Lab 7 PageRank

1 PageRank

One of the biggest changes in our lives in the decade was the availability of efficient and accurate web search. Google is the first search engine which is able to defeat the spammers who had made search almost useless. The technology innovation behind Google is called PageRank. This project is to implement the PageRank to find the most important Wikipedia pages on the provided the adjacency graph extracted from Wikipedia dataset using AWS Elastic MapReduce(EMR).

1.1 Definition of PageRank

PageRank is a function that assigns a real number to each page in the Web. The intent is that the higher the PageRank of a page, the more important it is. The equation is as follows:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} \quad (1)$$

Where d = 0.85, $p_1,p_2,...,p_N$ are the pages under consideration, $M(p_i)$ is the set of pages that link to p_i, $L(p_j)$ is the number of outbound links on page p_j, and N is the total number of pages. In our project, **we do not use teleport to deal with the sink node for simplicity.** At the initial point, each page is initialized with PageRank 1/N and the sum of the PageRank is 1. But the sum will gradually decrease with the iterations due to the PageRank leaking in the sink nodes.

Similar to Lab5, you will need to use Java + AWS Mapreduce for this lab.

2. Input and Output

You program should take two args, an input path and an output path, similar to lab5. The input path contains the output graph from lab5, like following:

input_path/
- *Part-0000*
- *Part-0001*
- *....*

Your output path should contain results in layout like this:

output_path/
- *num_nodes -- contains the number of nodes N (page titles) calculated from input graph.*
- *iter1.out --- contains result after 1st iteration of pagerank score calculation.*
- *iter8.out --- contains result after 8th iteration of pagerank score calculation.*
- *temp/ -- contains intermediate files that will be ignored by TA.*

Please note:

1. The ‘num_nodes’ file should just contain an integer number N, which will be used for PageRank score calculation.
2. Files iter1.out, iter8.out should only contain nodes that have >= 5/N pagerank score, and should be sorted in descending order of PageRank score. Here you will need to override the default sorter to sort in decreasing order(extends WritableComparator).

Example input & output can be found here.
You will need to write MapReduce jobs for the above tasks (calculating N, calculating pagerank scores, and sorting). You can use JAVA + HDFS api for reading N, renaming/moving files around if necessary.

3. What to submit

Your submission should include your source code, and a runnable Jar file that handles the input and output mentioned above.
Plea
Your submission layout should be exactly like the following to facilitate the TA’s grading:
Please also include a report on

1. how you calculate pagerank score
2. What challenges or difficulties you have for this lab.

Firstname_Lastname/ *-- Your firstname and lastname as shown on Canvas*
src/ *-- source code only, no deps or project folder.*
report.pdf
pagerank.jar *-- your jar file should be named exactly as shown here.*

Compress your folder as a zip file and submit it.
You submission layout, output layout and file names should be exactly as specified above, otherwise a 30% penalty would be applied.