Programming Assignment 5
Graph Processing on Spark

Due on Thursday April 9 before midnight

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, 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 your program on SDSC Comet. Optionally, you may use your laptop or IntelliJ Idea or Eclipse to help you develop your program, but you should test your programs on Comet before you submit them.

Setting up your Project

Login into Comet and download and untar project5:

```
wget http://lambda.uta.edu/cse6331/project5.tgz
tar xfz project5.tgz
chmod -R g-wrx,o-wrx project5
```

Project Description

You are asked to re-implement Project #3 (Graph Processing) using Spark and Scala. Do not use Map-Reduce. That is, your program must find 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 from any two vertices in the subgraph. Please look at Project #3 for an example.

An empty `project5/src/main/scala/Graph.scala` is provided, as well as scripts to build and run this code on Comet. **You should modify Graph.scala only**. Your main program should take the text file that contains the graph (small-graph.txt or large-graph.txt) as an argument.

The graph can be represented as `RDD[ ( Long, Long, List[Long] ) ]`, where the first Long is the graph node ID, the second Long is the group that this vertex belongs to (initially, equal to the node ID), and the List[Long] is the adjacent list (the IDs of the neighbors). Here is the pseudo-code:

```
var graph = /* read the graph from args(0); the group of a graph node is set to the node ID */

for (i <- 1 to 5)
   graph = graph.flatMap{ /* associate each adjacent neighbor with the node group number + the node itself with its group number*/ }
                .reduceByKey( /* get the min group of each node */ )
                .join( /* join with the original graph */ )
                .map{ /* reconstruct the graph topology */ }

/* finally, print the group sizes */
```

For example, for the node `(20,6,List(22,23,24))`, the flatMap must return the sequence `Seq((20,6),(22,6),(23,6),(24,6))`. The output (group sizes) must be sent to the output, not to a file.

You can compile Graph.scala using:

```
run graph.build
```

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

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

Your result should be the same as the solution in the Project #3 example. 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.

Optional: Use your laptop to develop your project

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

To install the project:

```
cd
wget http://lambda.uta.edu/cse6331/project5.tgz
tar xfz project5.tgz
```

To compile and run project5:

```
cd project5
mvn install
rm -rf output
~/spark-1.5.2-bin-hadoop2.6/bin/spark-submit --class Graph --master local[2] graph.jar small-graph.txt
```

What to Submit

You need to submit the following files only:

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

Submit Programming Assignment #5:

Select a file: Choose File no file selected

Send File

Last modified: 04/02/2020 by Leonidas Fegaras