #2: Processing Input Data from the First Ad Network

In this step, you will process the input data from the first ad network. First, you will create a Pig script in a file, and then you will run the script. Many people find working this way easier than working directly in the Grunt shell.

1. Edit the `first_etl.pig` file to complete the `LOAD` statement and read the data from the sample you just created. The following table shows the format of the data in the file. For simplicity, you should leave the `date` and `time` fields separate, so each will be of type `chararray`, rather than converting them to a

single field of type `datetime`.

Index	Field	Data Type	Description	Example
0	keyword	chararray	Keyword that triggered ad	`tablet`
1	campaign_id	chararray	Uniquely identifies the ad	`A3`
2	date	chararray	Date of ad display	`05/29/2013`
3	time	chararray	Time of ad display	`15:49:21`
4	display_site	chararray	Domain where ad shown	`www.example.com`
5	was_clicked	int	Whether ad was clicked	`1`
6	cpc	int	Cost per click, in cents	`106`
7	country	chararray	Name of country in which ad ran	`USA`
8	placement	chararray	Where on page was ad displayed	`TOP`

2. Once you have edited the `LOAD` statement, try it out by running your script in local mode:

```
$ pig -x local first_etl.pig
```

Make sure the output looks correct (i.e., that you have the fields in the expected order and the values appear similar in format to that shown in the table above) before you continue with the next step.

3. Make each of the following changes, running your script in local mode after each one to verify that your change is correct:

a. Update your script to filter out all records where the country field does not contain `USA`.

b. We need to store the fields in a different order than we received them. Use a `FOREACH` … `GENERATE` statement to create a new relation containing the fields in the same order as shown in the following table (the `country` field is not included since all records now have the same value):

Index	Field	Description
0	campaign_id	Uniquely identifies the ad
1	date	Date of ad display
2	time	Time of ad display
3	keyword	Keyword that triggered ad

4	display_site	Domain where ad shown
5	placement	Where on page was ad displayed
6	was_clicked	Whether ad was clicked
7	cpc	Cost per click, in cents

c. Update your script to convert the `keyword` field to uppercase and to remove any leading or trailing whitespace (Hint: You can nest calls to the two built-in functions inside the `FOREACH` ... `GENERATE` statement from the last statement).

4. Add the complete data file to HDFS:

```
$ hadoop fs -put $ADIR/data/ad_data1.txt /dualcore
```

5. Edit `first_etl.pig` and change the path in the `LOAD` statement to match the path of the file you just added to HDFS (`/dualcore/ad_data1.txt`).

6. Next, replace `DUMP` with a `STORE` statement that will write the output of your processing as tab-delimited records to the `/dualcore/ad_data1` directory.

7. Run this script in Pig's MapReduce mode to analyze the entire file in HDFS:

```
$ pig first_etl.pig
```

If your script fails, check your code carefully, fix the error, and then try running it again. Don't forget that you must remove output in HDFS from a previous run before you execute the script again.

8. Check the first 20 output records that your script wrote to HDFS and ensure they look correct (you can ignore the message "cat: Unable to write to output stream"; this simply happens because you are writing more data with the `fs -cat` command than you are reading with the `head` command):

```
$ hadoop fs -cat /dualcore/ad_data1/part* | head -20
```

a. Are the fields in the correct order?

b. Are all the keywords now in uppercase?

Step #3: Processing Input Data from the Second Ad Network

Now that you have successfully processed the data from the first ad network, continue by processing data from the second one.

1. Create a small sample of the data from the second ad network that you can test locally while you develop your script:

```
$ head -n 25 $ADIR/data/ad_data2.txt > sample2.txt
```

2. Edit the `second_etl.pig` file to complete the `LOAD` statement and read the data from the sample you just created (Hint: The fields are comma-delimited). The following table shows the order of fields in this file:

Index	Field	Data Type	Description	Example
0	campaign_id	chararray	Uniquely identifies the ad	`A3`
1	date	chararray	Date of ad display	`05/29/2013`
2	time	chararray	Time of ad display	`15:49:21`
3	display_site	chararray	Domain where ad shown	`www.example.com`
4	placement	chararray	Where on page was ad displayed	`TOP`
5	was_clicked	int	Whether ad was clicked	`Y`
6	cpc	int	Cost per click, in cents	`106`
7	keyword	chararray	Keyword that triggered ad	`tablet`

3. Once you have edited the `LOAD` statement, use the `DESCRIBE` keyword and then run your script in local mode to check that the schema matches the table above:

```
$ pig -x local second_etl.pig
```

4. Replace `DESCRIBE` with a `DUMP` statement and then make each of the following changes to `second_etl.pig`, running this script in local mode after each change to verify what you've done before you continue with the next step:

d. This ad network sometimes logs a given record twice. Add a statement to the `second_etl.pig` file so that you remove any duplicate records. If you have done this correctly, you should only see one record where the `display_site` field has a value of `siliconwire.example.com`.

e. As before, you need to store the fields in a different order than you received them. Use a `FOREACH` ... `GENERATE` statement to create a new relation containing the fields in the same order you used to write the output from first ad network (shown again in the table below) and also use the `UPPER` and `TRIM` functions to correct the `keyword` field as you did earlier:

Index	Field	Description
0	campaign_id	Uniquely identifies the ad
1	date	Date of ad display
2	time	Time of ad display
3	keyword	Keyword that triggered ad
4	display_site	Domain where ad shown
5	placement	Where on page was ad displayed
6	was_clicked	Whether ad was clicked
7	cpc	Cost per click, in cents

f. The date field in this data set is in the format `MM-DD-YYYY`, while the data you previously wrote is in the format `MM/DD/YYYY`. Edit the `FOREACH` ... `GENERATE` statement to call the `REPLACE(date, '-', '/')` function to correct this.

5. Once you are sure the script works locally, add the full data set to HDFS:

```
$ hadoop fs -put $ADIR/data/ad_data2.txt /dualcore
```

6. Edit the script to have it `LOAD` the file you just added to HDFS, and then replace the `DUMP` statement with a `STORE` statement to write your output as tab-delimited records to the `/dualcore/ad_data2` directory.

7. Run your script against the data you added to HDFS:

```
$ pig second_etl.pig
```

8. Check the first 15 output records written in HDFS by your script:

```
$ hadoop fs -cat /dualcore/ad_data2/part* | head -15
```

   a. Do you see any duplicate records?

   b. Are the fields in the correct order?

   c. Are all the keywords in uppercase?

   d. Is the date field in the correct (`MM/DD/YYYY`) format?

This is the end of the lab.

Lab: Analyzing Ad Campaign Data with Pig

During the previous lab, you performed ETL processing on data sets from two online ad networks. In this lab, you will write Pig scripts that analyze this data to optimize advertising, helping Dualcore to save money and attract new customers.

IMPORTANT: Since this lab builds on the previous one, it is important that you successfully complete the previous lab before starting this lab.

Step #1: Find Low Cost Sites

Both ad networks charge a fee only when a user clicks on Dualcore's ad. This is ideal for Dualcore since their goal is to bring new customers to their site. However, some sites and keywords are more effective than others at attracting people interested in the new tablet being advertised by Dualcore. With this in mind, you will begin by identifying which sites have the lowest total cost.

1. Change to the directory for this lab:

```
$ cd $ADIR/exercises/analyze_ads
```

2. Obtain a local subset of the input data by running the following command:

```
$ hadoop fs -cat /dualcore/ad_data1/part* \
| head -n 100 > test_ad_data.txt
```

You can ignore the message "cat: Unable to write to output stream," which appears because you are writing more data with the `fs -cat` command than you are reading with the `head` command.

Note: As mentioned in the previous lab, it is faster to test Pig scripts by using a local subset of the input data. Although explicit steps are not provided for creating local data subsets in upcoming labs, doing so will help you perform the labs more quickly.

3. Open the `low_cost_sites.pig` file in your editor, and then make the following changes:

    a. Modify the `LOAD` statement to read the sample data in the `test_ad_data.txt` file.

    b. Add a line that creates a new relation to include only records where `was_clicked` has a value of `1`.

    c. Group this filtered relation by the `display_site` field.

    d. Create a new relation that includes two fields: the `display_site` and the total cost of all clicks on that site.

    e. Sort that new relation by cost (in ascending order)

    f. Display just the first three records to the screen

4. Once you have made these changes, try running your script against the sample data:

```
$ pig -x local low_cost_sites.pig
```

5. In the `LOAD` statement, replace the `test_ad_data.txt` file with a file glob (pattern) that will load both the `/dualcore/ad_data1` and `/dualcore/ad_data2` directories (and does *not* load any other data, such as the text files from the previous lab).

6. Once you have made these changes, try running your script against the data in HDFS:

```
$ pig low_cost_sites.pig
```

Question: Which three sites have the lowest overall cost?

Step #2: Find High Cost Keywords

The terms users type when doing searches may prompt the site to display a Dualcore advertisement. Since online advertisers compete for the same set of keywords, some of them cost more than others. You will now write some Pig Latin to determine which keywords have been the most expensive for Dualcore overall.

1. Since this will be a slight variation on the code you have just written, copy that file as `high_cost_keywords.pig`:

```
$ cp low_cost_sites.pig high_cost_keywords.pig
```

2. Edit the `high_cost_keywords.pig` file and make the following three changes:

   a. Group by the `keyword` field instead of `display_site`

   b. Sort in descending order of cost

   c. Display the top five results to the screen instead of the top three as before

3. Once you have made these changes, try running your script against the data in HDFS:

```
$ pig high_cost_keywords.pig
```

Question: Which five keywords have the highest overall cost?

Bonus Lab #1: Count Ad Clicks

One important statistic we haven't yet calculated is the total number of clicks the ads have received. Doing so will help the marketing director plan the next ad campaign budget.

1. Change to the `bonus_01` subdirectory of the current lab:

```
$ cd bonus_01
```

2. Edit the `total_click_count.pig` file and implement the following:

   a. Group the records (filtered by `was_clicked == 1`) so that you can call the aggregate function in the next step.

   b. Invoke the `COUNT` function to calculate the total of clicked ads (Hint: Because we shouldn't have any null records, you can use the `COUNT` function instead of `COUNT_STAR`, and the choice of field you supply to the function is arbitrary).

   c. Display the result to the screen

3. Once you have made these changes, try running your script against the data in HDFS:

```
$ pig total_click_count.pig
```

Question: How many clicks did we receive?

Bonus Lab #2: Estimate The Maximum Cost of The Next Ad Campaign

When you reported the total number of clicks, the Marketing Director said that the goal is to get about three times that amount during the next campaign. Unfortunately, because the cost is based on the site and keyword, it isn't clear how much to budget for that campaign. You can help by estimating the worst case (most expensive) cost based on 50,000 clicks. You will do this by finding the most expensive ad and then multiplying it by the number of clicks desired in the next campaign.

1. Because this code will be similar to the code you wrote in the previous step, start by copying that file as `project_next_campaign_cost.pig`:

```
$ cp total_click_count.pig
project_next_campaign_cost.pig
```

2. Edit the `project_next_campaign_cost.pig` file and make the following modifications:

a. Since you are trying to determine the highest possible cost, you should not limit your calculation to the cost for ads actually clicked. Remove the `FILTER` statement so that you consider the possibility that any ad might be clicked.

b. Change the aggregate function to the one that returns the maximum value in the `cpc` field (Hint: Don't forget to change the name of the relation this field belongs to, in order to account for the removal of the `FILTER` statement in the previous step).

c. Modify your `FOREACH...GENERATE` statement to multiply the value returned by the aggregate function by the total number of clicks we expect to have in the next campaign

d. Display the resulting value to the screen.

3. Once you have made these changes, try running your script against the data in HDFS:

```
$ pig project_next_campaign_cost.pig
```

Question: What is the maximum you expect this campaign might cost?

Bonus Lab #3: Calculating Click-Through Rate (CTR)

The calculations you did at the start of this lab provided a rough idea about the success of the ad campaign, but didn't account for the fact that some sites display Dualcore's ads more than others. This makes it difficult to determine how effective their ads were by simply counting the number of clicks on one site and comparing it to the number of clicks on another site. One metric that would allow Dualcore to better make such comparisons is the Click-Through Rate (`http://tiny.cloudera.com/ade03a`), commonly abbreviated as CTR. This value is simply the percentage of ads shown that users actually clicked, and can be calculated by dividing the number of clicks by the total number of ads shown.

1. Change to the `bonus_03` subdirectory of the current lab:

```
$ cd ../bonus_03
```

2. Edit the `lowest_ctr_by_site.pig` file and implement the following:

a. Within the nested `FOREACH`, filter the records to include only records where the ad was clicked.

b. Create a new relation on the line that follows the `FILTER` statement which counts the number of records within the current group

c. Add another line below that to calculate the click-through rate in a new field named `ctr`

d. After the nested `FOREACH`, sort the records in ascending order of clickthrough rate and display the first three to the screen.

3. Once you have made these changes, try running your script against the data in HDFS:

```
$ pig lowest_ctr_by_site.pig
```

Question: Which three sites have the lowest click through rate?

If you still have time remaining, modify your script to display the three keywords with the highest click-through rate.

This is the end of the lab.