Project 1

New Attempt

Due Sep 28 by 11:59pm **Points** 100 **Submitting** a file upload

A Simple Map-Reduce Program

Description

The purpose of this project is to develop a simple Map-Reduce program on Hadoop for graph processing.

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 GTA only.

Platform

You will develop your program on your laptop and then on **SDSC Expanse**. Optionally, you may use IntelliJ IDEA or Eclipse to help you develop your program on your laptop, but you should test your programs on Expanse before you submit them.

How to develop your project on your laptop

You can use your laptop to develop your program and then test it and run it on Expanse. This step is optional but highly recommended because it will save you a lot of time. Note that testing and running your program on Expanse is required.

If you have Mac OS or Linux, make sure you have Java and Maven installed. On Mac, you can install Maven using Homebrew: `brew install maven`. On Ubuntu Linux, use: `apt install maven`.

On Windows 10, you need to install **Windows Subsystem for Linux (WSL 2)** **(https://docs.microsoft.com/en-us/windows/wsl/install-win10)** and then Ubuntu 20.04 LTS. It's OK if you have WSL 1 or an older Ubuntu. Then, open a unix shell (terminal) on WSL2 and do:

```
sudo apt update
sudo apt upgrade
sudo apt install openjdk-8-jdk maven
```

To install Hadoop and the project on Mac, Linux, or Windows WSL2, cut&paste and execute on the unix shell:

```
cd
wget https://mirror.nodesdirect.com/apache/hadoop/common/hadoop-3.2.2/hadoop-3.2.2.tar.gz
tar xfz hadoop-3.2.2.tar.gz
wget http://lambda.uta.edu/cse6331/project1.tgz
tar xfz project1.tgz
```

You should also set your JAVA_HOME to point to your java installation. For example, on Windows 10 do:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

To test Map-Reduce, go to `project1/examples/src/main/java` and look at the two Map-Reduce examples `Simple.java` and `Join.java`. You can compile both Java files using:

```
cd
cd project1/examples
mvn install
```

and you can run Simple in standalone mode using:

```
~/hadoop-3.2.2/bin/hadoop jar target/*.jar Simple simple.txt output-simple
```

The file `output-simple/part-r-00000` will contain the results.

To compile and run project1:

```
cd project1
mvn install
rm -rf output
~/hadoop-3.2.2/bin/hadoop jar target/*.jar Graph small-graph.txt intermediate output
```

The file `output/part-r-00000` will contain the results which must be equal to:

```
2 2
3 2
4 1
5 2
7 1
```

(also in the file small-solution.txt). The results of the first Map-Reduce job must be equal to that in the file small-intermediate.txt. After your project works correctly on your laptop (it produces the same results as the solution), copy it on Expanse:

```
cd
rm project1.tgz
tar cfz project1.tgz project1
scp project1.tgz xyz1234@login.expanse.sdsc.edu:
```

where `xyz1234` is your Expanse username.

Setting up your Project on Expanse

This step is required. If you'd like, you can develop this project completely on Expanse. Follow the directions on How to login on Expanse at **SDSC Expanse**. Please email the GTA if you need further help.

First, allow password-less login to local host (without it, you can't run Map-Reduce). First do `ssh localhost` using your password and exit using control-D. Then do:

```
ssh-keygen -t rsa
```

(press enter at each line). Then do:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod og-wx ~/.ssh/authorized_keys
```

Now you should be able to `ssh localhost` without a password.

Then, edit the file `.bashrc` (note: it starts with a dot) using a text editor, such as `nano .bashrc`, and add the following lines at the end (cut-and-paste):

```
module load gcc openjdk
module load slurm sdsc
SW=/expanse/lustre/projects/uot143/fegaras
alias run='srun --pty -A uot143 --partition=shared --nodes=1 --ntasks-per-node=1 --mem=2G -t 00:05:00 --wait=0 --export=ALL'
```

logout and login again to apply the changes. If you have already developed project1 on your laptop, copy `project1.tgz` from your laptop to Expanse. Otherwise, download project1 from the class web site:

```
wget http://lambda.uta.edu/cse6331/project1.tgz
```

Untar it:

```
tar xfz project1.tgz
rm project1.tgz
chmod -R g-wrx,o-wrx project1
```

Go to `project1/examples` and look at the two Map-Reduce examples `src/main/java/Simple.java` and `src/main/java/Join.java`. You can compile both Java files using:

```
run example.build
```

and you can run them in standalone mode using:

```
sbatch example.local.run
```

The file `example.local.out` will contain the trace log of the Map-Reduce evaluation while the files `output-simple/part-r-00000` `output-join/part-r-00000` will contain the results.

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

```
run graph.build
```

and you can run Graph.java in standalone mode over a small dataset using:

```
sbatch graph.local.run
```

The results generated by your program will be in the directory `output`. These results should be:

```
2 2
3 2
4 1
5 2
7 1
```

(also in the file small-solution.txt). The results of the first Map-Reduce job must be equal to that in file small-intermediate.txt. You should develop 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 graph.distr.run
```

This will process the graph on the large dataset `large-graph.txt` and will write the result in the directory `output-distr`. These results should be similar to the results in the file `large-solution.txt`. Note that running in distributed mode will use up at least 10 of your SUs. So do this a couple of times only, after you make sure that your program works correctly in standalone mode. You can check your SUs using: `expanse-client user`

Project Description

In this project, you are asked to implement a simple graph algorithm that needs two Map-Reduce jobs. A directed graph is represented as a text file where each line represents a graph edge. For example,

```
20,40
```

represents the directed edge from node 20 to node 40. First, for each graph node, you compute the number of node neighbors. Then, you group the nodes by their number of neighbors and for each group you count how many nodes belong to this group. That is, the result will have lines such as:

```
10	30
```

which says that there are 30 nodes that have 10 neighbors. To help you, I am giving you the pseudo code. The first Map-Reduce is:

```
map ( key, line ):
  read 2 long integers from the line into the variables key2 and value2
  emit (key2,value2)

reduce ( key, nodes ):
  count = 0
  for n in nodes
      count++
  emit(key,count)
```

The second Map-Reduce is:

```
map ( node, count ):
  emit(count,1)

reduce ( key, values ):
  sum = 0
  for v in values
      sum += v
  emit(key,sum)
```

You should write the two Map-Reduce job in the Java file `src/main/java/Graph.java`. An empty `src/main/java/Graph.java` has been provided, as well as scripts to build and run this code on Expanse. **You should modify the Graph.java only**. In your Java main program, args[0] is the graph file, args[1] is the intermediate directory that holds the output of the first Map-Reduce job, and args[2] is the output directory. All the data (input, intermediate, and output) must be in text format.

Optional: Use an IDE to develop your project

If you have a prior good experience with an IDE (IntelliJ IDEA or Eclipse), you may want to develop your program using an IDE and then test it and run it on Expanse. Using an IDE is optional; you shouldn't do this if you haven't used an IDE before.

On IntelliJ IDEA, go to New→Project from Existing Sources, then choose your project1 directory, select Maven, and then Finish. To compile the project, go to Run→Edit Configurations, use + to Add New Configuration, select Maven, give it a name, use Working directory: your project1 directory, Command line: install, then Apply.

On Eclipse, you first need to install **m2e** **(https://projects.eclipse.org/projects/technology.m2e)** (Maven on Eclipse), if it's not already installed. Then go to Open File...→Import Project from File System, then choose your project1 directory. To compile your project, right click on the project name at the Package Explorer, select Run As, and then Maven install.

Documentation

- The **The Map-Reduce API** **(https://hadoop.apache.org/docs/r3.2.2/api/index.html)**. The API has two variations for most classes: `org.apache.hadoop.mapreduce` and `org.apache.hadoop.mapred`. **You should only use the classes in the package `org.apache.hadoop.mapreduce`**
- The **org.apache.hadoop.mapreduce package** **(https://hadoop.apache.org/docs/r3.2.2/api/index.html?org/apache/hadoop/mapreduce/package-summary.html)**
- The **Job class** **(https://hadoop.apache.org/docs/r3.2.2/api/org/apache/hadoop/mapreduce/Job.html)**

What to Submit

You need to submit the following files only:

```
project1/src/main/java/Graph.java
project1/output-distr/part-r-00000
project1/graph.distr.out
```