Final Project Report

–Big Data Processing By G10

Scope of the project

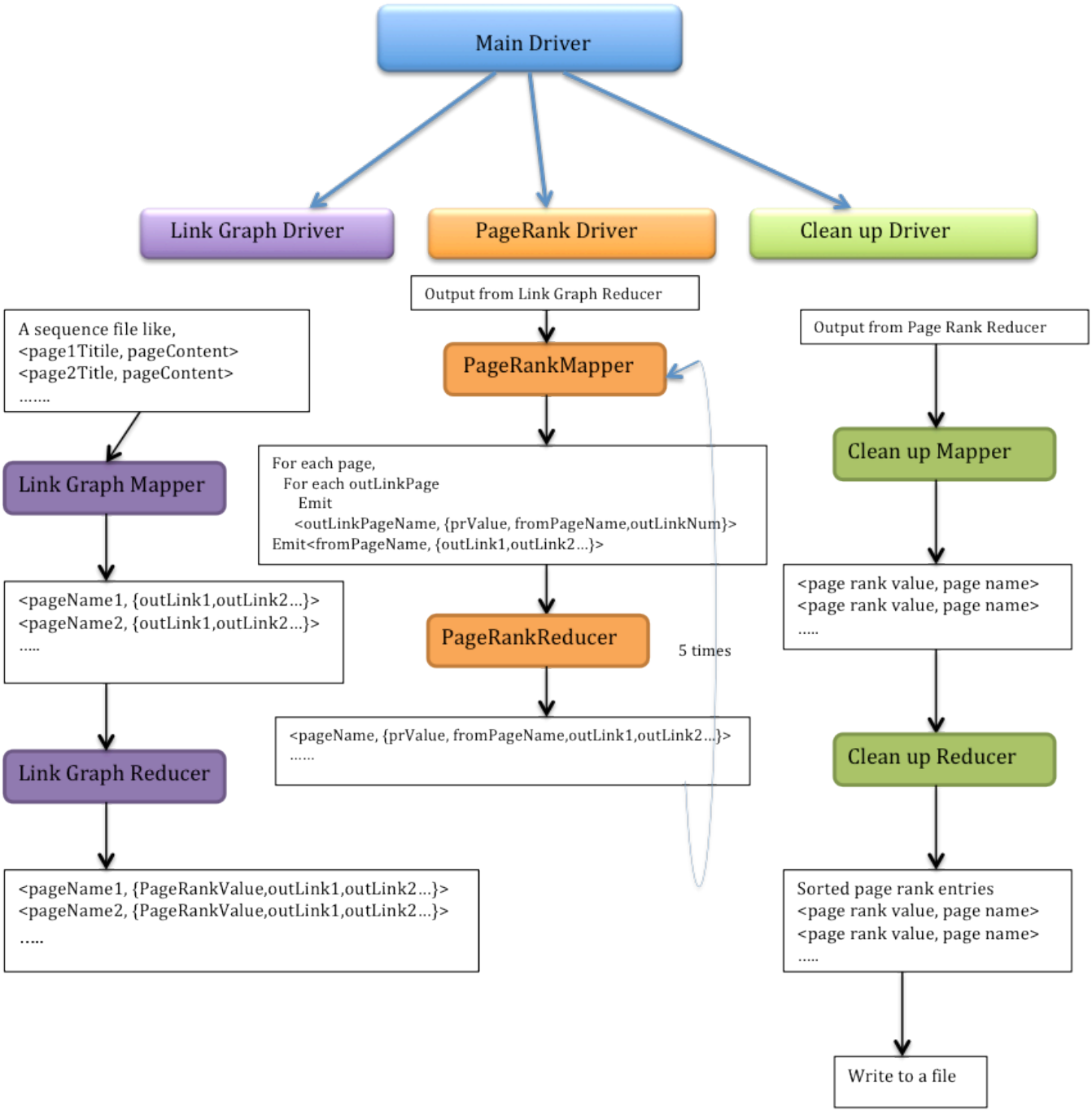
In this project, we designed and implemented a mini search engine, which is handles large-scale data. Based on the former homework, we added MapReduce technique to the system as our new major feature. MapReduce is a programming model and an associated implementation for processing and generating large data sets. We specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.

We start with the following hypotheses:
(1) No Link Spam;
(2) Web page content is static;
(3) Initial PageRank value is 0.15.

The system follows these steps to operate:
(1) Transfers a large number of individual small files in to a sequence file.
(2) Parses all the page-to-page links out of the page bodies, and maps each page to a list of the pages it links to. Set initial PageRank value.
(3) Processes 5 MapReduce passes sequentially. Each pass we used $0.85 \times Sum(PageRank) + 0.15$ to calculate PageRank values.
(4) Delete intermediate output files and Sort PageRank.

Architecture

PageRank Part



Approach Details

Classes

FileProcessor

We use the wiki corpus provided in the homework. Instead of using the files directly, we transfer the multiple small files to a single large file. In this single large file, each line represents a small original file.

The output sequence file is in the format as following:

<DocTitle1 content>
<DocTitle2 content>
<DocTitle3 content>
….

MainDriver

Main method is in this class.
It is the controller of *Link Graph phrase*, *PageRank phrase* and *Cleanup phrase*.
Parameter args specifies input and output path. Input is the output from class FileProcessor. Output is a file in the format as following and sorted on PageRank value in descending order.

<PageRank Value DocTitle1>
<PageRank Value DocTitle2>
….

LinkGraphDriver

Controller for *Link Graph Phase*, which set configurations for the MapReduce job that generates Link-Graph.
The input is the single large file generated from multiple small files in data/wiki corpus, which comes from MainDriver's input.
The output is Link Graph file in the following format:

< DocTitle1 PageRank Value OutLink1 OutLink2…>
< DocTitle2 PageRank Value OutLink1 OutLink2…>
….

LinkGraphMapper

Input key: line offset of input file
Input value: the whole line content
Output key: document title
Output value: OutLink
Process phrase: 1.Extract document from line content
2. Extract visible out links from line content
3. Output pairs like <DocTitle, OutLink>

LinkGraphReducer

Input key: DocTitle
Input value: <DocTitle, {OutLink1, OutLink2…}>

Output key: DocTitle
Output value: initial PageRank, OutLink1, OutLink2…
Process phrase: Add initial PageRank value in output value. Initial is set to be 0.15.

PageRankDriver

Controller for PageRank Phrase sets configurations for the MapReduce job, which calculates PageRank value. Since PageRank is an iterative algorithm, this MapReduce should run several times. We run 5 iterations in this system, as a result, the final output PageRank value is the result after 5 passes.
Input: output file from LinkGraphDriver
Output: 5 temporary files, each from per iteration. The output in the following format:
<DocTitle1 PageRank Value OutLink1 OutLink2…>
<DocTitle2 PageRank Value OutLink1 OutLink2…>
….
Since we need to iterate several times, input of Mapper and output of Reducer should be the same.

PageRankMapper

Input key: line offset
Input value: PageRank value, OutLink1, OutLink2…
Outputs of this Mapper are in two different formats.
Format 1
Output key: OutLink document title
Output value: PageRank value, from document title, number of out links in from document title
Format 2
Output key: from document title
Output value: OutLink1, OutLink2, OutLink3…
Process Phrase: 1. For each out link in one input value, emit outputs in format1
2. For each input value, emit a record in format2
We used different separators to distinguish the two kinds of output formats

PageRankReducer

Input key: document title
Input value:
Format1: PageRank value, from document title, number of out links in from document title
Format2: OutLink1, OutLink2, OutLink3…
Output key: document title
Output value: PageRank value, OutLink1, OutLink2, OutLink3…
Process Phrase: 1. Update PageRank value using

$$\left(\sum_{forAllFromPages} \frac{fromPageRankValue}{numberOfOutLinks}\right) + initalPageRankValue$$

2. Add the new PageRank value and all out links of this page to new output value

PageRankCleanUp
Method cleanupMain: Controller for PageRank Phrase sets configurations for the MapReduce job, which sorts PageRank and delete temporary files.
For Mapper:
Input key: document title
Input value: PageRank value, OutLink1, outlink2, OutLink3…
Output key: PageRank value (multiply 10,000 to make the result more readable)
Output value: document title
Process phrase: Extract PageRank value and document title
For Reducer:
Just a simple identity function

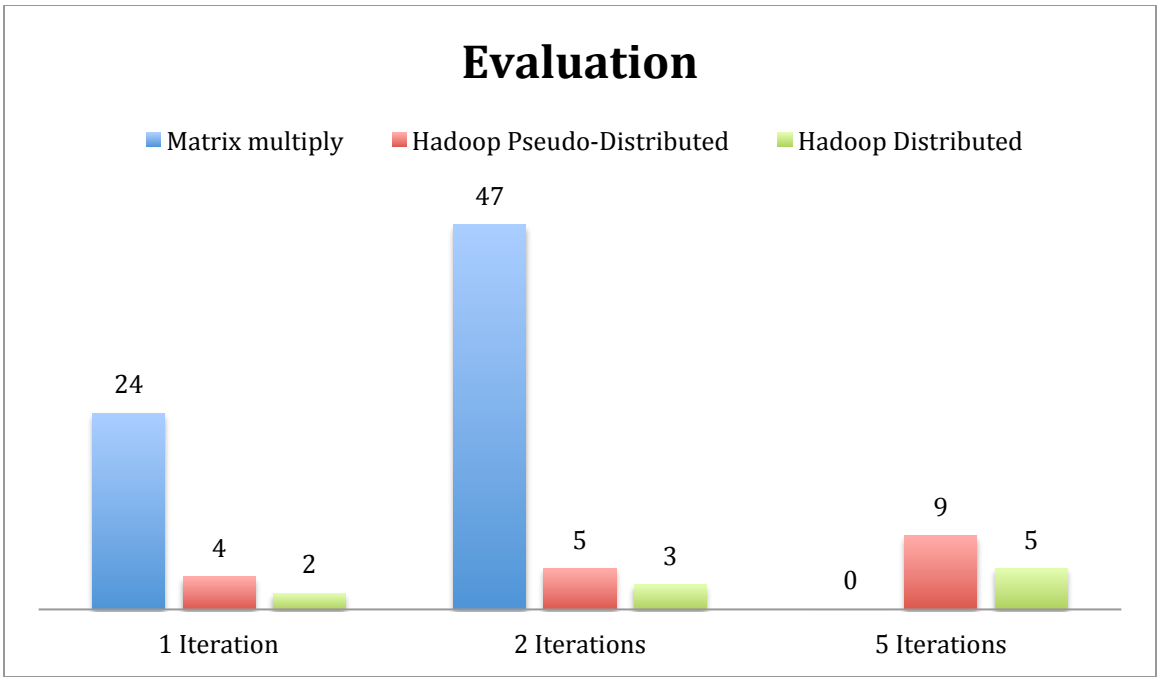
How to compile and run

1. Set up Hadoop environment
2. *mkdir pageRank_classes*
3. *javac -classpath folderContainsHadoopFiles&ScorceCode/hadoop-1.0.4/hadoop-core-1.0.4.jar -d pageRank_classes src/edu/nyu/cs/cs2580/*.java*
4. *jar -cvf folderContainsHadoopFiles&ScorceCode /pageRank.jar -C pageRank_classes/ .*
5. *hadoop-1.0.4/bin/hadoop fs -put input input (first input is the folder contains sequence file which generated from FileProcessor)*
6. *hadoop-1.0.4/bin/hadoop jar folderContainsHadoopFiles&ScorceCode/pageRank.jar edu.nyu.cs.cs2580.MainDriver ./input ./output*
7. *hadoop-1.0.4/bin/hadoop dfs -cat output/part-00000 > result*

Evaluation

The main reason we use MapReduce to process big data is that MapReduce provides a more efficient way to calculate PageRank for each page in large corpus. So the computing time is the key measure.

Compare the time consumed to compute PageRank of a settled corpus in different ways



We did not measure Matrix multiply-5 Iterations; it takes pretty long time

Conclusion

Our implementation of MapReduce runs on a large cluster of commodity machines and is scalable. According to the results of this experiment, MapReduce improves the efficiency of the calculation of PageRank Value obviously.