CSCI4180 (Fall 2022)

Assignment 2: Amazon EC2 and Iterative MapReduce

Due on November 17 (Thur), 2022, 23:59:59

Introduction

In all parts, you only need to configure Hadoop in *pseudo-distributed mode*. For the bonus part, you will need to configure Hadoop in *fully distributed mode*.

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

In this part, you need to demonstrate the following:

1. Configure Hadoop in pseudo-distributed mode on an Amazon EC2 instance.
2. Run the provided WordCount program on EC2.

Please show to the TAs that you can run Hadoop on Amazon EC2 during demos.

1 Part 2: Single-source Shortest Path Lengths by Parallel Dijkstra's Algorithm (40%)

In this part, you will need to write a MapReduce program to compute the **shortest path lengths** from a given **source node** in a graph dataset extracted from Twitter. We implement the algorithm via the parallel Dijkstra's algorithm. 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

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. Also, we attach a positive integer weight to each edge.

Problem: Given a graph $G = (V, E)$ and a source node $v_s \in V$, find the shortest path distance from v_s to every other reachable node in V. The source node v_s is provided as a command-line argument.

Input Format: Each line contains a tuple of (nodeID, nodeID, weight), 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, prev), separated by spaces, where "distance" means the shortest path distance, and "prev" means the incoming node that lies on the shortest path (we set "prev" as "nodeID" for the source node). Only output tuples for nodes that are reachable from v_s which is given as a command-line argument.

Sample Command:

```
hadoop jar [.jar file] ParallelDijkstra [infile] [outdir] [src] [iterations]
```

Sample Input:

1 2 7
1 3 20
2 3 3
3 1 5
4 1 9
5 6 10

Sample Output:

1 0 1
2 7 1
3 10 2

Notes:

- The sample assumes that the source node is 1.
- Since there is no path going to node 4, 5 or 6 from v_s, the tuples corresponding to these nodes should not be shown in the output.
- Your program (call it *ParallelDijkstra.java*) will take a command-line argument `Iterations` to indicate the maximum number of MapReduce iterations (the program may finish earlier if all shortest path distances have been found). We assume that `Iterations` are always a positive number.
- You will need to implement a class of the node structure (call it *PDNodeWritable.java*) to define the node attributes, such as adjacency lists.
- You will need to write a separate MapReduce program (call it *PDPreProcess.java*) to convert the input files into *adjacency list* format first.

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.

2 Part 3: PageRank Algorithm (40%)

In this part, you will need to write a MapReduce program to compute the PageRanks of nodes. We still use the same Twitter dataset in Part 1 by treating each node in the dataset as a "page" for the PageRank algorithm.

Problem: Given a graph $G = (V, E)$, find the PageRank values of all nodes in V. We do not consider the random jump and dangling nodes in this assignment. If there is indeed any dangling node, we do

not redistribute its PageRank mass during the map phase and the PageRank of the dangling node remains unchanged during the map phase (instead of zero).

Input Format: Each line contains a tuple of (nodeID, nodeID, weight), separated by spaces. Each tuple indicates a directed edge from the former node to the latter node. We ignore the edge weights in this problem, as they are not needed by the PageRank algorithm that we taught in class.

Output Format: Each line contains a tuple of (nodeID, PageRank value), separated by spaces. We only output tuples for nodes whose PageRank values are above certain threshold, where the threshold value (between 0 and 1) is given as a command-line argument.

Sample Command:

```
hadoop jar [.jar file] PageRank [iteration] [threshold] [infile] [outdir]
```

Notes:

- Your program (call it *PageRank.java*) will execute the PageRank algorithm over a fixed number of iterations. The program will take a command-line argument `Iterations` to indicate the number of MapReduce iterations needed to be executed. We assume that `Iterations` is at least one.
- As in Part 1, we need to implement a class of node structure (call it *PRNodeWritable.java*) and write a MapReduce program to convert the input files into adjacency list format (call it *PRPreProcess.java*).

Bonus (5%)

(a) (2%) Configure Part 2's program to run on Hadoop in fully distributed mode.

(b) (3%) The top 3 groups whose Part 2's program have the smallest running time in fully distributed mode 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 bonus 3% to each group.

Note that the program must return the correct answer in order to be considered for the bonus mark.

Notes

- To simplify our grading, we require that both parts use only a single reducer (by default, the number of reducers is one).

Submission Guidelines

Please at least submit the following files. Additional files are allowed.

Part 2:

- ParallelDijkstra.java
- PDNodeWritable.java

- PDPreProcess.java

Part 3:

- PageRank.java
- PRNodeWritable.java
- PRPreProcess.java

Declaration form for group projects:

- (`http://www.cuhk.edu.hk/policy/academichonesty/Eng_htm_files_(2013-14)/p10.htm`)

Demo will be arranged on the following day of the deadline. Have fun! :)