CSCI4180 (Fall 2013)

Assignment 2: HBase and Parallel BFS

Due on November 14, 2013, 23:59:59

1 Introduction

The first part of this assignment is to let students get familiar with running HBase on top of Hadoop. In the second part, students will have a chance to execute graph algorithms on top of Hadoop. In particular, parallel breath-first search will be used to find out the shortest path lengths from a single source in a Twitter social graph.

2 Part 1: Bigram Counting with HBase (50%)

This part is composed of three sub-parts.

Import: You need to first configure HBase correctly. Both HBase and HDFS must be hosted in the fully distributed mode. Then load the source data into a HBase table. Specifically, you need to create a table, in which each row stores a *single line of words*. You can design your own schema format for storing the original text. In this assignment, we will use two data sets, KJV Bible and Gutenberg, as in Assignment 1.

Write a helper program called *HBaseImport.java* that takes a command-line argument "directory" and loads all text files in the directory into HBase.

Counting: After that, you need to count the number of times that each bigram appears in each data set. A bigram is defined as two consecutive words. The last word in a line and the first word in the next line also form a bigram, for example, for the following sentence

"These words are in the fist line, and
these words are in the second line."

"**and these**" is also a bigram.

In this assignment, we add some requirements for the definition of bigrams:

1. Please use any non-English alphabet characters (i.e., anything except [a-zA-Z]) as delimiters. For example, any punctuation marks are delimiters. In the previous example, "line and" *is* a bigram.
2. Please change all characters to lower case. For example, "These" is the same as "these".

The output will be stored in another HBase table, in which each row stores the tuple *(bigram, count)*. The output table should be named **bigram_result**. The key of the table is the bigram itself (e.g. "t w"). You should store the count of each bigram in the column **result:count**. Marks will be deducted if you violate the naming convention.

Write a MapReduce program called *HBaseBigram.java* that loads the data from HBase, counts the Bigram, and writes the results to HBase.

Export: You should write a program that dumps the result from HBase. The program will take a command-line argument θ, which only outputs the bigrams that have the count at least θ to stdout in the following format.

Sample Output:

```
q b 2
b f 1
f j 1
```

Write a helper program called *HBaseExport.java* that outputs the results from HBase.

Note: You are allowed to submit an HBase script (e.g. init.hb) to create the table schema (but not inserting data). We will execute the script by calling the command `hbase shell init.hb` before the demo.

3 Part 2: Single-source Shortest Path Lengths by Parallel BFS (50%)

In this part, you will need to write a map-reduce program to compute the **shortest path lengths** from a given **source node** in a graph dataset extracted from Twitter. This can be achieved by parallel breadth-first search which is an iterative map-reduce algorithm. Details about the algorithm will be covered in tutorial. The Twitter dataset is obtained from the following reference:

- Haewoon Kwak, Changhyun Lee, Hosung Park, Sue Moon.
  "What is Twitter, a Social Network or a News Medium".
  19th World-Wide Web (WWW) Conference, April 2010.
  URL: http://an.kaist.ac.kr/traces/WWW2010.html

Definition: We model the Twitter network as a directed graph. Each user is represented as a *node* with a unique positive integer as the nodeID. When user 1 "follows" user 2 on Twitter, an *edge* is created from node 1 to node 2 in the graph.

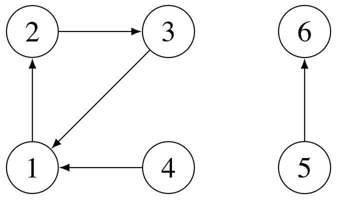
Problem: Given a graph $G = (V, E)$ and a source node $v_s \in V$, find the shortest path length (distance) from v_s to every other reachable node in V.

Input Format: Each line contains a tuple of (nodeID, nodeID), separated by spaces. Each tuple indicates a directed edge from the former node to the latter node.

Output Format: Each line contains a tuple of (nodeID, distance), separated by spaces. Only output tuples for nodes that are reachable from v_s which is given as a command-line argument.

Sample Input:

```
1 2
3 1
4 1
2 3
5 6
```



Sample Output: (for $v_s = 1$)
1 0
2 1
3 2

Note: Since there is no path going to node $4, 5$ or 6 from v_s, tuples corresponding to these nodes should not be shown in the output.

Hint: You may want to implement a different set of mapper/reducer to parse the input and transform it to adjacency list format. Throughout the iterative map/reduce process, feel free to store and retrieve any intermediate files in your own format on the Hadoop DFS. We would only look at your final output for grading.

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.

Bonus (5%)

The top 3 groups who have the total shortest running time for their programs in *both Parts 1 and 2* will receive the bonus marks. You may consider optimizing your programs or configuring 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:

Part 1:

- HBaseImport.java
- HBaseBigram.java
- HBaseExport.java

Part 2:

- ParallelBfs.java

Your separate MapReduce scripts must be implemented in Java.

The assignment is due on November 14, 2013, 23:59:59. Demo will be arranged on the next day. Have fun! :)