Matrix Multiplication on Map-Reduce

Description

The purpose of this project is to develop a Map-Reduce program on Hadoop to multiply two sparse matrices.

This project must be done individually. No copying is permitted. **Note: We will use a system for detecting software plagiarism, called**[**Moss (Links to an external site.)**](http://theory.stanford.edu/~aiken/moss/)**, which is an automatic system for determining the similarity of programs.** That is, your program will be compared with the programs of the other students in class as well as with the programs submitted in previous years. This program will find similarities even if you rename variables, move code, change code structure, etc.

Note that, if you use a Search Engine to find similar programs on the web, we will find these programs too. So don't do it because you will get caught and you will get an F in the course (this is cheating). Don't look for code to use for your project on the web or from other students (current or past). Just do your project alone using the help given in this project description and from your instructor and GTA only.

Platform

You will develop your program on SDSC Comet. Optionally, you may use your laptop/PC to develop your program first and then, after you make sure that it works there, you transfer it and test it to Comet. You may also use IntelliJ IDEA or Eclipse to help you develop your program, if you have done so in Project 1. Note that it is required that you test your programs on Comet before you submit them.

Setting up your Project

Login into Comet and download and untar project2:

wget http://lambda.uta.edu/cse6331/project2.tgz

tar xfz project2.tgz

chmod -R g-wrx,o-wrx project2

Go to project2/examples and look at the Map-Reduce example Join.java. You can compile src/main/java/Join.java using:

run example.build

and you can run it in standalone mode using:

sbatch example.local.run

Project Description

For this project, you are asked to implement matrix multiplication in Map-Reduce. This is described in Section 2.3.9 (page 38) in [Map-Reduce and the New Software Stack](https://uta.instructure.com/courses/52914/files/9798115/download)

[Actions](https://uta.instructure.com/courses/52914/assignments/749562)

. You should use two Map-Reduce jobs in the same Java file project2/src/main/java/Multiply.java. Do not use the method in Section 2.3.10. **You should modify Multiply.java only**. In your Java main program, args[0] is the first input matrix M, args[1] is the second input matrix N, args[2] is the directory name to pass the intermediate results from the first Map-Reduce job to the second, and args[3] is the output directory. Like in Project 1, the input file format for reading the input matrices and the output format for the final result must be text formats, while the format for the intermediate results between the Map-Reduce jobs must be binary formats. There are two small sparce matrices 4\*3 and 3\*3 in the files M-matrix-small.txt and N-matrix-small.txt for testing in standalone mode. Their matrix multiplication must return the 4\*3 matrix in solution-small.txt. Then there are 2 moderate-sized matrices 200\*100 and 100\*300 in the files M-matrix-large.txt and M-matrix-large.txt for testing in distributed mode. Their matrix multiplication must return the matrix in solution-large.txt.

You can compile Multiply.java using:

run multiply.build

and you can run it in standalone mode over the two small matrices using:

sbatch multiply.local.run

The result matrix in the directory output must be similar to result-matrix-small.txt. You should modify and run your programs in standalone mode until you get the correct result. After you make sure that your program runs correctly in standalone mode, you run it in distributed mode using:

sbatch multiply.distr.run

This will multiply the moderate-sized matrices and will write the result in the directory output-distr. Note that running in distributed mode will use up at least 10 of your SUs. So do this once or twice only, after you make sure that your program works correctly in standalone mode.

Pseudo-Code

To help you, I am giving you the pseudo code:

class Elem extends Writable {

short tag; // 0 for M, 1 for N

int index; // one of the indexes (the other is used as a key)

double value;

...

}

class Pair extends WritableComparable<Pair> {

int i;

int j;

...

}

(Add methods toString so you can print Elem and Pair.) First Map-Reduce job:

map(key,line) = // mapper for matrix M

split line into 3 values: i, j, and v

emit(j,new Elem(0,i,v))

map(key,line) = // mapper for matrix N

split line into 3 values: i, j, and v

emit(i,new Elem(1,j,v))

reduce(index,values) =

A = all v in values with v.tag==0

B = all v in values with v.tag==1

for a in A

for b in B

emit(new Pair(a.index,b.index),a.value\*b.value)

Second Map-Reduce job:

map(key,value) = // do nothing

emit(key,value)

reduce(pair,values) = // do the summation

m = 0

for v in values

m = m+v

emit(pair,m)

What to Submit

You need to submit the following files:

project2/src/main/java/Multiply.java

project2/multiply.local.out

project2/output-distr/part-r-00000

project2/multiply.distr.out