Project 5

| 11/8/2022

100/100 Points

Attempt 1

REVIEW FEEDBACK
11/8/2022

Attempt 1 Score:
100/100

Add Comment

Unlimited Attempts Allowed

Details

Graph Processing on Spark

Description

The purpose of this project is to develop a graph analysis program using Apache Spark.

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

As in the previous projects, you will develop and run your program on Expanse. Optionally, you may use your laptop to help you develop your program, l

Try Again

Using your laptop to develop your project

If you'd prefer, you may use your laptop to develop your program and then you would need to test it and run it on Expanse.

To install the project:

```
cd
wget http://lambda.uta.edu/cse6332/project5.tgz
tar xfz project5.tgz
```

To compile and run project5:

```
cd project5
mvn install
~/spark-3.1.2-bin-hadoop3.2/bin/spark-submit --class Graph --master local[2] target/*.jar small-graph.txt
```

Your result should be the same as the `small-solution.txt`.

Actually, if your laptop has enough memory, you can run your program on `large-graph.txt` too.

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 project 5 on your laptop, copy `project5.tgz` from your laptop to Expanse. Otherwise, download project5 using `wget http://lambda.uta.edu/cse6332/project5.tgz`.

Then untar project5:

```
tar xfz project5.tgz
chmod -R g-wrx,o-wrx project5
```

You can compile Graph.scala using:

```
run graph
```

Try Again

and you c

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

Your result should be the same as the `small-solution.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 graph.distr.run
```

This will work on the moderate-sized graph and will print the results to the output. It should be the same as `large-solution.txt`.

Project Description

You are asked to re-implement Project 3 (the connected components of a graph) using Spark and Scala. **Do not use Hadoop Map-Reduce**. A file `project5/src/main/scala/Graph.scala` is provided that has incomplete code that you need to complete, as well as scripts to build and run this code on Expanse. **You should modify Graph.scala only**.

Like in Project 3, your task is to write a Spark program that finds the connected components of any undirected graph and prints the size of these connected components. A connected component of a graph is a subgraph of the graph in which there is a path between any two vertices in the subgraph. As you can see in the incomplete `Graph.scala` program, the variable `graph` has type `RDD[ ( Long, Long, List[Long] ) ]`. That is, it is a dataset of vertices, where each vertex is a triple `(group,id,adj)`, where `id` is the vertex id, `group` is the group id (initially equal to id), and `adj` is the list of outgoing neighbors. For example, from the input line `8,5,6,7`, you need to return the tuple `(8,8,List(5,6,7))`. Then, during the for-loop, you need to construct a dataset which, for each vertex, it generates group candidates. For example, from the vertex `(4,8,List(5,6,7))` you generate the group candidates `(8,4), (5,4), (6,4)`, and `(7,4)`, that is, 8 can be in group 4, 5 can be in group 4, etc. Then, for each vertex id, you select the minimum group candidate. This will be the new group for this vertex at this iteration. These new groups are stored in the variable `groups`. Then, you need to reconstruct the graph with the new groups so that your program can do more iterations. After the loop, you print the sizes of each group.

What to Submit

As in the previous projects, you need to tar your project5 directory on Expanse, copy project5.tgz from Expanse to your laptop, and submit it using this

Try Again

```
project5/src/main/scala/Graph.scala
project5/graph.local.out
project5/graph.distr.out
```

Preview Unavailable

project5.tgz

Download (https://uta.instructure.com/files/23180202/download?download_frd=1&verifier=dcVBAvEBYbFFSY6cnap90jz6f8LDdAZ1NcgF6VSt)

Try Again