Lab 5: Data Analysis with Pig

Objective:

After completing this lab, you will be familiar with the Pig execution environment to perform data analysis using the various constructs supported by PigLatin. You will also be comfortable with passing arguments to a Pig Script based on Parameter substitution and writing user defined functions using Java.

Requirements:

1. HDP 2.3 Sandbox or a HDP Cluster
2. A terminal client or a way to SSH into a linux box
3. Eclipse

Due Date: Thursday 10/15 11:55 PM

Things to Include in your Submission:

Your submission should be a zip file named username_lab5.zip that comprises of 4 folders and one .pdf file:

1. Task1-This folder will contain all the .pig files needed to accomplish Task 1.The folder should also contain a text file that includes the command for executing your code on a test cluster.
2. Task2 -This folder will contain all the .pig files and all the .java files needed to accomplish Task 2. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster
3. Task3 -This folder will contain all the .pig files and all the .java files needed to accomplish Task 3. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster
4. Task4 -This folder will contain all the .pig files and all the .java files needed to accomplish Task 4. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster.
5. A .pdf file that contains answers to the various questions you were asked during the lab.

Please upload the username_lab5.zip file to the Lab 5 Drop Box on Moodle

Rubric:

1. **Task 1 (15 Points)**
    a. Working Pig Script with Correct logic to compute max, min & avg temperatures - 9 Points
    b. Input and Output Directory are passed as arguments via parameter substitution -3 Points
    c. Following submission instructions - 3 Points
2. **Task 2 (25 Points)**
    a. Working Pig Script with Correct logic to compute the Word Count - 15 Points
    b. Input and Output Directory are passed as arguments via parameter substitution -2 Points
    c. Output is comma separated - 2 Points
    d. Working User Defined function in Java that converts the results to Upper case - 5 Points
    e. Following submission instructions - 1 Points
3. **Task 3 (10 Points)**
    a. Working Pig Script with correct logic to compute the min, max and average temperatures. Working User Defined function in Java that handles the filtration. -7 Points
    b. Input and Output Directory are passed as arguments via parameter substitution -2 Points
    c. Following submission instructions - 1 Point
4. **Task 4 (50 Points)**
    a. Working Pig Script with correct logic to compute for each blog, the hit and error ratio of the caching service - 30 Points.
    b. Output include blog name, error ratio, hit ratio, year job was run, month job was run, day job was run and hour job was run as tab separated values - 5 Points.
    c. Input and Output Directory are passed as arguments via parameter substitution - 2 Points
    d. Date calculation to identify sub-directory is implemented with correct logic and the output is placed in a subdirectory with the following format(year-month-day) under the user provided output directory - 5 Points
    e. Ratio is computed correctly using a user defined function -7 Points
    f. Following submission instructions - 1 Point
5. **Question 1 - 10 Points** (5 points for explanation, 5 points for example)
6. **Question 2 - 15 Points** (5 points per command with 2.5 points for explanation and 2.5 points for example)

Feedback:

Please complete the anonymous feedback for the lab under Feedback → Lab Anonymous Feedback → Lab 5 Anonymous Feedback.

Pig

Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

At the present time, Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject). Pig's language layer currently consists of a textual language called Pig Latin, which has the following key properties:

- **Ease of programming.** It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
- **Optimization opportunities.** The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
- **Extensibility.** Users can create their own functions to do special-purpose processing.

Pig is made up of two pieces:

- The language used to express data flows, called Pig Latin.
- The execution environment to run Pig Latin programs. There are currently two environments: local execution in a single JVM and distributed execution on a Hadoop cluster.

A Pig Latin program is made up of a series of operations, or transformations, that are applied to the input data to produce output. Taken as a whole, the operations describe a data flow, which the Pig execution environment translates into an executable representation and then runs. Under the covers, Pig turns the transformations into a series of MapReduce jobs, but as a programmer you are mostly unaware of this, which allows you to focus on the data rather than the nature of the execution.

Why Pig?

Let us consider a simple use case. You have data logs from an e-commerce website that provides with you access to information about usernames and the various items they clicked on and the amount of time they spent on each shopping item. You have access to information about user profiles(age, interests, prior shopping history). If you are interested in constructing a personalized recommendation engine, information from these two and potentially other data sources need to be constructed. The presence of a large amount of data necessitates the use of the Hadoop ecosystem. One potential technique to deploy here would to be write a custom map-reduce program that uses the mappers to process information from these sources and uses the reduce phase to join the data (a reduce side join). To do this correctly, might require that the developer also implement a custom partitioner and a comparator. This is not a trivial application and could take valuable time.

Pig is specifically built for use cases like this and enables the developer to program this scenario at a high level. For instance, it could be something as simple as Load Dataset1, Load Dataset2, Join both Datasets and store the result in HDFS. As mentioned above, Pig converts these commands into optimized MapReduce jobs. Pig is very supportive of a programmer writing a query, since it provides several commands for introspecting the data structures in your program as it is written. Even more useful, it can perform a sample run on a representative subset of your input data, so you can see whether there are errors in the processing before unleashing it on the full dataset.

Pig isn't useful for all data processing tasks. Much like a MapReduce Job, Pig is built for batch processing data and is extremely useful when you have you to scan the entire dataset or at the very least a substantial portion of the data. It is not built to just read a few lines from a large dataset.

Execution Modes

Pig has two execution types or modes: local mode and MapReduce mode. In local mode, Pig runs in a single JVM and accesses the local filesystem. This mode is suitable only for small datasets and when trying out Pig.

The execution type is set using the -x or -exectype option. To run in local mode, set the option to local. This will start the Pig interactive shell, Grunt. It is very similar to the interactive ruby shell (IRB). It allows for the execution of Pig commands and operators from the shell, provides immediate response and is good for experimentation. It supports command history, line editing capabilities and launch previous defined pig scripts.

Pig - Local Mode

```
$ pig -x local

grunt >
```

In MapReduce mode, Pig translates queries into MapReduce jobs and runs them on a Hadoop cluster. . MapReduce mode is what you use when you want to run Pig on large datasets. To use MapReduce mode, you first need to check that the version of Pig is compatible with the version of Hadoop you are using. Pig releases will only work against particular versions of Hadoop. Pig honors the HADOOP_HOME environment variable for finding which Hadoop client to run. However, if it is not set, Pig will use a bundled copy of the Hadoop libraries. Note that these may not match the version of Hadoop running on your cluster, so it is best to explicitly set HADOOP_HOME.

Next, you need to point Pig at the cluster's namenode and jobtracker. If the installation of Hadoop at HADOOP_HOME is already configured for this, then there is nothing more to do. Otherwise, you can setHADOOP_CONF_DIR to a directory containing the Hadoop site file (or files) that define *fs.default.name* and *mapred.job.tracker.*

Alternatively, you can set these two properties in the *pig.properties* file in Pig's *conf* directory (or the directory specified by PIG_CONF_DIR). Once the cluster information has been configured, we can start Pig in mapreduce mode by typing in just Pig.

Pig - MapReduce Mode

```
$ pig

grunt >
```

Commands & Functions

PigLatin provides support for several statements and functions that are useful while processing data in Pig. You can find a listing of functions and statements here.

Task 1: Temperature Data Set (15 Points)

Create a program in Pig to find the minimum temperature, maximum temperature and the average temperature of each given year in the data set. Please use the sample dataset titled tempInput.txt as your input data set. The pig script should filter the data to ensure that only data records whose quality attribute(3rd column) is either zero or one are used by the script.The pig

program should use parameter substitution to accept the input location and the output directory for Pig. Please refer to the book on parameter substitution.

Turn in:

1. Create a folder called Task1
2. This folder will contain all the .pig files needed to accomplish Task 1.The folder should also contain a text file that includes the command for executing your code on a test cluster.

Rubric:

1) Working Pig Script with Correct logic to compute max, min & avg temperatures - 9 Points
2) Input and Output Directory are passed as arguments via parameter substitution -3 Points
3) Following submission instructions - 3 Points

Task 2: Word Count (25 Points)

Create a program in Pig to count the number of times each word appears in an input file.Please use the sample dataset titled testFile.txt as your input data set. The pig program should use parameter substitution to accept the input location and the output directory for Pig. Please refer to the Tokenize and Flatten commands. They might assist in the program.

- The final output should contain each word and for each word, the number of times it appears in the sample text. The above columns should be comma separated.
- The pig program should also include a basic user defined function called Upper (or more accurately edu.rosehulman.yourusername.Upper, for instance edu.rosehulman.mohan.upper) that converts each word to uppercase. Thus the output should contain each word in uppercase and for each word, the number of times it appears in the sample text.

Turn in:

1. Create a folder called Task2
2. This folder will contain all the .pig files and all the .java files needed to accomplish Task 2. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster.

Rubric:

1) Working Pig Script with Correct logic to compute the Word Count - 15 Points
2) Input and Output Directory are passed as arguments via parameter substitution -2 Points
3) Output is comma separated - 2 Points
4) Working User Defined function in Java that converts the results to Upper case - 5 Points
5) Following submission instructions - 1 Points

Task 3: Revisiting the temperature Data Set using UDF (10 Points)

We will be essentially working on the same code as Task 1. Create a program in Pig to find the minimum temperature, maximum temperature and the average temperature of each given year in the data set. Please use the sample dataset titled tempInput.txt as your input data set. The pig script should filter the data to ensure that only data records whose quality attribute(3rd column) is either zero or one are used by the script. The filtering that has been described above has to be handled by a user defined function. The pig program should use parameter substitution to accept the input location and the output directory for Pig.

Turn in:

1. Create a folder called Task3
2. This folder will contain all the .pig files and all the .java files needed to accomplish Task 3. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster.

Rubric:

1) Working Pig Script with Correct logic to compute the min, max and average temperatures. Working User Defined function in Java that handles the filtration. 7 Points
2) Input and Output Directory are passed as arguments via parameter substitution -2 Points
3) Following submission instructions - 1 Point

Task 4: Log Analysis (50 Points)

Pig is commonly used for log analysis. We have uploaded a sample log file in gzipped format from a website (sample.gz). Please note that Pig can directly read a gzipped file without a need for user provided decompression. This website uses a front end caching utility called Amazon

Cloudfront. Essentially it caches page requests so that the the website is only serving content when Cloudfront needs to renew its cache. It produces close to 1000 of these small zip logs per day. **The format for the logs is here.**

We are interested in a simple piece of functionality for this demonstration. We are interested in performing some simple statistical analysis on the edge result (Hit, Miss, Error, etc) This could yield results on which parts of the site are getting the most benefit out of the caching service. A cursory look at the log file will reveal that several URI's (the cs-uri-stem attribute) starts with something like /blog/[name]/...stuff... (ex /blogs/theanchoress/stuff) - there are around 400 different blogs that could fill the [name] in the URI. For instance if there is a blog called hadoopAnalysis, then you might have log entries that contain the following value for the cs-uri-stem attribute (/blogs/hadoopAnalysis/stuff)

We are going to generate statistics in terms of blog names. Specifically in this use case, it would be good to know which blogs are serving from the cache often and which are not serving from the cache very often if at all. If a cached version is given back to the client then Cloudfront reports a "Hit", anything else means there was an error or Cloudfront had to renew the cache by reaching to our origin servers. Data about a cache “hit” or “error” is available in the **x-edge-result-type attribute**.

In general we plan to express this by taking the number of Hits for all the URI's pertaining to a certain blog, and dividing it by the total number of URI's reported for the same particular blog. (Hits / Total requests). As an additional data point, we will further determine for each blog the ratio of the number of times Cloudfront reported an error/ total requests.

Output:

- The output of this program should look as follows. For instance if there is a blog called hadoopAnalysis, then you might see the following in the output

  hadoopAnalysis 0.7 0.3 2014 10 5 11

  The columns are blog name (hadoopAnalysis), hitRatio(0.7), errorRatio(0.3), year when the script was run(2014), month when the script was run(10), day when the script was run(5), hour when the script was run(11). You will find the DateTime functions useful in identifying the the last 4 attributes of the result.

- The pig program should use parameter substitution to accept the input location and the output directory for Pig. Further, the Pig program should write the output in a subdirectory formatted with the date that the script was run(for instance 2014-10-05) and place the results within the subdirectory. For instance if the output directory is specified

as /tmp/pigOutput and the script was run on the 5th of October, 2014, the output files should be under /tmp/pigOutput/2014-10-05.

You will find the examples on the use of declare command useful above to identify the date in question and the date command in linux (type in man date to see how to use it)

- The computation of the ratio will be handled by a user defined function. You will have a function that accepts two arguments,for instance the number of hits or errors and the total number of requests and returns the desired ratio.

A sample output is available in the Lab 5 directory on the class website.

Turn in:

1. Create a folder called Task4
2. This folder will contain all the .pig files and all the .java files needed to accomplish Task 4. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster.

Rubric:

1) Working Pig Script with Correct logic to compute for each blog, the hit and error ratio of the caching service - 30 Points.
2) Output include blog name, error ratio, hit ratio, year job was run, month job was run, day job was run and hour job was run as tab separated values - 5 Points.
3) Input and Output Directory are passed as arguments via parameter substitution - 2 Points
4) Date calculation to identify sub-directory is implemented with correct logic and the output is placed in a subdirectory with the following format(year-month-day) under the user provided output directory - 5 Points
5) Ratio is computed correctly using a user defined function -7 Points
6) Following submission instructions - 1 Point

Question 1 (10 points): Pig introduced the notion of a macro starting with Pig 0.9.1. What is a macro, explain it with an example. (Answer 5 points, Example 5 points)

Question 2 (15 points) Explain the following commands/functions with an example. Each command carries 5 points (2 points for the explanation, 3 points for the example)

a. COGROUP
b. RANK
c. STREAM

Due Date: Thursday 10/15 11:55 PM

Things to Include in your Submission:

Your submission should be a zip file named username_lab5.zip that comprises of 4 folders and one .pdf file:

1. Task1-This folder will contain all the .pig files needed to accomplish Task 1.The folder should also contain a text file that includes the command for executing your code on a test cluster.
2. Task2 -This folder will contain all the .pig files and all the .java files needed to accomplish Task 2. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster
3. Task3 -This folder will contain all the .pig files and all the .java files needed to accomplish Task 3. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster
4. Task4 -This folder will contain all the .pig files and all the .java files needed to accomplish Task 4. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster.
5. A .pdf file that contains answers to the various questions you were asked during the lab.

Please upload the username_lab5.zip file to the Lab 5 Drop Box on Moodle

Rubric:

1. **Task 1 (15 Points)**
    a. Working Pig Script with Correct logic to compute max, min & avg temperatures - 9 Points
    b. Input and Output Directory are passed as arguments via parameter substitution -3 Points
    c. Following submission instructions - 3 Points
2. **Task 2 (25 Points)**
    d. Working Pig Script with Correct logic to compute the Word Count - 15 Points
    e. Input and Output Directory are passed as arguments via parameter substitution -2 Points
    f. Output is comma separated - 2 Points
    g. Working User Defined function in Java that converts the results to Upper case - 5 Points

h. Following submission instructions - 1 Points

3. **Task 3 (10 Points)**
   i. Working Pig Script with correct logic to compute the min, max and average temperatures. Working User Defined function in Java that handles the filtration. -7 Points
   j. Input and Output Directory are passed as arguments via parameter substitution -2 Points
   k. Following submission instructions - 1 Point
4. **Task 4 (50 Points)**
   l. Working Pig Script with Correct logic to compute for each blog, the hit and error ratio of the caching service - 30 Points.
   m. Output include blog name, error ratio, hit ratio, year job was run, month job was run, day job was run and hour job was run as tab separated values - 5 Points.
   n. Input and Output Directory are passed as arguments via parameter substitution - 2 Points
   o. Date calculation to identify sub-directory is implemented with correct logic and the output is placed in a subdirectory with the following format(year-month-day) under the user provided output directory - 5 Points
   p. Ratio is computed correctly using a user defined function -7 Points
   q. Following submission instructions - 1 Point
5. **Question 1 - 10 Points** (5 points for explanation, 5 points for example)
6. **Question 2 - 15 Points** (5 points per command with 2.5 points for explanation and 2.5 points for example)

Feedback:

Please complete the anonymous feedback for the lab under Feedback → Lab Anonymous Feedback → Lab 5 Anonymous Feedback.