**Athens University of Economics and Business** Dept. of Informatics
**M.Sc. Program in Data Science**
**INF322: Social Network Analysis** Fall 2022
**Instructor: Katia Papakonstantinopoulou**

# Project 3: Apache Giraph

## An iterative graph processing system built for high scalability

*Due 11:59pm EEST November 27, 2022*

## General Instructions

Your answers should be as concise as possible.

**Submission instructions:** You should submit your report and code in a compressed directory uploaded via `https://eclass.aueb.gr`

*Submitting answers:* Prepare a report on your project in a single PDF file named *project3.pdf*

*Submitting code:* Prepare a `.txt` file named according to the instructions with the answer to the first problem and the completed Java files `SimpleLabelPropagationComputation.java` and `GiraphAppRunner.java` for the second problem.

## Problems

In this project you will be using and developing algorithms with Apache Giraph, the open-source counterpart of Google's Pregel, that is mainly contributed by Facebook. Apache Giraph's source code, Apache Hadoop and everything you will need for this project is preinstalled in the Virtual Machine provided by this course.[1] An Eclipse Java Project configured to execute Giraph jobs locally is also offered to you to allow for faster development.[2] Alternatively, you can proceed with installing Apache Giraph to your own platform by following the guidelines of this link: `http://giraph.apache.org/quick_start.html`.

[1] `https://goo.gl/CGm1YR`
[2] `https://drive.google.com/open?id=14ENVKXZpINZXvbdGhuHVIMU5zLgOLvTh`

# 1 Use Apache Giraph SimpleShortestPaths algorithm

The first part of this project requires that you familiarize yourselves with executing Apache Giraph *jobs*. The instructions will guide you through the entire procedure and you will have to submit the output of the execution.

First you need to start the *Hadoop Filesystem* and *Map Reduce*. To this end you will need to use the respective *scripts* that are placed in the hadoop installation directory (/usr/local/hadoop/).

**CAREFUL**: Hadoop related tasks must be executed through a dedicated hadoop user (hduser in the VM). To switch users you have to issue the `su` command:

```
$ su hduser
```

The scripts you need to run are under the `bin/` directory and are:

1. `start-dfs.sh`
2. `start-mapred.sh`

**CAREFUL**: The scripts must be initiated in this order and terminated in the reverse.

Then, you can verify that everything is up and running by issuing the `jps` command. You should see the following processes running:

- NameNode
- JobTracker
- DataNode
- SecondaryNameNode
- Jps
- TaskTracker

Now, you can use the web browser to see a web interface of both the filesystem and the map reduce administration. The local addresses are already bookmarked in the VM's browser but are also listed below:

- Filesystem: `http://localhost:50070`
- Map Reduce Administration: `http://localhost:50030`

If you navigate in the filesystem after following the first link, you will find that a sample graph is already present there (tiny_graph.txt). This is the graph that you have to use for the first part of this project.

In particular, you will execute the `SimpleShortestPaths` algorithm that is already implemented in Apache Giraph. To do that, run (as hduser) the following command:

```
/usr/local/hadoop/bin/hadoop jar /home/sna/eclipse/workspace/giraph/giraph-examples/target/giraph-examples-1.1.0-for-hadoop-1.2.1-
jar-with-dependencies.jar org.apache.giraph.GiraphRunner org.apache.giraph.examples.SimpleShortestPathsComputation
-vif org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputFormat -vip /user/hduser/input/tiny_graph.txt
-vof org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op /user/hduser/output/shortestpaths -w 1
```

This command provides Hadoop with a jar file and specifies the class of the jar file that is to be executed (`org.apache.giraph.GiraphRunner`). This class is provided with another Giraph class
(`org.apache.giraph.examples.SimpleShortestPathsComputation`)
that extends: `org.apache.giraph.graph.BasicComputation`.
Then, some arguments specify the vertex input format (-vif), the vertex input path (-vip), the vertex output format (-vof), the output path (-op) and the number of workers (-w). As this is a pseudo-distributed hadoop installation with only one worker, you cannot increase the number of workers to more than one.

After you have executed this command, and if everything is done according to the instructions, there will be a file in your HDFS in the path:
`"/user/hduser/output/shortestpaths"`, with the result of your computation. For this part of the project you have to submit this file. Browse through the hadoop filesystem and retrieve the file using your web browser. Alternatively, you can copy the file to your local filesystem using `hadoop dfs` command.

In addition, you should rename the file you retrieved to match the task id that saved your results. To retrieve the task id, browse through the job tracker to find your completed job, click on its id, then click on the completed map tasks, and there you will see the id of the task that saved your vertices. **Use it to name your file**. This will familiarize you with browsing through the details of a job's tasks where you can access useful information regarding the tasks you initiated, such as logs and errors.

## 2 Implement Label Propagation Community Detection with Apache Giraph

The `Label Propagation` algorithm for community detection [1] builds on the following idea: Suppose that a node $x$ has neighbors $x_1, x_2, ..., x_k$ and that each neighbor carries a label denoting the community to which they belong to. Then $x$ determines its community based on the labels of its neighbors.

For the second part of this homework, you will have to implement an iterative Pregel-like algorithm that assigns to all nodes of the graph a community id, according to the algorithm of [1].

You have to create a Java class that extends:
`org.apache.giraph.graph.BasicComputation`.
Such a class is provided with this homework and already takes care of certain implementation details. You should complete the implementation of this project with the following functionality:

- Every vertex should send an initial message to all its neighbors with the label of its community. Initially, every vertex forms a community on its own, so the vertex id can be used as the community label.
- While a vertex receives messages, it adopts the community of the majority of its neighbors. If there is a tie, the vertex adopts the smallest value. If there is only one neighbor, the vertex adopts its value if it's smaller than its current value.
- If a vertex adopts a new value, it will propagate the new value to its neighbors.
- The computation will terminate if no new values are assigned or the maximum number of iterations is reached. You can consider any number between 30 and 80 as the number of maximum iterations.

Your algorithm should be able to detect the communities of any input graph. However, a sample graph (input_graph.txt) is provided with this homework, to enable discussion of results on a common ground and ease development.

There are two alternatives you can follow for the second part of this homework:

If you plan to submit your algorithm for execution to hadoop, you should copy all Java files to their respective giraph directory and build giraph. For instance, `SimpleLabelPropagationComputation.java` should go to:

/home/sna/eclipse/workspace/giraph/giraph-examples/src/main/java/org/apache/giraph/examples/

To build Apache Giraph after performing changes to the source code, you have to execute the following from a terminal in the directory of giraph

(/home/sna/eclipse/workspace/giraph) as the `sna` user:
`mvn package -DskipTests`

For your convenience, an Eclipse Java Project is available here:
`https://drive.google.com/open?id=14ENVKXZpINZXvbdGhuHVIMU5zLgOLvTh`.

This project allows for local execution of giraph jobs, enabling you to speed-up development.
To use it, you should copy the the project into the '/home/sna/eclipse/workspace' directory and then import it into Eclipse (File ->Import ->Existing Projects Into Workspace). Then you can execute giraph jobs by running GiraphAppRunner.java as a Java Application. This class is configured to execute the example of the first part of this homework. You should make some minor adjustments to execute the code of the second part. Please note that some errors during execution using GiraphAppRunner are expected and should be ignored. Successful execution will create a directory containing a file named '_SUCCESS'.

**Bonus Question**: Illustrate the output of your algorithm using Gephi, given a graph of your choice of more than 100 vertices as input.

## References

[1] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.