CSCI4180 (Fall 2013)

Assignment 1: Counting by MapReduce

Due on Oct 10, 2013, 23:59:59

Introduction

The goals of this assignment are to let students get familiar with Hadoop MapReduce and simple MapReduce development. Parts 1-4 are done on our cloud testbed, while Part 5 is done on Windows Azure.

Part 1: Getting Started on Hadoop (15%)

In this part, you need to demonstrate the following:

1. Configure Hadoop in *fully distributed mode* on all your assigned virtual machines. In particular, you should configure one VM to run as the JobTracker, and all other VMs to run as the TaskTrackers.
2. Run the provided WordCount program on your Hadoop platform.

Please show to the TAs that you achieve the above requirements.

Note: We provide two sample datasets for you to test your program. One is the KJV Bible, and another one is the complete works of Shakesphere. Note that the sample datasets are not of large scale (each of them contains no more than 6MB), so you may not see the performance benefits when multiple mappers are used.

Part 2: Word Length Count (20%)

Please extend the WordCount program to count the length of each word with the optimization technique *in-mapper combining*. Call the program *WordLengthCount.java*.

Output Format:
Each line contains a tuple of (length, count), separated by a space. For example, (3,5) means that there are a total of five words of length three. The output is not necessarily printed in increasing order of length or count.

Sample Input:
The quick brown fox jumps
over the lazy dog.

Sample Output:
3 3
5 3
4 3

Time Limit: Note that the program should be completed within a reasonable timeframe for a given dataset. The time limit is set by the TAs. Marks will be deducted if the program runs too long.

Note: The punctuations should also be counted toward the length of a word for your convenience. For example, in the sample input, the word "dog." should be counted as a word with length 4 instead of 3.

Part 3: N-gram Initial Count (25%)

Using *WordLengthCount.java* as a starting point, extend it to count *N-gram initials* (call the program *NgramInitialCount.java*). An *N-gram* is a sequence of N consecutive words, where N is the input in the command-line argument. The *initials* of an *N-gram* are the first letters of the N words. See the sample below for deep understanding, assuming that $N = 2$.

Parameter Passing Format

```
$ hadoop jar [.jar file] [class name] [input dir] [output dir] [N]
```

Output Format: Each line contains a tuple of (1st word's initial, 2nd word's initial, count), separated by spaces. In this problem, **ONLY** output the count of N-gram initials which are all in alphabet. You can output the count of each kind of N-gram initials in any order but the initials should be case-sensitive. i.e., "A a" and "a a" should be counted and output separately.

Sample Input:
"The quick brown fox jumps
over the lazy dogs."

Sample Output:
q b 1
b f 1
f j 1
j o 1
o t 1
t l 1
l d 1

Time Limit: Note that the program should be completed within a reasonable timeframe for a given dataset. The time limit is set by the TAs. Marks will be deducted if the program runs too long.

Note: The word in the end of a line and the word in the beginning of the next line **ALSO** form an N-gram. i.e., "jumps over" is a N-gram for $N = 2$. So we need to count "j o". On the other hand, we may miss some N-grams that appear in two different HDFS blocks, and this is okay.

Part 4: Counting N-gram Initials Relative Frequencies (25%)

Extend Part 3's program to compute the N-gram initials relative frequencies (call the program *NgramInitialRF.java*). Again, in the problem, you **ONLY** need to consider the N-gram initials whose initials are in alphabet. The N-gram initials' relative frequency is defined as count("X Y Z")/count("X *") (for $N = 3$), where * stands for any **ALPHABET** initials. See the sample below for deep understanding, assuming that $N = 2$.

Parameter Passing Format

```
$ hadoop jar [.jar file] [class name] [input dir] [output dir] [N] [θ]
```

Output Format: Each line contains a tuple of (1st word's initial, 2nd word's initial, frequency), separated by spaces. You only need to output the N-grams whose relative frequency is at least θ, where θ is

a command-line argument and $0 \leq \theta \leq 1$.

Sample Input:
the quick brown fox jumps over the lazy dogs.
the Quick Brown fox jumps Over the Lazy dogs.
the Quick Brown fox Jumps Over the Lazy "dogs".

Sample Output:if $\theta = 0.6$ then the output should be:
f j 0.66666667
q b 1
b f 1
o t 1
l d 1
d t 1
Q B 1
B f 1
O t 1
L d 1
J O 1

Time Limit: Note that the program should be completed within a reasonable timeframe for a given dataset. The time limit is set by the TAs. Marks will be deducted if the program runs too long.

Note: In the sample, for the N-gram initial "f j", count("f j") = 2, count("f *") = 3, so its frequency is $2/3 \leq \theta$. For the N-gram initial "L d", count("L d") = 1, count("L *") = 1 because we **ONLY** consider alphabet initials. Therefore, (L ") should not be counted into count("L *").

Part 5: Configure Hadoop in Azure (15%)

In this part, you need to demonstrate the following:

1. Configure Hadoop in *fully distributed mode* on Windows Azure, using 4 compute instances that we assign to each group. In particular, you should configure one instance to run as the JobTracker, and all other instances to run as the TaskTrackers.
2. Run the provided WordCount program on Azure.

Please show to the TAs that you achieve the above requirements.

Bonus (5%)

The top 3 groups whose Part 4's programs have the smallest running time will receive the bonus marks. You may consider to optimize your programs or configure some parameters in Hadoop to make the programs perform better. If more than 3 groups have the best performance, we will still give out the 5% bonus to each group. Note that the program must return the correct answer in order to be considered for the bonus mark.

Submission Guidelines

You *must* at least submit the following files, though you may submit additional files that are needed:

- *WordLengthCount.java*
- *NgramInitialCount.java*
- *NgramInitialRF.java*

Your programs must be implemented in Java. Any program that is not compilable will receive zero marks!!

Please refer to the course website for the submission instructions.

The assignment is due on Oct 10 (Thurs), 2013, 23:59:59. Demo will be arranged on the next day.

Have fun! :)