ECE536 Computer Software Systems

Assignment #7 (5%)

Spring 2012

Due: Tuesday, 04/17/12 11:30pm

In this assignment, you will practise the well known *word count* problem and its variation on AWS Elastic MapReduce (EMR) platform.

Part 1 (30%): Execute the *word count* problem on AWS Elastic MapReduce (EMR) platform

There are many options for how to write your MapReduce jobs: in Java with HADOOP's native Java API, on in any programming language that is able to read from standard in and write to standard out with HADOOP Streaming. Michael G. Noll provides an example in Python in his posting, "Writing An Hadoop MapReduce Program In Python," at
www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/
You can copy and paste it as long as you fully understand this program, but you may also use any other language. You do not need to optimize the problem by using Python iterators and generators. As indicated in his article, you should test your code by Unix/Linux pipe: (cat data | map | sort | reduce) before executing them on AWS EMR. This test does not reuire installation of HADOOP.

After you verify your program's correctness, execute the *word count* problem on AWS Elastic MapReduce (EMR) platform on the following three data inputs (use utility "wc" to count the number of words):

- Alice's Adventures in Wonderland by Lewis Carroll at http://www.gutenberg.org/ebooks/11, about 4K words.
- Jane Eyre by Charlotte Bronte at http://www.gutenberg.org/ebooks/1260, about 20K words.
- Pick a text content by your choice such that the word count is larger than 500K.

What to submit:
- Capture of your AWS Management Console when running your word count program;
- Description of your third choice of input dataset, including its source and word count;
- List the first AND the last 10 lines of the output for all three datasets.

Part 2 (40%) A variation of the *word count* problem in part I.

Assume we are particularly interested in words that are neither too short (less than 10 characters) nor too long (equal or more than 20 characters). Rewrite your MapReduce programs in Part 1 to accommodate the change of requirements:
- At Reduce stage (not in Map stage!), eliminate those words out of the ROI (range of

interests.)
- Output for every words whose length are less than 20 but equal or greater than 10.
- The output should be sorted by the length of the words, instead of the alphabetical ordering.

As an example, take input of "Albuquerque is the largest city in the state of New Mexico, United States. It is the county seat of Bernalillo County and is situated in the central part of the state, straddling the Rio Grande. The city population was 545,852 as of the 2010 Census and ranks as the 32nd-largest city in the U.S. It has a metropolitan population of 887,077 as of the 2010 Census. Albuquerque is the 57th-largest United States metropolitan area." your output should be:
- 10 Bernalillo 1
- 10 population 2
- 11 straddling 1
- 12 Albuquerque 2
- 12 32nd-largest 1
- 12 57th-largest 1

After you verify your program's correctness, execute the *word count II* problem on AWS Elastic MapReduce (EMR) platform on the same datasets in Part 1. What to submit:
- Capture of your AWS Management Console when running your word count II program;
- Show your new program to solve the *word count II* problem; and
- List the first AND the last 10 lines of the output for all three datasets.

Part 3 (30%) Another variation of the *word count* problem in part II.

Assume we are interested in the most frequently used words that are neither too short (less than 10 characters) nor too long (equal or more than 20 characters). Rewrite your MapReduce programs in Part 2 to accommodate the change of requirements:
- At Map or Reduce stage, eliminate those words out of the ROI (range of interests.)
- Output for the most frequently used word for every different-length words. Chose the first word if there are ties;
- You can design a single-phase MapReduce, or a multiple phase MapReduce (taking the output of the first Reduce as the input of the second Map;
- The output should be sorted by the length of the words, instead of the alphabetical ordering.

As an example, take input of "Albuquerque is the largest city in the state of New Mexico, United States. It is the county seat of Bernalillo County and is situated in the central part of the state, straddling the Rio Grande. The city population was 545,852 as of the 2010 Census and ranks as the 32nd-largest city in the U.S. It has a metropolitan population of 887,077 as of the 2010 Census. Albuquerque is the 57th-largest United States metropolitan area." your output should be:
- 10 population 2
- 11 straddling 1
- 12 Albuquerque 2

After you verify your program's correctness, execute the *word count III* problem on AWS Elastic MapReduce (EMR) platform on the same datasets in Part 1. What to submit:
- Capture of your AWS Management Console when running your word count II program;
- Show your new program to solve the *word count III* problem; and
- List the first AND the last 10 lines of the output for all three datasets.

Assigned by Dr. Shu, shu@ece.unm.edu