Cloud Computing (INGI2145) - Lab Session 5

Muhammad Bilal, Marco Canini

1. Background

In this lab session, you will learn how to code a simple Giraph Job for Hadoop and how to run the job in the INGI2145-vm. We have taken a look at an example Giraph program that counts the number of neighbors that every node has (within the n-hop distance). We will look at the setup involved in creating and running a Giraph job for this program. The code is available in the repo in a file called `NHopNeighbors.java` and you should take a look at it before proceeding with the exercise. But first let's prepare the VM for this lab session…

To setup the VM for this exercise, you will need to start by pulling from the course Git repository to obtain the example code and the skeleton code for the exercise. The code is located inside the path `/labs/5_giraph/`. Be sure to check the `README.md` file to learn how to run the code and generate project files for IDEs (Eclipse, IDEA).

2. Running the Example Program

1. In order to run the example program go the the `skeleton` directory in the lab folder.
2. Run `prep.sh`.
    - This script will stop any dfs or yarn process already running (which might not be the case if you are running this for the first time but it's useful).
    - It removes the HDFS directory and formats the name node (this step is only necessary when you run into problems with start YARN or HDFS)
    - Starts HDFS and YARN again
    - Creates an input directory in HDFS and copy the graph file to HDFS
3. Now run `./gradlew fatJar` to build the program. This might take some time (~5-7 mins) since the jar will be created with **all** its dependencies.
4. Next execute the `./run-nhop.sh` script to run the program. This submits the Giraph job to your Hadoop instance. You should expect to see some warnings that can be ignored. At the end, you will see the statistics about the job you executed. Look for the lines below "Giraph Stats" and "Giraph Timers". Notice that the computation within Giraph supersteps is rather fast O(~ms) although the shutdown step is much longer O(~s).
5. To see the output of the file, you can copy it to your local file system using the copyToLocal argument of hdfs command like this:

```
$HADOOP_HOME/bin/hdfs dfs -copyToLocal <path hdfs> <local path>
```

Note: the output paths are specified in the run script. For example, for `run-nhop.sh`, the HDFS path is /`output/nhopneighbors`.

3. Programming exercise

In this lab session, you will be required to implement two algorithms in Giraph. Your task will be to write only the compute part of the program. The rest of the program is provided in the skeleton code in the path `src/be_uclouvain_ingi2145_lab05/`. These algorithms have been discussed in the lectures.

3.1 Input and Output Formats

You'll take as input a text file containing an example graph formatted as vertices with adjacency lists. We already provide the code to load the input file. The format of the input file is:

```
[vertexId, vertexValue, [[dstId1,edgeValue1],[dstId2,edgeValue2]]]
```

Where the adjacency list has the outgoing edges connected to a vertex and the destination vertex connected to those edges.

We will use this format for all our programs (for simplicity). The only difference from the perspective of semantics of input is that we consider the vertex value to have different meaning depending on the algorithm. In n-hops neighbor count, the vertex value represents the count of neighbors. In the shortest path computation algorithm it will represent the distance of the vertex from the source node and in PageRank the vertex value will be the page rank of that vertex.

The output format will be similar to the input format in our case (again for simplicity, even though this output format is not very efficient for our use cases), except that the vertex value will have the finished results that we want i.e., shortest distance from the source node or the page rank.

3.2 Single Source Shortest Path Algorithm

In Single Source Shortest Path algorithm, we want to compute the distance of all vertices in a graph from a given source node. The source node can be provided as an integer value with a command line argument named `ca` as shown in the script `run-sssp.sh`. In this algorithm we will take into account the edge values in the adjacency list and use the value as the distance between the two nodes that they connect.

Write your code in the `compute()` function of `SingleSourceShortestPath.java`
Run `./gradlew fatJar` to build your program and use the script `run-sssp.sh` to run.

3.3 Page Rank Algorithm

In PageRank algorithm, we want to compute the rank of all vertices in a graph based on the incoming edges and the nodes from which these edges are. In this algorithm the edge values in the adjacency list and use the value as the distance between the two nodes that they connect.

Write your code in the `compute()` function of `PageRank.java`
Run `./gradlew fatJar` to build your program and use the script `run-pagerank.sh` to run.