Project 2

| 10/7/2022

100/100 Points

Attempt 1

REVIEW FEEDBACK
10/2/2022

Attempt 1 Score:
100/100

Add Comment

Unlimited Attempts Allowed

Details

Block Matrix Addition using Map-Reduce

Description

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

This project must be done individually. No copying is permitted. **Note: We will use a system for detecting software plagiarism, called Moss (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 GTAs only.

Platform

You will develop your program on SDSC Expanse. Optionally, you may use your laptop/PC to develop your program first and then, after you make sure

if you have

Try Again

Project Description

For this project, you are asked to implement block matrix addition in Map-Reduce. You will take two sparse matrices as input, convert them to block matrices, and then perform block matrix addition on them. The sum of two matrices `M` and `N` is the matrix `R` such that $R_{ij}=M_{ij}+N_{ij}$. A sparse matrix is a dataset of triples `(i,j,v)`, where `i` and `j` are the indices and `v` is that matrix value at the indices `i` and `j`. A triple is constructed with the Java class Triple. A block matrix is a dataset of blocks. Each block is a dense Java matrix of size rows*columns constructed with the Java class Block (here, rows=columns=100). A block matrix is stored in HDFS as a binary file (in SequenceTextInputFormat) of key-values, where the key is a pair of block coordinates (ci,cj) (constructed with the Java class Pair) and the value is a block. A matrix element M_{ij} is stored inside the block with block coordinates (i/rows,j/columns) at the location (i%rows,j%columns) inside the block. The block matrix addition of M and N is done by finding blocks from M and N with the same block coordinates and by adding the blocks together using regular matrix addition in Java.

Your project is to convert two sparse matrices M and N which are read from files to block matrices and then add them using block matrix addition. First, you need to convert a sparse matrix to a block matrix using the following Map-Reduce pseudo-code (called twice, for M and for N):

```
map ( key, line ):
  read i,j,v from the line (delimiter is ",")
  emit( Pair(i/rows,j/columns), Triple(i%rows,j%columns,v) )

reduce ( pair, triples ):
  b = empty block
  for each (i,j,v) in triples:
     b[i,j] = v
  emit( pair, b )
```

Block matrix addition is done using a reduce-side join (see pages 27-30 of [bigdata-l03.pdf (https://uta.instructure.com/courses/118779/files/21361934?wrap=1)](https://uta.instructure.com/courses/118779/files/21361934?wrap=1)) using the following pseudo-code:

```
map1 ( pair, block ):
  emit( pair, block )  // do nothing

map2 ( pair, block ):
  emit( pair, block )  // do nothing

reduce ( pair, blocks ):
  s = empty block
  for b ir
     s +=
  emit( pa
```

Try Again

You should use three Map-Reduce jobs in the same Java file `project2/src/main/java/Add.java`, two for converting the input matrices to block matrices and one for block matrix addition. **You should modify Add.java only**. First, you need to add code for the methods in the classes Pair, Triple, and Block. In your Java main program, args[0] is the first input matrix M, args[1] is the second input matrix N, args[2] and args[3] are the temporary output directories of the first 2 map-reduce programs, and args[4] is the final output directory. The input and output of your program must be files in text formats but the intermediate results of your Map-Reduce jobs must be Sequence file formats. There are two small sparse matrices of sizes 4*3 in the files M-matrix-small.txt and N-matrix-small.txt for testing in local mode. Your job final results must be similar to those in `solution-small.txt`. Then, there are 2 moderate-sized matrices 4000*3000 in the files M-matrix-large.txt and N-matrix-large.txt for testing in distributed mode (located on Expanse). So each block matrix will be 40*30 blocks. Your job final results must be similar to those in `solution-large.txt`.

Setting up your project on your laptop

You can use your laptop to develop your program and then test it and run it on Expanse. Note that testing and running your program on Expanse is required. If you do the project on your laptop, download and untar project2:

Note 09/28/2022: The project2.tgz has been updated. Please download the new one.

```
wget https://lambda.uta.edu/cse6332/project2.tgz
tar xfz project2.tgz
```

Note 10/01/2022: Replace the following line on both add.local.run and add.distr.run (use uot182 instead of uot143):

```
SW=/expanse/lustre/projects/uot182/fegaras
```

To compile and run project2 on your laptop:

```
cd project2
mvn install
rm -rf left_tmp right_tmp output
~/hadoop-3.2.2/bin/hadoop jar target/*.jar Add M-matrix-small.txt N-matrix-small.txt left_tmp right_tmp output
```

The file `output/part-r-00000` will contain the results which must be the same as in solution-small.txt. After you make sure that the project works corr

Try Again

Setting up your project on Expanse

This step is required. If you'd like, you can develop this project completely on Expanse. If you have already developed project2 on your laptop, copy `project2.tgz` from your laptop to Expanse. Otherwise, download project2:

```
wget https://lambda.uta.edu/cse6332/project2.tgz
tar xfz project2.tgz
chmod -R g-wrx,o-wrx project2
```

You can compile `Add.java` on Expanse using:

```
run add.build
```

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

```
sbatch add.local.run
```

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

```
sbatch add.distr.run
```

This will write the result in the directory `output-distr` and must be similar to `solution-large.txt`.

What to submit

On Expanse, make sure that the following files in your project2 directory exist and are correct:

```
project2/src/main/java/Add.java
project2/add.local.out
project2/output/part-r-00000
project2/output-distr/part-r-00000
project2/add.distr.out
```

Archive yc

Try Again

```
tar cfz project2.tgz project2
```

On your laptop, copy project2.tgz from Expanse to your laptop:

```
scp xyz1234@login.expanse.sdsc.edu:project2.tgz ./
```

Finally, upload your project2.tgz from your laptop using this web page

Preview Unavailable

project2.tgz

Download

(https://uta.instructure.com/files/22640183/download?download_frd=1&verifier=0MQyPTE9GzECSKmZ1aARhB3kw6d153f1mbVr8Uzb)

Try Again