CSE427 – Homework 5

M. Neumann

Due THU 02/25/2016 10am

Getting Started

Update your SVN repository. Find instructions on how to checkout, update, and commit to your SVN repository here: `http://sites.wustl.edu/neumann/resources/cse427s_resources/`

When needed, you will find additional materials for *homework x* in the folder `hwx`. So, for the current assignment the folder is `hw5`.

++++ Read the submission instructions carefully! ++++

Indicating Group Work

Use the file `partners.txt` to indicate group work. Follow these instructions exactly, to ensure to get credit!

- `partners.txt` needs to include up to 2 wustlkeys in the first two lines (one line per wustlkey)
- **first line/wustlkey is the repository, where the solution is located**. We will **only** consider the submission in this repository!
- Every student in a group needs to have **the same** `partners.txt` in the hwx folder in their repository (indicating that the partnership are **mutually accepted**)!
- If you do not have a partner, try to find one. If you want to submit on your own, indicate your wusltkey in the first line of `partners.txt` and leave the second line blank.

Problem 1: Analyzing Web Server Logs (70%)

For this problem you will analyze a log file from a web server to count the number of hits made from each unique IP address.
Your task is to count the number of hits made from each IP address in the sample (anonymized) web server log file that you uploaded to the `weblog` directory in HDFS in Lab 1.

You are free to implement your MapReduce job in Java or using Hadoop streaming. You can use the Mapper and Driver Java stubs provided in:

```
~/workspace/log_file_analysis/src/stubs
```

(a) **Data:** Look at the `weblog` data and describe the record entries. Recall that you created a sample dataset called `testlog` in Lab 1. What fraction of the full `weblog` data is contained in this dataset?

(b) **Design:** Write down the Mapper input and output, as well as, the Reducer input and output for the following test input:

```
10.223.157.186 - - [15/Jul/2009:21:24:17 -0700] "GET /assets/img/media.jpg HTTP/1.1" 200 110997
10.223.157.186 - - [15/Jul/2009:21:24:18 -0700] "GET /assets/img/pdf-icon.gif HTTP/1.1" 200 228
10.216.113.172 - - [16/Jul/2009:02:51:28 -0700] "GET / HTTP/1.1" 200 7616
10.216.113.172 - - [16/Jul/2009:02:51:29 -0700] "GET /assets/js/lowpro.js HTTP/1 .1" 200 10469
10.216.113.172 - - [16/Jul/2009:02:51:29 -0700] "GET /assets/css/reset.css HTTP/1.1" 200 1014
```

What is the Reducer doing? Have you seen this before? (Remember your answer to this question for the next part.)

(c) **Implement Parts:** Implement the necessary parts for a MapReduce program to count the number of hits made from each IP address in the access log file. Your final result should be a file in HDFS containing each IP address, and the count of log hits from that address.

(d) **Test Parts:** Implement three tests in `TestProcessLogs.java` for testing the your implementation, one each for the Mapper, Reducer, and the entire MapReduce flow. (If you implemented parts for a streaming job, provide input examples (as `TestInput.txt`) and a bash script (`TestProcessLogs.sh`) that tests your Mapper, Reducer, and the entire MapReduce flow.)

(e) **Testing Locally:**

(i) What is the difference between testing a MapReduce program locally and running it on the cluster considering input and output location, print statements (to `stderr` or `stdout`), as well as job management?

(ii) Run your program on the `testlog` data **locally** using the command line. Provide the command in the `hw5.txt` file. If you did not implement `ToolRunner`, adjust the driver and provide the respective lines in the `hw5.txt` file.

(iii) Run your program on the `testlog` data **locally** using Eclipse. Find instructions on how to do this on the course webpage under Resources and HowTos (`http://www.cse.wustl.edu/~m.neumann/sp2016/cse427/protected/EclipseRef_Local.pdf`). Which way do you prefer (give 2 arguments to support your decision)?

(f) **Run in Pseudo-distributed Cluster:** Execute your program on the pseudo cluster using the `testlog` data. How many different IP addresses are in the `weblog` data? Did every line in `testlog` contribute to a count? You do not have to write any code to answer this question; simply look at the result file and the number of lines in `testlog`.

(g) **Run Job:** Now, your MapReduce program is ready for submission to the real Hadoop cluster. What do you have to bear in mind when performing this step? As we do not have a real cluster, submit a job again, now using the full `weblog` data.

(h) **Results:** Check the results in HDFS. How many different IP addresses are in the `weblog` data? How many hits were made from the following IP addresses: 10.1.100.199, 10.1.100.5, 10.99.99.58? Why are the IP addresses (globally) sorted?

Submit your answers to (a,b,e-h) by editing the **hw5.txt** file in your SVN repository. Submit your answer to (c) by adding your Mapper, Reducer, and Driver to the hw5 folder in your SVN respoitory. Submit your answer to (d) by adding TestProcessLogs.java or TestProcessLogs.sh and TestInput.txt to the hw5 folder in your SVN respoitory; do not forget to **svn add the files before committing**. Do not submit any jar files for this problem!

To commit your solution run:

```
$ svn commit -m 'hw5 submission' .
```

Problem 2: Custom WritableComparable (30%)

In this problem you will write a simple program that counts the number of occurrences of each full name (i.e., first and last name) in a list of names.

Use the following **test input** to test your implementation:

```
Smith Joe 1963-08-12 Poughkeepsie, NY
Murphy Alice 2004-06-02 Berlin, MA
Smith Joe 1832-01-20 Sacramento, CA
```

The output for this test input should look like this:

```
(Smith,Joe) 2
(Murphy,Alice) 1
```

(a) Implement a custom WritableComparable type holding two strings.
Use the stub StringPairWritable provided in:

```
~/workspace/writables/src/stubs
```

Note that Eclipse automatically generated the hashCode and equals methods in the stub file. You can generate these two methods in Eclipse by right-clicking in the source code and choosing "Source" -> "Generate hashCode() and equals()".

(b) Implement a MapReduce program counting the number of occurrences of each (full) name (i.e., first and last name) in a list of names provided in ~/training_materials/developer/data/nameyeartestdata on your local filesystem.

Your MapReduce program requires a Reducer that sums the number of occurrences of each key. This is the same function that the SumReducer used previously in wordcount, except that SumReducer expects Text keys, whereas the Reducer for this job will get StringPairWritable keys. You may either adapt the SumReducer to accommodate other types of keys, or you can use the LongSumReducer Hadoop library class, which does exactly the same thing.

(c) Test your code using local job runner or by submitting a Hadoop job to the (pseudo-)cluster as usual. If you submit the job to the cluster, you will need to copy the data to HDFS first.

Submit your answer to (a) by adding **StringPairWritable.java** to the hw5 folder in your SVN repository. Submit your implementation for (b) as **StringPairTest.jar**; make sure **all required**

classes are included! For answer (c) submit the result file.

To add the .jar and results files to your SVN repo before committing run:

```
$ svn add StringPairTest.jar
$ svn add part-r-00000
$ svn commit -m 'hw5 submission' .
```

Bonus Problem (5% up to a max. of 100%) - no group work!

Write a review for this homework and store it in the file `hw5_review.txt` provided in your SVN repository (and commit your changes). This file should only include the review, **no other information** such as name, wustlkey, etc. Remember that you are not graded for the content of your review, solely it's completion.

You can only earn bonus points if you write **at least 50 words**. Bonus points are given to the **owner of the repository only** (no group work!).