CSCI4180/ESTR4106 (Fall 2018)

Assignment 1: Counting by MapReduce

Due on October 18, 2018, 23:59:59

Introduction

The goals of this assignment are to let students get familiar with Hadoop MapReduce and simple MapReduce development.

Part 1: Getting Started on Hadoop (20%)

In this part, you need to demonstrate the following:

1. Configure Hadoop in *fully distributed mode* on all your assigned virtual machines.
2. Run the provided WordCount program on your Hadoop platform.

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

More instructions will be provided in the tutorials. Make sure that you strictly follow the instructions (e.g., the versions of JDK and Hadoop, the configuration of the software); otherwise the TAs cannot troubleshoot the problems and you will risk losing marks!

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*.

Each line contains a tuple of (length, count), separated by a space character. For example, "3 5" means that there are a total of five words of length three. The output should be printed in the sorted order of the length.

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

Sample Output:
3 3
4 3
5 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: Bigram Initial Count (20%)

Using *WordLengthCount.java* as a starting point, extend it to count *bigram initials* (call the program *BigramInitialCount.java*). A *bigram* is a sequence of two consecutive words. The *initials* of a *bigram* are the first letters of the two words of the bigram. In this problem, we output the count of bigram initials which are both English alphabets (please check Wikipedia what it means by "alphabets").

Each line contains a tuple of (1st word's initial, 2nd word's initial, count), where the two consecutive words are separated by non-alphabet characters (e.g., space, tab, newline, punctuations, or numbers). Note that the initials should be case-sensitive; for example, "A a" and "a a" should be counted and output separately. Here, you may output the counts of bigram initials in any order.

Command Format:
`$ hadoop jar [.jar file] [class name] [input dir] [output dir]`

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

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

Notes:

- We treat the punctuations "`'`" and "`.`" as delimiters. We treat only "`The`" and "`dogs`" as words.
- For a word like "`ca't2e`", we treat it as two words "`ca`" and "`t2e`".
- The word in the end of a line and the word in the beginning of the next line can also form a bigram. For example, "`jumps over`" is a bigram.
- Multiple non-alphanumeric characters are treated as a single delimiter. For example, "`quick ,,,,, brown`" will form a bigram "`quick brown`".
- We may miss some bigrams that appear in two different HDFS blocks, but it's okay.
- You don't need to implement in-mapper combining here, but you are strongly encouraged to think of any possible way to speed up your program.

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.

Part 4: Counting Bigram Initials Relative Frequencies (20%)

Extend Part 3's program to compute the bigram initials relative frequencies (call the program *BigramInitialRF.java*). Again, in the problem, you only need to consider the bigram initials that are English alphabets. The relative frequency of a bigram initial is defined as count("X Y")/count("X *") where * stands for an alphabet initial.

Each line contains a tuple of (1st word's initial, 2nd word's initial, frequency). You only need to output the bigrams whose relative frequency is at least θ, where θ is a command-line argument and $0 \leq \theta \leq 1$.

Command Format:
`$ hadoop jar [.jar file] [class name] [input dir] [output dir] [theta]`

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.

Notes:

- For the bigram initial "f j", since count("f j") = 2 and count("f *") = 3, the relative frequency is $2/3 \leq \theta$.
- For the bigram initial "L d", since count("L d") = 2 and count("L *") = 2 (note that we treat "'" as a delimiter).

1 Part 5: Configure Hadoop on Amazon EC2 (20%)

In this part, you need to demonstrate the following:

1. Configure Hadoop in fully distributed mode on Amazon EC2, using three compute instances.
2. Run the provided WordCount program on EC2.

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

Bonus (5%)

The top 3 groups whose Part 4s 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*
- *BigramInitialCount.java*
- *BigramInitialRF.java*

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

The testing platform during demo is provided by the TAs. If you need to change the configuration of Hadoop, make sure that you do it dynamically inside the code. You are not allowed to modify any configuration files during demo.

Please refer to the course website for the submission instructions.

The assignment is due on Oct 18, 2018, 23:59:59. Demo will be arranged on the next day.

Have fun! :)