Project 8

New Attempt

Due Dec 2 by 11:59pm **Points** 100 **Submitting** a file upload

Graph Partition using GraphX

Description

The purpose of this project is to write the graph partition program using Pregel on Spark GraphX.

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 your program on SDSC Comet. Optionally, you may use your laptop 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 project8:

```
wget http://lambda.uta.edu/cse6331/project8.tgz
tar xfz project8.tgz
chmod -R g-wrx,o-wrx project8
```

Then, **remove** the following 3 lines that you added in .bashrc in Project7:

```
export JAVA_HOME=$SW/java-se-8u41-ri
export HADOOP_HOME=$SW/hadoop-2.6.5
export HIVE_HOME=$SW/apache-hive-2.1.0-bin
```

Logout and login again to apply the changes. Look at the example project8/example/src/main/scala/SSSPExample.scala which implements the single source shortest path.

Project Description

You are asked to re-implement Project #5 (Graph Processing) using Pregel on Spark GraphX. That is, your program will partition your graph into 10 clusters. An empty `project8/src/main/scala/Partition.scala` is provided, as well as scripts to build and run this code on Comet. **You should modify Partition.scala only**. **You should use the pregel method from the GraphX Pregel API (https://spark.apache.org/docs/latest/graphx-programming-guide.html#graph-builders) only to write your code**. Your main program should take the text file that contains the graph (small-graph.txt or large-graph.txt) as an argument and print the results to the output. The stopping condition is when the number of repetition reaches 5.

You can compile Partition.scala using:

```
run partition.build
```

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

```
sbatch partition.local.run
```

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 partition.distr.run
```

This will work on the moderate-sized graph and will print the results to the output.

The following pseudo-code to do graph clustering using Pregel:

1. Read the input graph and construct the RDD of edges
2. Use the graph builder Graph.fromEdges to construct a Graph from the RDD of edges
3. Access the VertexRDD and change the value of each vertex to be the -1 except for the first 5 nodes (these are the initial cluster number)
4. Call the Graph.pregel method in the GraphX Pregel API to calculate the new cluster number for each vertex and propagate this number to the neighbors. For each vertex, this method changes its cluster number to the max cluster number of its neighbors only if the current cluster number is -1.
5. Group the graph vertices by their cluster number and print the partition sizes (you can use Spark RDD methods like in Project 5)

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. First, remove the 4 lines that you added in .bashrc in Project 7. Logout and login again to apply the changes.

To install the project:

```
cd
wget http://lambda.uta.edu/cse6331/project8.tgz
tar xfz project8.tgz
```

To compile and run project8:

```
cd project8
mvn install
~/spark-3.1.2-bin-hadoop3.2/bin/spark-submit --class Partition --master local[2] target/cse6331-project8-0.1.jar small-graph.txt
```

You should modify and run your programs in local mode until you get the correct result (as in small-solution.txt).

Look at the example `example/src/main/scala/SSSPExample.scala` which implements the single source shortest path. You can compile and run it using:

```
cd example
mvn install
```

```
~/spark-3.1.2-bin-hadoop3.2/bin/spark-submit --class SSSPExample --master local[2] target/cse6331-project8-0.1.jar
```

Documentation

You can learn more about GraphX at:

- **GraphX Programming Guide** **(https://spark.apache.org/docs/3.1.2/graphx-programming-guide.html)**
- **GraphX Pregel** **(https://spark.apache.org/docs/3.1.2/graphx-programming-guide.html#pregel-api)**
- **Graph API** **(https://spark.apache.org/docs/3.1.2/api/scala/org/apache/spark/graphx/Graph.html)**
- **GraphOps API** **(https://spark.apache.org/docs/3.1.2/api/scala/org/apache/spark/graphx/GraphOps.html)**
- **EdgeTriplet API** **(https://spark.apache.org/docs/3.1.2/api/scala/org/apache/spark/graphx/EdgeTriplet.html)**

What to Submit

Zip and submit your project8 directory, which must contain the following files:

```
project8/src/main/scala/Partition.scala
project8/partition.local.out
project8/partition.distr.out
```