DA 450 Apache Pig Essentials Lab Guide

Fall 2015

Using This Guide

Icons Used in the Guide

This lab guide uses the following ions to draw attention to different types of information:

Note: Additional information that will clarify something, provide details, or help you avoid mistakes.

CAUTION: Details you **must** read to avoid potentially serious problems.

Q&A: A question posed to the learner during a lab exercise.

Try This! Exercises you can complete after class (or during class if you finish a lab early) to strengthen learning.

Lab Files

Here is a description of the materials (data files, licenses etc.) that are supplied to support the labs. These materials should be downloaded to your system before beginning the labs.

DA450_LabFiles.zip	Scripts, data, and other files needed to complete this lab
DA450_LABGUIDE.pdf	Exercises, answer keys, and information about this lab
CONNECT_TO_MAPR_ SANDBOX.pdf	Description of how to connect to your Sandbox or AWS cluster and copy necessary files to your node or cluster

Get Started

Lab Overview

This short exercise helps you set up the environment for the rest of the day's labs. We'll begin by connecting to the MapR Cluster (MapR Sandbox or AWS) and transferring the DA450_LabFiles files to the Cluster. These instructions assume you have basic familiarity with working in a command line interface.

Part 1: Prepare the MapR Cluster (Sandbox or AWS) for Lab

Part 2: Load the Training Files

Lab Procedure

Part 1: Connect to the MapR Cluster (Sandbox or AWS) for Lab

Refer to the CONNECT TO MAPR.pdf file for instructions on how to connect to the MapR Sandbox or AWS Cluster for this lab.

Part 2: Load the training files

The training materials include a file called DA450_LABFILES.zip, which contains data, source code, reference materials, and utilities that you will use to complete the lab exercises. Refer to the CONNECT TO MAPR.pdf file for instructions on how to copy DA450_LabFiles.zip from your computer to the cluster you connected in Part 1.

Verify the files were copied to your MapR Sandbox or AWS Cluster correctly using `ls`, then unpack them using `unzip`.

```
[user01@maprdemo ~]$ ls
[user01@maprdemo ~]$ unzip DA450_LabFiles.zip
```

Here you will not need to replace the username or IP address, as these should appear at the prompt in the terminal if you are connected to the Sandbox or AWS cluster properly. Change the working directory to the DA450_LabFiles folder using `cd DA450_LabFiles`, and then list the files using `ls`.

```
[user01@maprdemo ~]$ cd DA450_LabFiles
[user01@maprdemo ~]$ ls
```


Lesson 1: Pig in the Hadoop Ecosystem

Lab Overview

This short exercise introduces you to the Pig command line interface. We will connect to Pig using grunt and run a few simple commands. These instructions assume you have basic familiarity with working in a command line interface.

Lab Procedure

1. Open a shell. Log in to your MapR Sandbox or AWS Cluster using the procedure outlined above.
2. Connect using the Grunt shell by typing pig.

   ```
   $ pig
   ```

3. Use the help command to show what's available at the grunt shell prompt.

   ```
   grunt> help
   ```

   You should see something like this:

   ```
   grunt> help
   Commands:
   <pig latin statement>; - See the PigLatin manual for details: http://hadoop.apache.org/pig
   File system commands:
       fs <fs arguments> - Equivalent to Hadoop dfs command: http://hadoop.apache.org/common/docs/current/hdfs_shell.html
   Diagnostic commands:
       describe <alias>[::<alias] - Show the schema for the alias. Inner aliases can be described as A::B.
       explain [-script <pigscript>] [-out <path>] [-brief] [-dot|-xml] [-param <param_name>=<param_value>]
           [-param_file <file_name>] [<alias>] - Show the execution plan to compute the alias or for entire script.
           -script - Explain the entire script.
           -out - Store the output into directory rather than print to stdout.
           -brief - Don't expand nested plans (presenting a smaller graph for overview).
           -dot - Generate the output in .dot format. Default is text format.
           -xml - Generate the output in .xml format. Default is text format.
           -param <param_name - See parameter substitution for details.
           -param_file <file_name> - See parameter substitution for details.
           alias - Alias to explain.
       dump <alias> - Compute the alias and writes the results to stdout.
   ```

4. The `quit` command exits the grunt shell.

   ```
   grunt> quit
   ```


Lesson 2: Extract, Transform, Load

Lab Overview

The purpose of these labs is to gain experience with data manipulation in Pig. For each exercise, we will load data into a Pig relation, and build a data flow using that data to illustrate the concepts of extracting, transforming, and loading (ETL) data. This lab will use real weather data from a select number of research stations throughout the continent of Antarctica. The original files can be downloaded from the British Antarctic Survey at https://legacy.bas.ac.uk/met/READER/data.html.

- `LOAD`
- `DESCRIBE and ILLUSTRATE`
- `FOREACH…GENERATE`
- `STORE`

Lab 2.1 Procedure

1. **Start the grunt shell**

   Log in to your assigned cluster and start the grunt using the procedure from Lesson 1.

2. **Load the temperature data file**

   In this step, we'll use the `LOAD` command to load data into a Pig relation that we'll name `TEMPERATURE`. The data file for this table has been provided in the `DA450_LabFiles` folder, and is called `raw_temperature.csv`. This file has fourteen columns. The first is the station name, the second is the year, and the other twelve are the months. Each cell is the average temperature of that month and year at that station in degrees Celsius. To load this file, use:

   ```
   grunt> TEMPERATURE = LOAD
   '/user/user01/DA450_LabFiles/raw_temperature.csv' USING
   PigStorage(',') as (station:chararray, year:int, jan:float,
   feb:float, mar:float, apr: float, may: float, jun:float,
   jul:float, aug:float, sep:float, oct:float, nov:float,
   dec:float);
   ```

   Make sure you replace the user name with your own in the file path!

Note: Like Hive and SQL, Pig functions are typed in all caps as a convention, but are actually case-insensitive. However, relation names are case-sensitive. Unlike Hive, Pig does not require a terminating semicolon for every command. However, using the semicolon is a convention of Pig Latin. Since some commands require the semicolon and relation names are case-sensitive, it's best use to the standard conventions.

You can verify that the data loaded by using the `DUMP` command. Running this command will trigger a MapReduce process and may take a few moments.

```
grunt> DUMP TEMPERATURE;
```

```
grunt> DUMP TEMPERATURE;
(Signy,1947,0.8,0.9,0.1,-0.9,-7.7,-10.1,-10.2,-8.2,-3.9,-1.3,0.1,-1.6)
(Signy,1948,0.7,-0.3,-0.7,-3.7,-6.3,-4.8,-13.3,-12.4,-7.5,-2.1,-3.9,-0.2)
(Signy,1949,-0.3,-0.9,-1.1,-8.6,-8.2,-13.1,-14.6,-11.4,-7.4,-2.9,-2.4,-0.6)
```

You may notice that there are many missing values, as well as both positive and negative numbers throughout the data.

Lab 2.2 Procedure

3. Use `DESCRIBE` and `ILLUSTRATE` to examine the relation

The `DESCRIBE` command shows the schema of Pig relations. Try it now:

```
grunt> DESCRIBE TEMPERATURE;
```

```
grunt> DESCRIBE TEMPERATURE;
TEMPERATURE: {station: chararray,year: int,jan: float,feb: float,mar: float,apr: float,may:
 float,jun: float,jul: float,aug: float,sep: float,oct: float,nov: float,dec: float}
```

The `ILLUSTRATE` command shows the schema as well as a sample of the data in a relation in a formatted display. Try this now:

```
grunt> ILLUSTRATE TEMPERATURE;
```

```
grunt> ILLUSTRATE TEMPERATURE;
-------------------------------------------------------------------------------------------
| TEMPERATURE     | station:chararray     | year:int     | jan:float     | feb:float     | mar:float
-------------------------------------------------------------------------------------------
|                 | Faraday               | 1997         | 1.7           | 0.7           | -0.2
-------------------------------------------------------------------------------------------
```

Lab 2.3 Procedure

4. Transform the TEMPERATURE relation

Let's try a few transformations on the data. Rather than have each row represent an entire year's data, let's restructure this table so that each row corresponds to a single data point. First, let's create a new relation with all the data from the month of January:

```
grunt> JAN = FOREACH TEMPERATURE GENERATE station, year,
'jan' AS month, jan AS celsius;
grunt> DUMP JAN;
```

Note: You can also refer to column names using a dollar sign, followed by the column number, counting from zero. Since the `station` column is `$0`, the `year` column is `$1`, and the `jan` column is `$2`, the transformation above could be rewritten as:

```
grunt> JAN = FOREACH TEMPERATURE GENERATE $0 as
station, $1 as year, 'jan' AS month, $2 as celsius;
```

Once you transform the data by every month in this manner, you can append all the data into one relation using `UNION, then order it by year`:

```
grunt> TEMPERATURE2 = UNION JAN, FEB, MAR, APR, MAY, JUN,
JUL, AUG, SEP, OCT, NOV, DEC;
```

While you could use `DUMP` to see this data to verify the transformation was done correctly, it will print a lot of rows! Try `ILLUSTRATE` or use `LIMIT` to create a new relation:

```
grunt> ILLUSTRATE TEMPERATURE2;
grunt> TEMPERATURE3 = LIMIT TEMPERATURE2 10;
grunt> DUMP TEMPERATURE3;
```

5. Store the new relation in HDFS

Use the `STORE` command to write the `TEMPERATURE2` relation we transformed in Step 4 as HDFS output for later use. Remember to replace `user01` with your own!

```
grunt> STORE TEMPERATURE2 INTO
'/user/user01/DA450_LabFiles/temperature' USING
PigStorage(',');
```

Once this step is complete, look at the output directory using the `fs` command. Notice that the files look like the output of a MapReduce job.

```
grunt> fs -ls /user/user01/DA450_LabFiles/temperature
```

Note: You can also use hadoop `fs` from the bash command line. Use `quit` to exit the grunt shell, and try this command:

```
$ hadoop fs -ls
/user/user01/DA450_LabFiles/temperature
```

Remember to type `pig` to get back into the grunt shell.

Use `cat` on the part files to see your output:

```
grunt> cat /user/user01/DA450_LabFiles/temperature/part-m-
00000
```

6. Store the relation as a CSV file

Right now, our temperature data is in 12 different parts, one for each month, saved as output files in the Hadoop Distributed File System (HDFS). We'd like to be able to save this as a single file, preferably something like a CSV file, which we can use later in other programs such as Apache Drill or spreadsheet software. From the grunt shell:

```
grunt > fs -getmerge
/user/user01/DA450_LabFiles/temperature* ./temperature.csv
```

Try this! The `raw_temperature.csv` file we used to create the `TEMPERATURE` and `TEMPERATURE2` relations was slightly processed version of the original data from the British Antarctic Survey. Also included in your `DA450_LabFiles` folder are eight text files that have not been processed. Using what you learned in this lesson, try to write a Pig Latin script to load this data, transform it into a similar relation to `TEMPERATURE2`, then save it as `stationame_windspeed.csv`. You should have eight files when you are done.

The wind speed files contain only numerical data. The first column is the year, and the next twelve columns correspond to each month, which each contain the average wind speed in knots at that station for that month. The files are space-delimited, not comma-delimited. You will need to transform the data so that there is a `station`, `year`, `month`, and `knots` column, analogous to the `station`, `year`, `month`, and `celsius` columns in the temperature dataset.

Try to write this ETL on your own. You may need to use some user defined functions (UDFs) such as `STRSPLIT()`. Refer to the Apache Pig documentation for more information about these functions: http://pig.apache.org/docs/r0.12.0/func.html.

If you need help, refer to the answer key on page 12 for one method to accomplish this ETL, using the Clean Air station data as an example.

Lesson 3: Data Manipulation

Lab Overview

The purpose of these labs is to gain experience with data manipulation in Pig. For each exercise, we will load data into a Pig relation, and build a data flow using that data to illustrate the concepts of data manipulation. This lab will use real weather data from a select number of research stations throughout the continent of Antarctica. The original files can be downloaded from the British Antarctic Survey at https://legacy.bas.ac.uk/met/READER/data.html.

- Use `FILTER`
- Use user-defined functions

Lab 3.1 Procedure

1. **Start the Grunt shell**

   Log in to your assigned cluster and start the Grunt shell with the procedure from Lab 1.

2. **Load the temperature data**

   Relations in Pig are not persistent. If you exited the grunt shell, the relations you used in the previous lab will be gone. For example, if you try `DUMP TEMPERATURE;` now, you will get an error.

   Let's reload the transformed temperature data we saved as `TEMPERATURE`:

   ```
   TEMPERATURE = LOAD '/user/user01/temperature.csv' USING
   PigStorage(',') as (station:chararray, year:int,
   month:chararray, celsius:float);
   ```

3. **Transform the temperature data**

   Not everyone may be familiar with the Celsius system of measurement. Let's make a new relation, `TEMPERATURE2`, which contains a new column `fahrenheit`. We'll use the `ROUND()` UDF to truncate our results.

   ```
   TEMPERATURE2 = FOREACH TEMPERATURE GENERATE station, year,
   month, celsius, ROUND(celsius*1.8+32) AS fahrenheit:float;
   ```

4. **Filter the temperature data**

   Let's say we're only interested in the data from the South Pole. The name of the weather station at the South Pole is `Clean_Air`:

   ```
   SPTEMP = FILTER TEMPERATURE2 BY station == 'Clean_Air';
   ```


Let's save this for later use. Remember to change the username in the file path!

```
STORE SPTEMP INTO '/user/user01/DA450_LabFiles/southpole'
USING PigStorage(',');
fs -ls /user/user01/DA450_LabFiles/southpole
fs -getmerge /user/user01/DA450_LabFiles/southpole*
./southpole.csv
```

How would you further filter the data so that it only contains months where the average temperature was below -60 degrees Celsius? Create a SOUTHPOLE2 relation on your own. Then try ordering the data from coldest to warmest, or from newest to oldest. Use DUMP to verify your results. If you need help, refer to the answer key on page 12.

Lab 3.2 Procedure

5. Transforming the wind speed data

If you already did the *Try This!* from Lesson 2 on page 7, you should have a file called `clean_air_windspeed.csv` in your `user01` or equivalent directory. If not, copy and paste the ETL from the answer key on page 12 to generate the CLEANAIR4 relation, or load the `clean_air_windspeed.csv` file into a new relation called CLEANAIR4.

Like degrees Celsius, not everyone may be familiar with knots as a unit of measurement. Let's convert knots into kilometers per hour (kph) and miles per hour (mph):

```
SPWIND = FOREACH CLEANAIR4 GENERATE station, year, month,
knots, ROUND(knots*1.852) AS kph:float, ROUND(knots*1.15)
AS mph:float;
```

6. Join two tables

We have already seen the UNION command to append one dataset after another, but there are several other ways to join and combine data in Pig as well. The JOIN command works similar to JOIN in SQL or Hive. Let's make a single relation, SPWEATHER:

```
SPWEATHER = JOIN SPWIND by (station, year, month),
SPTEMP by (station, year, month);
```

If you examine this relation with ILLUSTRATE or DESCRIBE, you will see the station, year, and month columns twice. Let's eliminate them:

```
SPWEATHER2 = FOREACH SPWEATHER GENERATE $0, $1, $2, $3, $7;
```

Note: Depending on the nature of your data, you might not always want to use JOIN or UNION. Other Pig commands include GROUP and COGROUP Refer to the Apache Pig documentation at http://doc.mapr.com/display/MapR/Pig for more information about these and other functions in Pig.

Try this! Now that you are familiar with the basic commands of Pig, let's try some different data! If you have completed the Hive course or are already familiar with SQL, visit https://www.mapr.com/developercentral/code/using-hive-and-pig-baseball-statistics to download baseball statistics and practice using Hive and Pig together.

Appendix: Answer Key

Lesson 2, Try This! (transforming the wind speed data)

```
CLEANAIR = LOAD
   '/user/user01/DA450_LabFiles/Clean_Air.All.wind_speed.txt'
   as (winddata:chararray);
CLEANAIR2 = FOREACH CLEANAIR GENERATE STRSPLIT(winddata,
   '\\s+') as winddata2;
CLEANAIR3 = FOREACH CLEANAIR2 GENERATE FLATTEN(winddata2) as
   (year:int, jan:float, feb:float, mar:float, apr: float,
   may: float, jun:float, jul:float, aug:float, sep:float,
   oct:float, nov:float, dec:float);

JAN = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'jan' AS month, jan AS knots;
FEB = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'feb' AS month, feb AS knots;
MAR = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'mar' AS month, mar AS knots;
APR = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'apr' AS month, apr AS knots;
MAY = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'may' AS month, may AS knots;
JUN = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'jun' AS month, jun AS knots;
JUL = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'jul' AS month, jul AS knots;
AUG = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'aug' AS month, aug AS knots;
SEP = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'sep' AS month, sep AS knots;
OCT = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'oct' AS month, oct AS knots;
NOV = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'nov' AS month, nov AS knots;
DEC = FOREACH CLEANAIR3 GENERATE 'Clean_Air' AS station, year
   AS year, 'dec' AS month, dec AS knots;

CLEANAIR4 = UNION JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP,
   OCT, NOV, DEC;
```


```
STORE CLEANAIR4 INTO '/user/user01/DA450_LabFiles/cleanair'
   USING PigStorage(',');
fs -getmerge /user/user01/DA450_LabFiles/cleanair* ./
   cleanair_windspeed.csv
```

Lesson 3, Step 4: Filter the temperature data

```
   SOUTHPOLE2 = FILTER SOUTHPOLE BY celsius < -60;
   SOUTHPOLE3 = ORDER SOUTHPOLE2 BY year;
   SOUTHPOLE4 = ORDER SOUTHPOLE2 BY celsius;
```