Homework 8: Hadoop and Pig

Due date: March 13, 2014 at 11:59pm, in the dropbox AT MOST 1 LATE DAYS!

TURN IN INSTRUCTIONS: Turn in eleven files to the Catalyst dropbox. What to turn is marked in red

Problem 0: Setup your Pig Cluster

1. **AMAZON CODES:** You should have received your Amazon code by email. Please email the TA or instructor if you did not get the code.

2. Follow these instructions to setup the cluster and . NOTE: It will take you a good **60 minutes** to go through all these instructions without even trying to run example.pig at the end. But they are worth it. You are learning how to use the Amazon cloud, which is by far the most popular cloud today! At the end, the instructions will refer to *example.pig*. This is the name of the sample program that we will run in the next step.

3. Download the project archive, hw8.tar.gz.
4. You will find example.pig in hw8.tar.gz.

   example.pig is a Pig Latin script that loads and parses the billion triple dataset that we will use in this assignment into triples: (subject, predicate, object). Then it groups the triples by their object attribute and sorts them in descending order based on the count of tuple in each group.

5. Follow the README.txt: it provides more information on how to run the sample program called example.pig.

6. There is nothing to turn in for Problem 0

Useful Links

Pig Latin wiki page

Homework Description

As we discussed in class, we live in a "big data" era: our society is generating data at an unprecedented scale and rate. In fact, we are generating so much data that we are unable to take advantage of most of that data. This is quite unfortunate.

A large fraction of this data takes the form of gigantic graphs: A social network is a graph where vertices represent people and edges represent friendships. The Web is a graph where vertices represent pages and edges represent hyperlinks between pages. These graphs are very large and are difficult to study. One of the key challenges is that many graph algorithms are difficult to parallelize.

In this assignment, we will perform some basic analysis over one such graph. This graph is representative of other important graphs. The graph that we will study comes from the billion triple dataset. This is an RDF dataset that contains a billion (add or take a few) triples from the Semantic Web. Some Webpages on the Web have a machine-readable description of their semantics stored as RDF triples: our dataset was obtained by a crawler that extracted all RDF triples from the Web.

RDF data is represented in triples of the form:

```
subject  predicate  object  [context]
```

The [context] is not part of the triple, but is sometimes added to tell where the data is coming from. For example, file `btc-2010-chunk-200` contains the two "triples" (they are actually "quads" because they have the context too):

```
<http://www.last.fm/user/ForgottenSound> <http://xmlns.com/foaf/0.1/nick> "ForgottenSound"
<http://rdf.opiumfield.com/lastfm/friends/life-exe> .
<http://dblp.l3s.de/d2r/resource/publications/journals/cg/WestermannH96>
<http://xmlns.com/foaf/0.1/maker> <http://dblp.l3s.de/d2r/resource/authors/Birgit_Westermann>
<http://dblp.l3s.de/d2r/data/publications/journals/cg/WestermannH96> .
```

The first says that Webpage <http://www.last.fm/user/ForgottenSound> has the nickname "ForgottenSound"; the second describes the maker of another webpage. `foaf` stands for *Friend of a Friend*. Confused ? You don't need to know what they mean; some of the many triples refer to music, http://dbtune.org, others refer to company relationships, etc. For our purpose, these triples are just a large collection of triples. There were 317 2GB files in the billion triple dataset when we downloaded it. We uploaded them to Amazon's Web Services in S3: there were some errors, and only 251 uploaded correctly, for a total of about 550 GB of data.

This graph is similar in size to the web graph. As part of this assignment, we will compute the out-degree of each node in the graph. The out-degree of a node is the number of edges coming out of the node. This is an important property. If a graph is random, the out-degree of nodes will follow an exponential distribution (i.e., the number of nodes with degree d should be exp(- c*d) for some constant c). We will write the script in Problem 2, where we will run it on a small data sample. We will run the script on the big graph in Problem 4. What is very interesting is that we will find the distribution of node out-degrees to follow a power law (1/d^k for some constant k and it will look roughly like a straight-line on a graph with logarithmic scales on both the x and y axes) instead of an exponential distribution. If you look at Figures 2 and 3 in this paper, you will find that the degrees of web pages on the web, in general, follow a similar power law distribution. This is very interesting because it means that the Web and the semantic Web cannot be modeled as random graphs. They need a different theoretical model.

In Problem 3, we will look for paths of length 2 in a sub-graph of our big graph. This is a simple version of more complex algorithms that try to measure the diameter of a graph or try to extract other related properties. We will do all this on a very real 0.5TB graph! How cool will that look on your resume: "Analyzed properties of a 0.5TB (a billion vertices) graph using Pig/Hadoop".

You will access the following datasets in S3, throught pig (using the LOAD command -- see example.pig)

s3n://uw-cse344-test/cse344-test-file -- 250KB. This is used in example.pig. Always use this file for debugging your scripts first!

s3n://uw-cse344/btc-2010-chunk-000 -- 2GB. You will use this dataset in questions 1, 2, 3..

s3n://uw-cse344 -- 0.5TB. This directory contains 251 files btc-2010-chunk-000 to btc-2010-chunk-317 (since only 251 of the original 318 files uploaded correctly). You will use this in problem 4.

It is not necessary for the assignment, but if you want to inspect the files directly, you can access them over the Internet using urls of the following form (Note that accessing the 0.5TB file in this way is not recommended!):

`http://uw-cse344.s3.amazonaws.com/btc-2010-chunk-000`

`http://uw-cse344-test.s3.amazonaws.com/cse344-test-file`

Problem 1: Getting started with Pig on chunk-000

Note: You will need to copy the output of your Pig scripts from the Hadoop filesystem for all problems. You can find instructions to do this here.

Modify `example.pig` to use the file `uw-cse344/btc-2010-chunk-000` instead of `uw-cse344-test/cse344-test-file`. Run on an AWS cluster with **10 nodes**, and answer the following questions (also see hints below).

1.1 How many MapReduce jobs are generated by example.pig?

1.2 How many reduce tasks are within the first MapReduce job? How many reduce tasks are within later MapReduce jobs?

1.3 How long does each job take? How long does the entire script take?

1.4 What is the schema of the tuples after each of the following commands in example.pig?

- After the command `ntriples = ...`
- After the command `objects = ...`
- After the command `count_by_object = ...`

Hint 1: Use the job tracker to see the number of map and reduce tasks for your MapReduce jobs.

Hint 2: To see the schema for intermediate results, you can use Pig's interactive command line client `grunt`, which you can launch by running Pig without specifying an input script on the command line. When using `grunt`, a command that you may want to know about is `describe` . To see a list of other commands, type `help`.

What you need to turn in:
Run your program on **btc-2010-chunk-000**, and submit your answers to problems 1.1 - 1.4 in a file named `problem1-answers.txt`.

Problem 2A: Compute a Histogram on cse344-test-file

Using the 'cse344-test-file' file, write a Pig script that groups tuples by the subject column, and creates/stores histogram data showing the distribution of counts per subject, then generate a scatter-plot of this histogram. The histogram consists of:

- The x-axis is the counts associated with the subjects, and
- The y-axis is the total number of subjects associated with each particular count.

So, for each point (x,y) that we generate, we mean to say that y subjects each had x tuples associated with them after we group by subject. Run your script on an AWS cluster and record the mapreduce jobs information (cluster size, # MapReduce jobs, runtimes, # reduce tasks per job). Copy the results to your local machine. Generate a log-log scatter-plot graph, using either `excel` or `gnuplot` to plot the histogram points. Save, and turn in, the plot in some image format, e.g. jpeg or png.

A few comments to help you get started:

- We expect that your script will (1) group the input data by subject and count the tuples associated with each subject then (2) group the results by these intermediate counts (x-axis values) and compute the final counts (y-axis values).
- To get more familiar with the Pig Latin commands, we suggest that you also take a look at the Pig Latin Wiki Page.
- Copying files. After you run your job on the cluster, you will need to copyToLocal (i.e. move the files to your local directory) and "cat" together all files named `part-*`. Once the results have been cat-ed together, copy them back to your local machine. Alternatively, you can use the "hadoop dfs -getmerge" command. See instructions.
- Generating the plot. If you use excel, then: (a) import the tab-separated text file in excel, (b) generate a scatter-plot, (c) click on each axis and make it logarithmic. If you use `gnuplot`, we have prepared a script which makes it easier for you to run `gnuplot`. Use the files plot.sh and plot.gnu as follows:

```
chmod +x plot.sh
./plot.sh PIG_RESULTS_FILE
```

The script generates a PNG image of the plot in your current directory. Your PIG_RESULTS_FILE needs to be tab-separated and have two columns, x and y. The data also needs to be (numerically) sorted by x. You can sort either using Pig or simply run Unix' `sort -n input > output` after your job has completed (by default sorting in Pig is alphabetical).

DEBUGGING:

- Since you are using the small test file in this question, you can run a small, **1-node cluster**.
- In this question, we are debugging the script. The output of this question is thus not going to be terribly interesting. In fact, your scatterplot should only have two points: (1,1) and (3,333).
- To debug a Pig Latin script, try to run Pig as follows:

  ```
  pig -x local
  ```

  Run all commands as you normally would, except for store. You need to store your results locally:

  ```
  store my_final_output into '/tmp/finaloutput' using PigStorage()
  ```

- Once you are done debugging in local mode, try to run your script by issuing real MapReduce jobs. That is run with "pig" instead of "pig -x local" (remember to change the store command).

What you need to turn in: Nothing. This was a debug step.

Problem 2B: Compute a Histogram on chunk-000

Now run your script from Problem 2A on 'btc-2010-chunk-000' file. Please use a **5-node cluster**.

Note: this script took about 21 minutes with 5 nodes.

What you need to turn in:
Run your program on **btc-2010-chunk-000**, and submit four files: (a) your Pig program in `problem2.pig`. (b) your scatter-plot in `problem2.png`, or `problem2.jpeg` (or some other picture format), (c) your computed result file (`problem2-results.txt`), (d) your MapReduce jobs information (`problem2-answers.txt`).

Problem 3: Compute a Join on chunk-000

In this problem we will consider the subgraph consisting of triples whose subject matches rdfabout.com: for that, filter on `subject matches '.*rdfabout\\.com.*'`. Find all chains of lengths 2 in this subgraph. More precisely, return all sextuples (subject, predicate, object, subject2, predicate2, object2) where object=subject2.

Note: Newer versions of Pig will automatically drop the duplicate column in the join output. In that case, you do NOT need to add the column back.

Suggestions on how to proceed:

- First filter the data so you only have tuples whose subject matches 'rdfabout.com'.
- Make another copy of the filtered collection (it's best to re-label the subject, predicate, and objects, for example to subject2, predicate2, object2).
- Now join the two copies:
  - the first copy of the 'rdfabout.com' collection should match on object.
  - the second copy of the 'rdfabout.com' collection should match on subject2.
- Remove duplicate tuples from the result of the join
- Order the results by the predicate from the first copy

As above, first debug your script for this problem using the test file. Once your script is debugged, then run it on the bigger file 'btc-2010-chunk-000'. While debugging on the test file, make the following two changes:

1) Use the following filter

```
subject matches '.*business.*'
```

2) Change the join predicate to be subject=subject2

Otherwise, you will not get any results.

Once you are done debugging, change the filter and the join condition and run this script on an AWS cluster with as many nodes as you like and on the bigger file 'btc-2010-chunk-000'. Add a comment to your pig script describing the number of nodes you used and how long it took your script to run.

Note: this script took about 18 minutes with 10 nodes.

What you need to turn in:
Run your program on **btc-2010-chunk-000**, and submit two files: (a) your Pig program in `problem3.pig`, and (b) your computed result file (`problem3-results.txt`).

Problem 4: Compute a Histogram on the Entire Dataset

Compute the histogram in Problem 2 on the **entire 0.5TB dataset**. Use as many nodes as you like up to 19 nodes. We would need to ask explicit permission form Amazon to run bigger clusters.

You need to modify the load instruction to:

```
raw = LOAD 's3n://uw-cse344' USING TextLoader as (line:chararray);
```

Note: this query will take more than ***4 hours to run***. Plan accordingly, and monitor carefully: if anything looks wrong, abort, fix, restart.

When you are done, appreciate how *relatively* quick and easy it was to analyze a 0.5TB graph!

What you need to turn in:
Run your program on the entire dataset **uw-cse344** and submit four files: (a) your Pig program in `problem4.pig`. (b) your scatter-plot in `problem4.png`, or `problem4.jpeg` (or some other picture format), (c) your computed result file (`problem4-results.txt`), (d) your MapReduce jobs information (`problem4-answers.txt`).